Advancing Graph Neural Networks: Expressivity Enhancement, Information Flow Optimization, and Dynamic Process Modelling

Asela Hevapathige

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

March 2026

Certificate Of Authorship/ Originality

I, Asela Hevapathige, declare that this thesis is submitted in fulfilment of the requirements for the degree of Doctor of Philosophy, in the School of Computing at the Australian National University.

I certify that this thesis is my own original work, except where otherwise indicated. I also certify that this document has not been submitted for obtaining an award at any other academic institution.

To the people of Sri Lanka — with heartfelt gratitude for the gift of free education

Acknowledgments

This thesis would not have been possible without the support, guidance, and encouragement of many people. First and foremost, I wish to express my deepest gratitude to my former primary supervisor, Associate Professor Qing Wang, who passed away in the final stages of my PhD journey. Her continuous support, guidance, and patience throughout my doctoral research shaped me as a researcher in ways I am still discovering. She was a tremendous mentor who believed in my work even when I had doubts, and her constant feedback and encouragement made this thesis possible. It has been a great privilege to have worked under her supervision, and I will forever be grateful for her immense support during my PhD study. Her legacy continues to inspire my work.

I am deeply grateful to my current primary supervisor, Dr. Ahad N. Zehmakan, who has been there throughout my PhD and stepped in during a difficult transition. His wisdom and guidance have been invaluable in helping me complete this thesis. Whenever I faced challenges, he was always there with support and advice, and I am truly thankful for his mentorship. I would also like to thank my co-supervisor, Dr. Asiri Wijesinghe, for his generous assistance and insights throughout this journey.

I extend my sincere thanks to all my colleagues at the School of Computing for creating a stimulating and supportive research environment. I am also grateful to the administrative team at the School of Computing, the HDR team, and the IT support staff at ANU for their assistance and patience with my countless queries over the years.

To my parents and my brother, thank you for your unconditional love, support, and understanding throughout this long journey. Your belief in me has been my anchor. Finally, I thank all my friends from Sri Lanka, whose friendship and encouragement from afar have kept me going through the challenges of doctoral research.

Publications

Contributions from work presented in this thesis have been published across multiple peer-reviewed top-tier conferences and journals. A list of publications in chronological order is given below:

- **A. Hevapathige, A. N. Zehmakan, and Q. Wang.** Depth-Adaptive Graph Neural Networks via Learnable Bakry-Émery Curvature. *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025 (Hevapathige et al., 2025b)
  The concepts and formulations of this curvature-based Graph Neural Network model, theoretical analysis, model architecture and evaluations are presented in **Chapter 4**.

- **A. Hevapathige, Q. Wang, and A. N. Zehmakan.** DeepSN: A Sheaf Neural Framework for Influence Maximization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025 (Hevapathige et al., 2025a)
  The concepts and formulations of this sheaf neural framework, theoretical analysis, model architecture and evaluations are presented in **Chapter 7**.

- **A. Hevapathige, A. Wijesinghe, A. N. Zehmakan.** Beyond Fixed Depth: Adaptive Graph Neural Networks for Node Classification Under Varying Homophily. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026 (Hevapathige et al., 2026)
  The concepts and formulations of this adaptive depth Graph Neural Network model, theoretical analysis, model architecture and evaluations are presented in **Chapter 5**.

- **A. Hevapathige and Q. Wang.** Permutation-Invariant Graph Partitioning: How Graph Neural Networks Capture Structural Interactions? *Neural Networks*, 2026 (Hevapathige and Wang, 2026)
  The concepts and formulations of this expressive Graph Neural Network model, theoretical analysis, model architecture and evaluations are presented in **Chapter 3**.

- **A. Hevapathige, A. Wijesinghe, and A. N. Zehmakan.** Graph Neural Diffusion via Generalized Opinion Dynamics. Under review.
  The concepts and formulations of this diffusion Graph Neural Network model, theoretical analysis, model architecture, and evaluations are presented in **Chapter 6**.

Abstract

Graph Neural Networks (GNNs) have emerged as powerful tools for learning representations from graph-structured data, yet fundamental limitations constrain their effectiveness across diverse applications. This thesis addresses three critical challenges in graph representation learning: enhancing structural expressivity, optimizing information propagation, and modeling dynamic processes.

We first tackle the expressivity limitations of standard message-passing neural networks, which are bounded by the 1-Weisfeiler-Lehman test. Through permutation-invariant graph partitioning, we develop Graph Partitioning Neural Networks that capture complex structural interactions by explicitly modeling how different graph components interact. Our theoretical analysis demonstrates that this approach efficiently approaches 3-WL expressivity while maintaining computational tractability.

Moving beyond uniform aggregation strategies, we develop two complementary frameworks for adaptive depth allocation in GNNs. The first integrates learnable Bakry-Émery curvature to capture both structural properties and diffusion dynamics, establishing that nodes with higher curvature require fewer message-passing iterations for effective representation learning. The second provides theoretical foundations connecting neighborhood characteristics to optimal aggregation strategies under varying homophily conditions, revealing that aggregation requirements depend on the interplay between same-label neighbors, opposite-label neighbors, and local degree distribution. Both frameworks eliminate the need for separate architectures for homophilic and heterophilic graphs.

We then address limitations of existing diffusion-based GNNs through two novel frameworks. The Generalized Opinion Dynamics Neural Framework unifies multiple opinion dynamics models, incorporating node-specific stubbornness, dynamic neighborhood influence, and structural regularization to enable diverse convergence configurations including single consensus, multi-consensus, and individualized consensus. For influence maximization, we develop Deep Sheaf Networks that formulate propagation as a sheaf diffusion-reaction process, introducing pointwise and coupled dynamics operators to capture both intrinsic node transformations and neighborhood interactions across progressive and non-progressive diffusion models. A subgraph-based optimization strategy reduces the combinatorial search space while accounting for overlapping influence between nodes.

In summary, the theoretical contributions of this thesis establish rigorous foundations for understanding structural interactions, information flow optimization, and dynamic process modeling in graph neural networks. Extensive experiments across node classification, graph classification, graph regression, influence estimation, and influence maximization tasks demonstrate that our approaches consistently outperform existing methods.

Contents

Abstract **v**

Notations **xvi**

1 Introduction **1**
1.1 Graph Representation Learning . . . 2
1.2 Graph Neural Networks . . . 3
1.3 Research Objectives and Contributions . . . 5
1.3.1 Permutation-Invariant Graph Partitioning: How Graph Neural Networks Capture Structural Interactions? . . . 6
1.3.2 Depth-Adaptive Graph Neural Networks via Learnable Bakry-Émery Curvature . . . 8
1.3.3 Beyond Fixed Depth: Adaptive Graph Neural Networks Under Varying Homophily . . . 9
1.3.4 Graph Neural Diffusion via Generalized Opinion Dynamics . . . 10
1.3.5 DeepSN: A Sheaf Neural Framework for Influence Maximization 11
1.4 Thesis Roadmap . . . 12
1.5 Thesis as a Unified Research Program . . . 14

2 Chapter 2 **15**
2.1 Basic Notations and Definitions . . . 15
2.2 Graph Neural Networks . . . 15
2.2.1 Spatial GNNs . . . 15
2.2.2 Diffusion GNNs . . . 17
2.3 Graph-based Downstream Tasks . . . 18
2.3.1 Node Classification . . . 18
2.3.2 Graph Classification . . . 18
2.3.3 Graph Regression . . . 18
2.3.4 Influence Maximization . . . 19
2.3.5 Influence Estimation . . . 19
2.4 Expressivity of GNNs . . . 19
2.4.1 MPNNs and Weisfeiler-Lehman (1-WL) Test . . . 19
2.4.2 MPNNs Beyond 1-WL Test . . . 20
2.5 Homophilic and Heterophic Graphs . . . 21
2.6 Oversmoothing in GNNs . . . 23
2.7 Curvature and GNNs . . . 24
2.7.1 Curvature Notions . . . 24

2.7.1.1 Ollivier-Ricci Curvature . . . 24
2.7.1.2 Forman-Ricci Curvature . . . 25
2.7.2 Curvature-based GNNs . . . 25
2.8 Opinion Dynamics and GNNs . . . 25
2.8.1 Opinion Dynamic models . . . 26
2.8.1.1 French-DeGroot (FD) Model . . . 26
2.8.1.2 Friedkin-Johnsen (FJ) Model . . . 26
2.8.1.3 Hegselmann–Krause (HK) Model . . . 27
2.8.2 Opinion Dynamic-based GNNs . . . 27
2.9 Adaptive GNNs . . . 27
2.10 GNNs for Influence Maximization . . . 28

3 Chapter 3 **30**
3.1 Overview . . . 30
3.2 Graph Partitioning and Graph Isomorphism . . . 32
3.2.1 Preliminaries . . . 32
3.2.2 Graph Partitioning Scheme . . . 33
3.3 Graph Partitioning Neural Networks . . . 34
3.4 Theoretical Analysis . . . 36
3.4.1 Expressivity Analysis . . . 36
3.4.2 Complexity Analysis . . . 40
3.5 Practical Choices of Partitioning Schemes . . . 40
3.6 Experimental Design . . . 42
3.6.1 Datasets . . . 42
3.6.2 Baselines . . . 42
3.6.3 Experimental Setups . . . 43
3.6.4 Hyper-parameters . . . 44
3.7 Results and Discussion . . . 44
3.7.1 Graph Classification . . . 44
3.7.2 Graph Regression . . . 45
3.7.3 Node Classification . . . 45
3.7.4 Impact of Different Graph Partitioning Schemes . . . 46
3.7.5 Impact of Different Structural Interactions . . . 49
3.7.6 Impact of Individual GPNN Components . . . 50
3.8 Summary . . . 51

4 Chapter 4 **52**
4.1 Overview . . . 52
4.2 Bakry-Émery Curvature . . . 54
4.2.1 Preliminaries . . . 54
4.2.2 Formulation . . . 54
4.3 Curvature and Information Propagation . . . 55
4.4 Depth-Adaptive Graph Neural Networks . . . 57
4.4.1 Adaptive Layer Depth Mechansim . . . 57

4.4.2 Learning to Estimate Curvature . . . 58
4.4.3 GNN Training Loss Function . . . 59
4.5 Theoretical Analysis . . . 60
4.5.1 Complexity Analysis . . . 60
4.5.2 Feature Distinctiveness and Curvature . . . 60
4.6 Experimental Design . . . 62
4.6.1 Datasets . . . 62
4.6.2 Baselines . . . 63
4.6.3 Evaluation settings . . . 63
4.6.4 Model Hyper-Parameters . . . 64
4.7 Results, and Discussion . . . 64
4.7.1 Node Classification . . . 64
4.7.2 Node Regression . . . 65
4.7.3 Graph Classification . . . 65
4.7.4 Graph Regression . . . 66
4.7.5 Hyper-Parameter Analysis . . . 66
4.7.6 Oversmoothing Analysis . . . 67
4.7.7 Comparison with Existing Curvature Notions . . . 67
4.7.8 Runtime and Parameter Complexity analysis . . . 68
4.7.9 Visualization Analysis . . . 69
4.8 Summary . . . 70

5 Chapter 5 **71**
5.1 Overview . . . 71
5.2 Theoretical Analysis . . . 72
5.2.1 Preliminaries . . . 73
5.2.2 Class Concepts . . . 73
5.2.3 GNN Architecture . . . 73
5.2.4 Assumptions . . . 74
5.2.5 Aggregation Effect Analysis . . . 74
5.2.6 Multi-Layer Analysis . . . 77
5.2.7 Limitations of our Theoretical Framework . . . 79
5.3 Adaptive Depth Graph Neural Networks . . . 80
5.3.1 Stopping Depth Assignment . . . 81
5.3.2 Message Passing Framework . . . 81
5.3.3 Depth Benefit Metric Calculation . . . 82
5.3.4 Training Objective . . . 83
5.3.5 Fast Variant . . . 83
5.3.6 Complexity Analysis . . . 83
5.4 Experimental Design . . . 84
5.4.1 Datasets . . . 84
5.4.2 Baselines . . . 84
5.4.3 Evaluation Setting . . . 85
5.4.4 Model Hyper-Parameters . . . 85

5.5 Results and Discussion . . . 85
5.5.1 Node Classification . . . 85
5.5.2 Case Study . . . 85
5.5.3 Oversmoothing Analysis . . . 87
5.5.4 Hyperparameter Sensitivity Analysis . . . 88
5.5.5 Scalability Analysis . . . 89
5.5.6 Visualization Analysis . . . 90
5.6 Summary . . . 90

6 Chapter 6 **93**
6.1 Overview . . . 93
6.2 A Generalized Opinion Dynamics Framework for Graph Neural Diffusion . . . 95
6.2.1 Preliminaries . . . 95
6.2.2 Diffusion Mechanism in GODNF . . . 95
6.2.2.1 Matrix Form . . . 96
6.2.2.2 A Simplified Variant: $\text{GODNF}_{\text{Static}}$ . . . 97
6.2.3 Mapping of GODNF Components to Opinion Dynamic Models 97
6.2.4 Bounded Weight Evolution . . . 98
6.2.5 Convergence Conditions of GODNF . . . 98
6.2.6 Convergence-Preserving Regularization . . . 99
6.2.6.1 Regularization Term . . . 99
6.2.6.2 Training Loss . . . 100
6.3 Theoretical Analysis . . . 100
6.3.1 Complexity Analysis . . . 100
6.3.2 Representation of Diverse Convergence Configurations . . . 100
6.3.2.1 Single Consensus . . . 101
6.3.2.2 Multi Consensus . . . 101
6.3.2.3 Individualized Consensus . . . 102
6.4 Experimental Design . . . 103
6.4.1 Datasets . . . 103
6.4.2 Baselines . . . 104
6.4.3 Evaluation settings . . . 104
6.4.4 Model Hyperparameters . . . 105
6.4.5 Implementation Details . . . 105
6.5 Results and Discussion . . . 105
6.5.1 Node Classification . . . 105
6.5.2 Influence Estimation . . . 106
6.5.3 Robustness to adversarial attacks . . . 107
6.5.4 Oversmoothing Analysis . . . 109
6.5.5 Impact of Different Components . . . 109
6.5.6 Parameter Analysis . . . 110
6.5.7 Scalability Analysis . . . 111
6.5.8 Case Study . . . 112

6.6 Summary . . . 113

7 Chapter 7 **115**
7.1 Overview . . . 115
7.2 DeepSN Framework . . . 117
7.2.1 Learning to Estimate Influence . . . 117
7.2.1.1 Topological Sheaf Diffusion . . . 117
7.2.1.2 Sheaf Reaction Diffusion . . . 119
7.2.2 Sheaf GNN Training . . . 122
7.2.3 Optimizing Seed Selection . . . 123
7.3 Theoretical Analysis . . . 124
7.3.1 Complexity Analysis . . . 124
7.3.2 Node Feature Separability . . . 124
7.4 Experimental Design . . . 126
7.4.1 Datasets . . . 127
7.4.2 Experimental Setups . . . 127
7.4.3 Baselines . . . 127
7.4.4 Hyper-parameters . . . 128
7.4.5 Computational Resources . . . 128
7.5 Results and Discussion . . . 128
7.5.1 Influence Maximization . . . 128
7.5.2 Influence Estimation . . . 128
7.5.3 Impact of Influence Maximization Component . . . 128
7.5.4 Impact of Layer Depth and Feature Dimension . . . 130
7.5.5 Impact of Reaction Operators . . . 131
7.6 Summary . . . 131

8 Chapter 8 **133**
8.1 Summary of Contributions . . . 133
8.2 Computational Cost and Practical Trade-offs . . . 135
8.3 Cross-Method Comparison . . . 136
8.3.1 Graph Classification: GPNN vs. BEC-GNN . . . 136
8.3.2 Node Classification: BEC-GNN vs. AD-GNN vs. GODNF . . . 136
8.3.3 Influence Estimation: BEC-GNN vs. GODNF vs. DeepSN . . . 137
8.4 Limitations . . . 138
8.5 Future Work . . . 138

List of Figures

1.1 Visualization of graph representation learning that maps each node of a graph into a low-dimensional space . . . 2
1.2 Visualization of graph representation learning that maps entire graph into a low-dimensional space . . . 2

3.1 The expressivity of GPNN variants is analysed in terms of (1) *interaction type*: $E^{\star}$ (intra-edges), $E^{\diamond}$ (intra-edges and inter-edges), and $E^{\dagger}$ (all edges); (2) *expressive power*: 1-WL and 3-WL, under different permutation-invariant graph partitioning schemes $\lambda_{\perp}$, $\lambda_{\leq 1\text{-}WL}$, and $\lambda_{\geq 3\text{-}WL}$. . . . 31
3.2 Two pairs of non-isomorphic graphs, (G_1, H_1) and (G_2, H_2), are partitioned by node degrees, grouping nodes with the same degree. Boundary and partitioned subgraphs are shown, with different subgraphs highlighted in red and blue. (a) $G_1 \overset{PI}{\simeq} H_1$ and $G_1 \overset{II}{\simeq} H_1$; (b) $G_2 \overset{PI}{\simeq} H_2$, but $G_2 \overset{II}{\not\simeq} H_2$. . . . 34
3.3 A high-level workflow of λ_{core}-GPNN for an input graph. The intra-edges are marked using grey color. GPNN generates node representations by considering interactions within and between partitions. . . . 36
3.4 Node distributions concerning partitions under five graph partitioning schemes for TU Datasets. . . . 47
3.5 Node distributions concerning partitions under five graph partitioning schemes for OGB Datasets. . . . 48
3.6 Component-wise ablation study on MUTAG and PROTEINS datasets. Each bar represents model performance when a specific component is removed. . . . 50

4.1 High-level overview of the proposed model: nodes are clustered based on learned curvature, where message-passing depth is adaptively adjusted according to curvature. Both the curvature functions and the GNN are jointly optimized via a loss function. . . . 57
4.2 Model sensitivity to hyper-parameters . . . 66
4.3 Oversmoothing comparison: (a) Cora; (b) Texas. . . . 67
4.4 Curvature visualization . . . 69
4.5 2D node embedding visualization . . . 70

5.1 High-level workflow of AD-GNN: First, the depth benefit metric for each node is computed using their node profile and learned label probabilities. Then, the depth for each node is determined based on a learnable threshold mechanism. Nodes that have higher depth benefit metrics will receive more aggregation layers compared to others. 82
5.2 Case studies to empirically validate observations from Theorem 8. . . . 87
5.3 Oversmoothing comparison. 88
5.4 Sensitivity analysis of parameter λ on AD-GCN. 88
5.5 Degree distribution of datasets. 89
5.6 Layer-wise Embedding Visualization for Citeseer Dataset. Sil and CH acronyms refer to the silhouette score and the Calinski-Harabasz score, respectively. 91
5.7 Average depth benefit metric computed per degree in homophilic and heterophilic datasets. 92

6.1 High-Level Architecture of GODNF: Raw features are first transformed to retrieve initial features. These initial features are iteratively updated based on four aspects: (a) Current feature retention, (b) Initial feature attachment, (c) Neighborhood influence, and (d) Structural regularization. The model is trained in an end-to-end manner. 97
6.2 Node classification performance under different adversarial attacks. . . 108
6.3 Deep layer comparison for node classification 109
6.4 Performance comparison of GODNF component variants on Cora ML dataset 110
6.5 Sensitivity analysis of α 111
6.6 Distribution of learned node stubbornness λ_i 111
6.7 Step-wise embedding changes of GODNF over time steps for the Cora Full for node classification. 112
6.8 Visualisation of colour-coded node labels learned by GODNF for different convergence configurations. 113

7.1 The DeepSN framework consists of two phases: (a) learning to estimate influence with sheaf GNN; b) optimizing seed selection using the subgraphs, IM model, and trained sheaf GNN. 117
7.2 Performance of DeepSN for influence estimation in terms of MAE (Mean Absolute Error), compared to baseline methods, under IC, LT, and SIS diffusion models. 130
7.3 Impact of layer depth and feature dimension on DeepSN's performance 131

List of Tables

1.1 Chapter-to-contribution map summarising key theoretical insights and experimental support for each proposed method. 13

3.1 Time complexity analysis of different embedding computations, where **VE** refers to node embeddings and **IE** refers to interaction embeddings. 40
3.2 Dataset statistics for graph classification and regression 43
3.3 Dataset statistics for node classification 43
3.4 Graph classification performance is reported as accuracy (%) for TU datasets and ROC (%) for OGB datasets. For the TU datasets, settings 1 and 2 correspond to the configurations used in Xu et al. (2019) and Feng et al. (2022), respectively. For OGB datasets, we employ the setup introduced by Hu et al. (2020). The best results for each setting are highlighted in **blue** for setting 1 and **black** for setting 2. Baseline results are sourced from Feng et al. (2022), Wijesinghe and Wang (2022), and Zhao et al. (2022a). 44
3.5 Graph regression MAE for ZINC 12K dataset.The best results are highlighted in **black** and the second best results are highlighted in **blue**. For the baseline results, we refer to Bodnar et al. (2021). The methods marked by # are based on pre-selected domain-specific structural information. 45
3.6 node Classification Accuracy (%) averaged over 10 random splits. The best results are highlighted in **black**. 45
3.7 Ablation study accuracy (%) ± standard deviation (%). Highest results in each partitioning scheme are highlighted in **black**. 46
3.8 Statistics for five graph partitioning schemes used in experiments. . . . 49
3.9 Comparison of GPNN variants in graph classification accuracy (%) with different interaction types. 49

4.1 Dataset statistics for node classification and regression. 62
4.2 Dataset statistics for graph classification and regression. 63
4.3 Node classification accuracy ± std (%). The baseline results are sourced from Zhu et al. (2020); Sun et al. (2024); Brody et al. (2022); Han et al. (2023b). 64
4.4 Influence estimation performance MAE ± std under IC and LT diffusion models. 65

4.5 Graph classification accuracy ± std (%) for TU datasets, ROC ± std (%) for the OGB dataset, and graph regression MAE ± std for ZINC dataset, with baseline results sourced from Wijesinghe and Wang (2022), Feng et al. (2022), and Hu et al. (2020). 66
4.6 Ablation study on curvature notions. For node classification, accuracy ± std(%) is reported; for node regression, MAE ± std is used. The best results are highlighted in **bold**. 68
4.7 Runtime and parameter complexity comparison for node and graph classification tasks. 68

5.1 Dataset statistics. 84
5.2 Node classification accuracy ± standard deviation (%) on homophilic graphs. The best results are highlighted. Baseline results are sourced from Suresh et al. (2021); Platonov et al. (2023); Zheng et al. (2023b), and Chen et al. (2025). 86
5.3 Node classification accuracy ± standard deviation (%) on heterophilic graphs. The best results are highlighted. Baseline results are sourced from Suresh et al. (2021); Platonov et al. (2023); Zheng et al. (2023b), and Chen et al. (2025). 86
5.4 Performance comparison of AD-GCN variants. 90

6.1 Comparison of Opinion Dynamics Models and GODNF Components. Note that for FD and FJ models, we consider formulations without self-loops, hence no current feature retention. While these models incorporate neighbourhood influence, they utilise static (time-invariant) connection weights and topologies, thus marked as lacking *Dynamic Neighbourhood Influence*. Further, we consider the formulation where HK includes agent's current opinion in bounded confidence averaging. 97
6.2 Dataset statistics for node classification 103
6.3 Dataset statistics for influence estimation task 104
6.4 Node classification accuracy ± standard deviation (%) on heterophilic datasets. The best and second-best results are highlighted in blue and orange, respectively. The baseline results are sourced from Chen et al. (2025). 105
6.5 Node classification accuracy ± standard deviation (%) on homophilic datasets. The best and second-best results are highlighted in blue and orange, respectively. The baseline results are sourced from Chen et al. (2025). 106
6.6 Influence estimation (IC model) mean absolute error ± standard deviation. Lower value indicates better performance. The best and second-best results are highlighted in blue and orange, respectively. 107
6.7 Influence estimation (LT model) mean absolute error ± standard deviation. Lower value indicates better performance. The best and second-best results are highlighted in blue and orange, respectively. 107

6.8 Influence estimation (SIS model) mean absolute error ± standard deviation. Lower value indicates better performance. The best and second-best results are highlighted in blue and orange, respectively. 108
6.9 Scalability comparison for node classification. Best results are highlighted in blue. OOM refers to out-of-memory. 112

7.1 Summary of Dataset Statistics 127
7.2 Performance comparison under Independent Cascade (IC) diffusion model. - indicates out-of-memory error. The best results are highlighted in **bold**. Baseline results are sourced from Ling et al. (2023). . . 129
7.3 Performance comparison under Linear Threshold (LT) diffusion model. - indicates out-of-memory error. The best results are highlighted in **bold**. Baseline results are sourced from Ling et al. (2023). 129
7.4 Performance comparison under Susceptible-Infected-Susceptible (SIS) diffusion model. The best results are highlighted in **bold**. Baseline results are sourced from Ling et al. (2023). 129
7.5 Comparison of DeepSN with various influence maximization approaches for Cora-ML dataset. 130
7.6 Impact of reaction terms on DeepSN performance. 131

8.1 Comparison of GPNN and BEC-GNN on graph classification benchmarks. We adopt the setup outlined by Feng et al. (2022). Best results are highlighted in **bold**. 137
8.2 Cross-chapter comparison on node classification datasets. We adopt the setup outlined by Chen et al. (2025). Best results per dataset are in **bold**. 137
8.3 Comparison on influence estimation MAE at 10% seed set under IC and LT diffusion models. We adopt the experimental setup oulined by Ling et al. (2023). Best results per dataset are in **bold**. 137

Notations and Terminology

Notation	Meaning
General Mathematical Notation	
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_{\geq 0}$	Set of non-negative real numbers
$\mathbb{R}^+$	Set of positive real numbers
$\mathbb{N}$	Set of natural numbers
$[n]$	Set $\{1, 2, \ldots, n\}$
$\mathbb{I}(\cdot)$	Indicator function
$\mathbb{E}[\cdot]$	Expectation operator
$\mathrm{Var}[\cdot]$	Variance operator
$\lvert\cdot\rvert$	Cardinality of a set or absolute value
$\{\{\cdot\}\}$	Multiset notation
$\mathcal{O}(\cdot)$	Big-O notation (computational complexity)
$:=$	Definition
$\approx$	Approximately equal
$\propto$	Proportional to
$\equiv$	Equivalent
$\simeq$	Isomorphic
$\lim_{t\to\infty}$	Limit as t approaches infinity
$\frac{\partial}{\partial t}$	Partial derivative with respect to time
$\langle\cdot,\cdot\rangle$	Inner product
$\trianglelefteq$	Incidence relation (node-edge)
Sets and Graphs	
$G = (V, E)$	Graph with node set V and edge set E
$G = (V, E, X)$	Graph with node set V, edge set E, and node features X
$G = (V, E, X, y)$	Graph with node set V, edge set E, node features X and labels y
V	Set of nodes in a graph
E	Set of edges in a graph
$V(G)$	Set of nodes in graph G
$E(G)$	Set of edges in graph G
$n = \lvert V\rvert$	Number of nodes
$m = \lvert E\rvert$	Number of edges
$e = (v, u)$	Edge between nodes v and u

Continued on next page

Continued from previous page

Notation	Meaning
V_{train}	Set of training nodes
E_{train}	Set of edges between training nodes
Node Properties and Neighborhoods	
$N(v)$	Neighborhood of node v (1-hop neighbors)
$N_d(v)$	d-hop neighborhood of node v
d_v	Degree of node v
d_{max}	Maximum node degree in the graph
$\delta(u, v)$	Shortest-path distance between nodes u and v
y_v	Label for node v
N_v^+	Set of same-label neighbors of node v
N_v^-	Set of opposite-label neighbors of node v
d_v^+	Number of same-label neighbors of node v
d_v^-	Number of opposite-label neighbors of node v
$\hat{d}_v^+$	Estimated count of same-label neighbors
$\hat{d}_v^-$	Estimated count of opposite-label neighbors
Matrices and Linear Algebra	
I	Identity matrix
A	Adjacency matrix
$\tilde{A}$	Adjacency matrix with self-loops ($A + I$)
D	Degree matrix (diagonal matrix of node degrees)
$\tilde{D}$	Degree matrix corresponding to $\tilde{A}$
L	Graph Laplacian
L_g	Normalized graph Laplacian
L_F	Sheaf Laplacian
$\hat{L}_F$	Positive definite sheaf Laplacian
$\|\cdot\|$	General norm
$\|\cdot\|_1$	L_1 norm
$\|\cdot\|_2$	L_2 (Euclidean) norm
$\|\cdot\|_\infty$	Infinity norm
$\|\cdot\|_{op}$	Operator norm
$\otimes$	Kronecker product
$\odot$	Hadamard (element-wise) product
$\oplus$	Direct sum operator
Graph Neural Network Operations	
$\text{AGG}(\cdot)$	Aggregation function
$\text{UPD}(\cdot)$	Update function
$\text{CMB}(\cdot)$	Combining function
$\text{READOUT}(\cdot)$	Graph-level readout function

Continued on next page

Continued from previous page

Notation	Meaning
Graph Isomorphism and Expressivity (Chapter 3)	
$G \simeq H$	Graphs G and H are isomorphic
$G \overset{PI}{\simeq} H$	Graphs G and H are partition-isomorphic
$G \overset{II}{\simeq} H$	Graphs G and H are interaction-isomorphic
1-WL	1-dimensional Weisfeiler-Lehman test
3-WL	3-dimensional Weisfeiler-Lehman test
k-WL	k-dimensional Weisfeiler-Lehman test
$\sqsubseteq$	At least as expressive as
$\sqsubset$	Strictly more expressive than
Graph Coloring and Partitioning (Chapter 3)	
$\lambda = (\lambda_V, \lambda_E)$	Partition coloring (node and edge coloring)
λ_V	Node coloring function
λ_E	Edge coloring function
C_V	Set of node colors
C_E	Set of edge colors
$\{S_1, \ldots, S_k\}$	Set of partitioned subgraphs
φ_j	Subgraph invariant for partition j
$\Phi = \{\varphi_j\}_{j \in [1,k]}$	Family of subgraph invariants
f_Φ	Graph partitioning scheme
$V_B(G)$	Set of border nodes in graph G
$B(G)$	Boundary subgraph of G
$E^\star$	Intra-partition edges
$E^\diamond$	Inter-partition edges
$E^\dagger$	All edges
$\lambda_\perp$	Trivial partition coloring
$\lambda_\top$	Complete partition coloring
Geometric Curvature Measures (Chapter 4)	
$\kappa(v)$	Bakry-Émery curvature at node v
$\hat{\kappa}(v)$	Estimated curvature at node v
$\kappa_{true}(v)$	True curvature at node v
$\kappa_{OR}(u, v)$	Ollivier-Ricci curvature of edge (u, v)
$\kappa_{FR}(u, v)$	Forman-Ricci curvature of edge (u, v)
$\Delta f(v)$	Weighted Laplacian of function f at node v
$\Gamma(f, f)(\cdot)$	Squared gradient operator
$\Gamma_2(f, f)(\cdot)$	Convexity operator
Signal Propagation and Classification Quality (Chapter 5)	

Continued on next page

Continued from previous page

Notation	Meaning
μ_c	Class prototype for class c
μ_0, μ_1	Class prototypes for binary classification
Δ^2	Signal variance (class separation)
σ^2_{intra}	Noise variance (within-class variation)
ϵ_v	Noise term for node v
ϵ_u^+	Noise term for same-label neighbor u
ϵ_u^-	Noise term for opposite-label neighbor u
α_v	Signal preservation factor for node v
$\hat{\alpha}_v$	Estimated signal preservation factor
$\hat{\alpha}_{v,degree}$	Degree-based signal preservation factor
Q_v	Classification quality metric for node v
Q_v^n	Classification quality after n layers
ϵ_v^n	Depth benefit metric for node v at depth n
$\tilde{\epsilon}_v^n$	Normalized depth benefit metric for node v at depth n
$T(v)$	Stopping depth for node v
$t_{\max}$	Maximum allowable depth
f_δ	Learnable similarity function parameterized by δ
$\tau_\theta(t)$	Learnable threshold function parameterized by θ
y_{uv}	Same-label indicator for edge (u, v)
h_{edge}	Edge homophily
h_{node}	Node homophily

Time-Dependent Graph Processes (Chapter 6)

$w_{ij}(t)$	Influence weight from node j to i at time t
$W(t)$	Influence matrix at time t
$W^\star$	Converged influence matrix
$\Delta W(t)$	Weight adjustment at time t
$\eta(t)$	Time-dependent learning rate

Cellular Sheaves and Diffusion (Chapter 7)

$(G, \mathcal{F})$	Cellular sheaf on graph G
$\mathcal{F}$	Sheaf structure
$\mathcal{F}_v$	Node sheaf for node v
$\mathcal{F}_e$	Edge sheaf for edge e
$\mathcal{F}_{v \trianglelefteq e}$	Transformation map from node v to edge e
$C^0(G; \mathcal{F})$	Space of 0-cochains
$C^1(G; \mathcal{F})$	Space of 1-cochains
δ	Coboundary map
$\delta(x)_e$	Coboundary at edge e
$H^0(G; \mathcal{F})$	Harmonic cochains (kernel of sheaf Laplacian)
Δ_F	Sheaf diffusion operator

Continued on next page

Continued from previous page

Notation	Meaning
$\mathcal{A}(X_v(t))$	Pointwise dynamics operator
$\mathcal{R}_v(X(t), \mathcal{A}, S_t)$	Coupled dynamics operator
Φ_v^1, Φ_v^2	Reaction coefficient vectors for node v
κ_v^1, κ_v^2	Reaction parameter vectors for node v
Φ^1, Φ^2	Matrices of stacked reaction coefficients
T_ϕ	Neural network for seed selection parameterized by ϕ

Chapter 1

Introduction

Graphs are used to model many complex relationships in real-world entities in various disciplines, including chemistry, physics, computer science, biology, and sociology (Balaban, 1985; Riaz and Ali, 2011; Roberts, 2012). These graph structures capture intricate dependencies and interactions that are often non-linear and multi-dimensional in nature. For instance, in molecular chemistry, graphs represent atomic bonds and molecular configurations; in social networks, they model interpersonal relationships and information flow; while in biological systems, they capture protein-protein interactions and metabolic pathways. The versatility of graph representations lies in their ability to encode both local neighbourhood properties and global structural patterns, making them indispensable for understanding complex systems where traditional tabular data representations fall short (Hamilton, 2020).

In the past few decades, machine learning approaches have been increasingly used to analyze graph structures (Stanković et al., 2020; Zhang et al., 2021). The main reason for this is the ability of machine learning, particularly deep learning methods, to extract beneficial insights from hidden patterns in data. Traditional graph algorithms, while computationally efficient for specific tasks, often struggle with the heterogeneous and noisy nature of real-world graph data. Machine learning approaches can implicitly learn meaningful representations that capture both node features and structural information simultaneously, thereby facilitating the understanding of complex networks. This paradigm shift has enabled breakthrough applications in drug discovery, recommendation systems, fraud detection, and knowledge graph completion, where the integration of feature learning and structural reasoning proves crucial for achieving superior performance in diverse predictive tasks (Wu et al., 2020; Zhou et al., 2020a; Liang et al., 2022).

This thesis addresses fundamental challenges in graph representation learning, a core area of graph machine learning. The central problem is encoding graphs, including their nodes, edges, and structural properties, into continuous vector spaces known as embeddings. Graph Neural Networks (GNNs) have emerged as the dominant approach for graph representation learning, which employs neural architectures to learn these representations (Wu et al., 2020; Zhou et al., 2020a). Their ability to capture complex patterns in graph-structured data has positioned them as transformative tools across numerous domains.

1.1 Graph Representation Learning

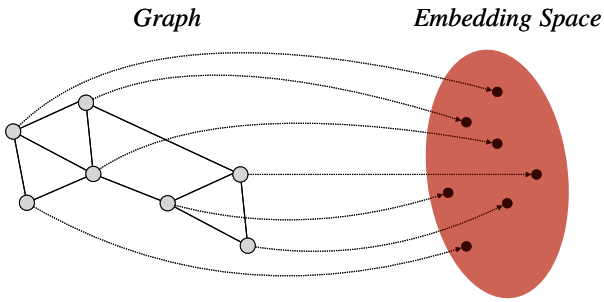


Figure 1.1: Visualization of graph representation learning that maps each node of a graph into a low-dimensional space

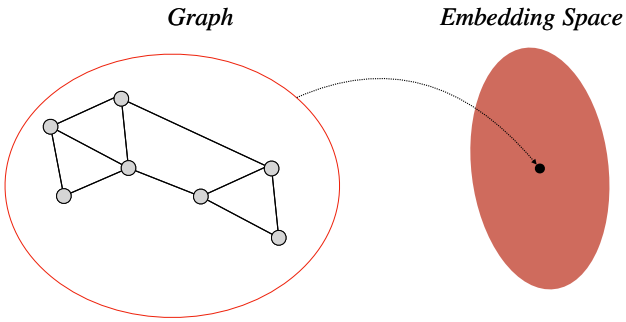


Figure 1.2: Visualization of graph representation learning that maps entire graph into a low-dimensional space

Representation learning enables machine learning systems to implicitly learn meaningful feature representations from raw data, thereby eliminating the need for manual feature engineering (Zhong et al., 2016). These techniques focus on learning optimal data transformations directly from input data, adapting to specific dataset characteristics and task objectives rather than relying on pre-designed features or human-crafted preprocessing steps. Neural architectures, such as Convolutional Neural Networks (CNNs) (Li et al., 2021b; Ketkar and Moolayil, 2021) for images and Recurrent Neural Networks (RNNs) (Medsker and Jain, 1999; Yu et al., 2019) for sequential data, implicitly discover task-relevant features that often surpass manually engineered alternatives, significantly enhancing model performance across diverse applications. However, these architectures are designed for grid-structured data and cannot directly applied on irregular graph structures, where nodes and edges exhibit non-Euclidean relationships and variable connectivity.

Existing grid-based neural architectures cannot generalize to non-Euclidean do-

mains, which has led to the emergence of graph representation learning. This subfield addresses the challenge of encoding graph-structured data by transforming complex interconnections and structural patterns into compact numerical vectors while preserving topological information. Unlike traditional approaches that rely on hand-crafted graph metrics such as node degree or centrality, these methods automatically learn the most relevant structural features for downstream tasks at both node level (Figure 1.1) and graph level (Figure 1.2)(Liang et al., 2022).

Early works in graph representation learning encompass several methodological approaches. Matrix factorization techniques decompose adjacency matrices (Singh and Gordon, 2008; Kaveh and Rahami, 2008), while random walk-based methods like DeepWalk and Node2Vec sample graph paths (Perozzi et al., 2014; Grover and Leskovec, 2016). Spectral embedding approaches leverage graph Laplacians (Belkin and Niyogi, 2001; Luo et al., 2003), and autoencoder architectures enable unsupervised learning (Berahmand et al., 2024). Additional approaches include variational methods for probabilistic representations (Jordan et al., 1999) and attention-based mechanisms that weight structural importance (Sun et al., 2023a; Soydaner, 2022).

Despite their diversity, all methods face shared challenges stemming from the irregular structure of graphs: variable neighbourhood sizes, complex structural relationships, and the absence of standard operations like convolution. These challenges motivated the development of Graph Neural Networks, which leverage deep learning principles to enable end-to-end, task-specific learning directly from graph structure.

This thesis focuses on Graph Neural Networks (Liang et al., 2022; Wu et al., 2020), a distinctive family of approaches that leverages neural network principles. Fundamentally, GNNs harness the power of neural architectures to create learning models capable of complex nonlinear mappings, adaptive feature learning, and multi-layer representations. This neural network foundation provides GNNs with greater expressiveness and learning capacity compared to other graph representation techniques.

1.2 Graph Neural Networks

Graph Neural Networks (GNNs) emerged from the challenge that traditional neural architectures, while highly effective for grid-like data such as images and sequences, struggle to handle the irregular and relational nature of graph-structured information. Sperduti and Starita (Sperduti and Starita, 1997) introduced the concept of complex recursive neurons, which extend traditional neural networks to handle structured data like graphs and trees, marking one of the earliest applications of neural networks to graph structures. Subsequently, many researchers expanded on this idea to develop more sophisticated approaches for graph learning. The formal conceptualization of GNNs was established by Gori et al. (Gori et al., 2005), which laid the theoretical foundation for learning node representations through iterative neighborhood information propagation. Building on these foundational concepts, researchers

introduced various enhancements to improve the learning process in graphs.

The evolution of GNN architectures has led to diverse methodological approaches, which can be broadly categorized based on how they implement convolution operations on graph structures and process the underlying connectivity patterns (Lu et al., 2025). These different paradigms each offer unique advantages and trade-offs in terms of computational efficiency, theoretical foundations, and practical applicability.

There are three main classes of GNNs: spatial, spectral, and diffusion-based models. The first major class, known as spatial GNNs, operates directly on the node level of the graph. These models perform convolution-like operations through message-passing mechanisms between connected nodes (Puri et al., 2025). Spatial GNNs focus on aggregating information from local neighborhoods, where each node updates its representation by combining features from its immediate neighbors through learnable transformation functions. This spatial paradigm addresses the fundamental challenge of maintaining permutation invariance (Shayestehfard, 2023) while working with irregular graph topologies, ensuring consistent node representations regardless of node ordering. Early spatial methods (Li et al., 2016) established gating mechanisms similar to those found in recurrent networks (Cho et al., 2014), allowing nodes to selectively incorporate information from different neighbors based on learned importance weights. Subsequent developments introduced attention mechanisms that further refined this selective information aggregation, enabling nodes to dynamically determine the relative importance of their neighbors during the representation update process (Veličković et al., 2018; Brody et al., 2022; Sun et al., 2023a). Modern spatial architectures have also incorporated sophisticated techniques, such as residual connections (Liu et al., 2021a; Chen et al., 2024; Kelesis et al., 2025) and normalization techniques (Cai et al., 2021; Chen et al., 2023a; Scholkemper et al., 2025) to improve training stability and model expressiveness. The computational efficiency of spatial approaches has made them particularly attractive for large-scale applications, especially when combined with sampling strategies that limit neighborhood sizes during training and inference (Hamilton et al., 2017; Liu et al., 2021b).

Complementing spatial methods, spectral GNNs approach the problem from a frequency-domain perspective. These models leverage principles of graph signal processing by transforming data into the spectral domain via eigendecomposition of graph matrices (Balcilar et al., 2021; Bo et al., 2023). These approaches define convolution operations using spectral filters that operate on the graph's eigenspace, naturally inheriting permutation invariance from the mathematical properties of graph spectra (Wang and Zhang, 2022b). The Graph Convolutional Network (GCN) (Kipf and Welling, 2017) represents a significant advancement in this area by introducing localized spectral filters that operate within limited neighborhoods of each node. While spectral methods offer strong theoretical foundations rooted in signal processing theory, they encounter computational limitations due to the requirements of eigendecomposition.

Beyond spatial and spectral GNNs, diffusion-based models frame message passing as a continuous information flow process across the graph. These models lever-

age mathematical operators such as the graph Laplacian or transition matrices to simulate a diffusion process that captures how information naturally propagates through networked systems (Gasteiger et al., 2019; Chamberlain et al., 2021; Thorpe et al., 2022a; Zhao et al., 2021; Choi et al., 2023). Unlike conventional approaches limited to local neighborhood aggregation, the diffusion process enables multi-step propagation that captures long-range dependencies and global structural patterns (Xia et al., 2021; Dan et al., 2023). This broader perspective can improve performance on tasks where local connectivity alone fails to capture the underlying relationships between nodes.

1.3 Research Objectives and Contributions

The complex and irregular characteristics of graph-structured data create significant challenges in developing robust graph neural network models. This thesis addresses these challenges through three interconnected research objectives, each corresponding to a specific research question (RQ), as listed below.

RQ1: How can we enhance the expressive power of GNNs beyond traditional 1-WL limitations?

Standard message-passing GNNs are bounded by the expressive power of the 1-Weisfeiler-Lehman test, restricting their capacity to capture complex structural interactions and distinguish certain non-isomorphic graphs.

- **Objective 1**: Develop permutation-invariant graph partitioning methods to capture complex structural interactions that exceed the expressivity bounds of the 1-WL test.
- **Chapter Connection**: Addressed in Chapter 3. This work introduces Graph Partitioning Neural Networks (GPNNs) that utilize three distinct interaction types (intra-partition, inter-partition, and comprehensive) to achieve expressivity levels approaching k-WL while maintaining computational efficiency.

RQ2: How can information flow in GNNs be optimized based on geometric properties and neighborhood characteristics?

Existing message-passing schemes fail to leverage geometric characteristics of graph structure and lack mechanisms to adapt information propagation based on local curvature, topological properties, and varying homophily patterns in node neighborhoods.

- **Objective 2**: Create adaptive mechanisms that dynamically adjust information propagation for individual nodes based on their local geometric properties and neighborhood characteristics.
- **Chapter Connection**: Explored in Chapter 4 and Chapter 5. In Chapter 4, we develop the Bakry-Émery Curvature GNN (BEC-GNN) framework that learns

node-wise curvature measures to determine optimal aggregation depths, addressing over-smoothing and information flow optimization challenges. In Chapter 5, we introduce Adaptive Graph Neural Networks (AD-GNN) with theoretical frameworks, including signal preservation factors and classification quality metrics, to guide aggregation depth selection strategies in graphs with varying homophily patterns.

RQ3: How can GNN models effectively capture and optimize complex dynamic propagation processes in networks?

Traditional diffusion-based GNNs lack mathematically grounded frameworks for modeling heterogeneous propagation behaviors and network influence dynamics, failing to provide interpretable mechanisms that capture diverse temporal diffusion patterns and optimize complex propagation processes.

- **Objective 3**: Develop mathematically rigorous frameworks that model heterogeneous diffusion mechanisms and optimize network influence propagation.
- **Chapter Connection**: Addressed in Chapter 6 and Chapter 7. In Chapter 6, we present the Generalized Opinion Dynamics Diffusion Framework (GODNF) that incorporates multiple opinion dynamics models into a trainable neural diffusion mechanism with provable convergence properties. In Chapter 7, we introduce the Deep Sheaf Network (DeepSN) architecture that combines sheaf theory-based diffusion-reaction processes with subgraph-based optimization for influence maximization tasks.

These three research objectives form a cohesive progression that systematically advances graph neural network capabilities. This thesis is structured into seven chapters. Following this introduction, Chapter 2 surveys the relevant literature that forms the foundation for our contributions. Chapters 3 to 7 present the main research contributions, where each contribution targets specific challenges in graph representation learning. Chapter 8 concludes the thesis with a discussion on limitations and future work.

In the following, we present an overview of the contributions made in Chapters 3 to 7, one by one.

1.3.1 Permutation-Invariant Graph Partitioning: How Graph Neural Networks Capture Structural Interactions?

GNNs have established themselves as fundamental tools for graph-based learning tasks, with Message-Passing Neural Networks (MPNNs) representing the most widely adopted paradigm due to their computational efficiency and intuitive design principles (Gilmer et al., 2017; Kipf and Welling, 2017). However, the representational capabilities of MPNNs are fundamentally constrained by the 1-Weisfeiler-Lehman (1-WL) test (Weisfeiler and Leman, 1968), which limits their ability to distinguish

between certain non-isomorphic graph structures. This limitation has motivated extensive research into developing more expressive architectures that can surpass these theoretical bounds while maintaining practical applicability.

Traditional approaches to enhancing GNN expressivity have primarily focused on incorporating structural properties as preprocessing features (Bouritsas et al., 2022; Barceló et al., 2021; Wijesinghe and Wang, 2022; Xu et al., 2024), utilizing higher-order substructures (Morris et al., 2019, 2020b; Abu-El-Haija et al., 2019), expanding message-passing receptive fields (Nikolentzos et al., 2020; Feng et al., 2022; Wang et al., 2022b), or extracting subgraph-based information (Zhang and Li, 2021; Bevilacqua et al., 2022; Cotta et al., 2021; Wang and Zhang, 2022a). While these methods have achieved notable improvements, they often overlook a crucial aspect of graph analysis: the complex interactions between different structural components within a graph. Real-world networks typically consist of diverse structural elements that exhibit varying topological properties, and understanding how these components interact can provide valuable insights for representation learning.

This chapter addresses how to systematically capture structural interactions within graphs to enhance GNN expressivity through permutation-invariant graph partitioning. Unlike conventional approaches that treat graphs as monolithic structures, we decompose them into substructures with distinct topological characteristics. The key insight is that interactions within and across these partition boundaries contain valuable information for learning robust representations. Crucially, any partitioning scheme must preserve permutation invariance to ensure isomorphic graphs are treated identically regardless of node ordering. The main contributions of this chapter are:

- **Theoretical Framework**: We establish novel connections between graph partitioning and graph isomorphism by introducing the concepts of partition-isomorphism and interaction-isomorphism. These theoretical constructs provide a rigorous foundation for analyzing how structural interactions contribute to graph distinguishability.
- **Architectural Innovation**: We present Graph Partitioning Neural Networks (GPNNs), a novel architecture that integrates structural interactions through permutation-invariant graph partitioning. This approach enables the systematic exploration of intra-partition, inter-partition, and comprehensive interactions within graph structures.
- **Expressivity Analysis**: Through comprehensive theoretical analysis, we demonstrate that GPNNs can surpass the expressive power of 1-WL while efficiently approaching the capabilities of 3-WL. This represents a significant advancement in balancing computational efficiency with representational power.
- **Practical Implementation**: We employ computationally efficient partitioning schemes, including k-core and degree-based approaches, that enable meaningful subgraph extraction while preserving essential structural properties. These schemes provide practical pathways for implementing our theoretical framework.

1.3.2 Depth-Adaptive Graph Neural Networks via Learnable Bakry-Émery Curvature

Recent advances in geometric deep learning have highlighted the potential of incorporating curvature-based measures to enhance GNN capabilities (Ye et al., 2019; Topping et al., 2022; Nguyen et al., 2023; Fesser and Weber, 2024). Curvature (Bochner, 1946), originally derived from differential geometry, provides insights into the local geometric properties of spaces and has been successfully adapted to discrete graph structures (Forman et al., 1999; Ollivier, 2007, 2009). When applied to graphs, curvature captures connectivity patterns and relational properties that influence information flow, offering valuable guidance for addressing common GNN limitations such as over-smoothing and over-squashing (Qureshi et al., 2023).

However, existing curvature-based approaches in GNNs suffer from significant limitations. Most current methods treat curvature as a fixed, pre-computed property derived solely from graph topology, limiting their adaptability to specific learning contexts. These approaches typically employ curvature in decoupled techniques such as graph rewiring (Topping et al., 2022; Nguyen et al., 2023; Fesser and Weber, 2024) or sampling (Liu et al., 2023), which fail to dynamically interact with the evolving feature representations during training. Furthermore, traditional curvature measures focus primarily on structural properties while overlooking the critical interplay between node features, diffusion dynamics, and task-specific requirements.

This chapter addresses these limitations by introducing a novel framework that integrates learnable Bakry-Émery curvature into adaptive depth mechanisms for GNNs to improve information propagation. Unlike traditional curvature measures that rely solely on discrete graph structure, Bakry-Émery curvature (Ambrosio et al., 2015) captures both intrinsic structural properties and diffusion dynamics, providing a more comprehensive understanding of information propagation patterns within graphs. The main contributions of this chapter are:

- **Theoretical Foundations**: We establish a rigorous connection between Bakry-Émery curvature and feature distinctiveness in GNNs, demonstrating that nodes with higher curvature exhibit faster information diffusion and shorter mixing times, requiring fewer layers for effective representation learning. Conversely, nodes with lower curvature benefit from deeper aggregation to capture broader neighborhood information.
- **Computational Innovation**: We develop an efficient, learnable approximation strategy that selects task-specific function subsets for curvature estimation, making the approach scalable to large graphs. This strategy overcomes the computational intractability of evaluating curvature over entire function spaces while preserving the essential geometric properties.
- **Adaptive Architecture**: We introduce a depth-adaptive mechanism that uses estimated node curvature to dynamically determine optimal message-passing depths for each node. This approach enhances feature distinctiveness and captures finer structural variations while preventing over-smoothing and under-reaching issues.

- **Model Agnostic Framework**: Our approach serves as a general enhancement mechanism that can be integrated into existing GNN architectures, including GCN, GAT, GraphSAGE, and other message-passing frameworks, demonstrating broad applicability across different model types.

1.3.3 Beyond Fixed Depth: Adaptive Graph Neural Networks Under Varying Homophily

GNNs have emerged as powerful tools for node classification across diverse domains, demonstrating remarkable success in applications ranging from social network analysis to bioinformatics (Sun et al., 2024; Khoshraftar and An, 2024). The foundation of most GNN architectures rests on the homophily assumption, which considers that connected nodes tend to share similar labels or characteristics (McPherson et al., 2001; Zhu et al., 2020). While this assumption has proven effective for many real-world networks, including citation networks and knowledge graphs, it significantly constrains GNN performance when applied to heterophilic graphs, where connected nodes often exhibit different properties or belong to different classes (Zheng et al., 2023a). Traditional GNNs struggle in these heterophilic graphs because their message-passing mechanisms, designed under the homophily assumption, may inadvertently propagate conflicting signals that degrade classification performance rather than enhance it.

Recent research has begun addressing these limitations through specialized heterophily-aware architectures and alternative message-passing strategies (Zhu et al., 2021; Zheng et al., 2022; Luan et al., 2022; Zhu et al., 2023a; Wang et al., 2024a). However, existing approaches face two critical shortcomings. First, the conventional binary categorization of graphs as either homophilic or heterophilic oversimplifies the complex nature of real-world networks, which often exhibit varying levels of homophily across different regions and nodes. Second, most current methods apply uniform aggregation strategie, specifically, fixed layer depths across all nodes, overlooking the fact that different nodes may require different propagation strategies based on their local neighborhood characteristics and structural contexts.

This chapter addresses these fundamental limitations through a novel observation: optimal information propagation strategies are inherently node-specific rather than graph-global. The uniform nature of current GNN architectures, where fixed aggregation depths are applied universally across all nodes, limits their ability to adapt to the diverse information propagation needs that exist within individual graphs. The research contributions of this chapter are:

- **Theoretical Contributions:** We establish rigorous theoretical connections between local neighborhood characteristics and optimal message-passing strategies. Our analysis introduces signal preservation factors that quantify how neighborhood label composition affects class-discriminative information during aggregation. This framework extends to multi-layer analysis, demonstrating how effects compound across deeper networks, and enables the derivation of theoretically grounded depth benefit metrics for dynamic depth allocation.

- **Architectural Innovation:** Building on our theoretical insights, we propose Adaptive Depth Graph Neural Networks (AD-GNN), a novel architecture that dynamically adjusts aggregation depth for individual nodes based on their specific information propagation requirements. Our approach includes two variants to accommodate different computational requirements.
- **Unified Framework for Heterogeneity:** Unlike existing approaches that require separate architectures or preprocessing steps for homophilic versus heterophilic graphs, our framework provides a unified solution that seamlessly accommodates both patterns within a single architecture.

1.3.4 Graph Neural Diffusion via Generalized Opinion Dynamics

Diffusion-based GNNs have emerged as a promising alternative to traditional message-passing architectures, modeling information propagation through networks as continuous or discrete dynamical processes analogous to physical phenomena such as heat diffusion and random walks (Chamberlain et al., 2021; Gasteiger et al., 2019; Thorpe et al., 2022a). These approaches offer theoretical advantages over conventional GNNs by explicitly modeling the temporal evolution of node representations and enabling the capture of long-range dependencies. However, existing diffusion GNNs suffer from three critical limitations that constrain their effectiveness across diverse graph types and learning scenarios.

First, current diffusion GNNs rely on homogeneous diffusion mechanisms with static dynamics, assuming uniform information propagation across all nodes and fixed temporal evolution. This prevents adaptation to node-specific behaviors and dynamic patterns that characterize real-world networks. Second, computational complexity increases substantially with deeper architectures due to numerical integration requirements and unshared parameters, while interpretability diminishes as information flow becomes increasingly obscured. Third, theoretical understanding of convergence properties remains limited, with insufficient characterization of representational capacity and the conditions under which different diffusion behaviors emerge.

This chapter addresses these fundamental challenges through a novel observation: effective diffusion processes in complex networks require heterogeneous, adaptive mechanisms that can accommodate diverse propagation patterns within a single framework. Rather than assuming uniform diffusion dynamics, we propose that optimal information flow should reflect the inherent diversity of node behaviors and neighborhood characteristics present in real-world graphs. Our approach draws inspiration from opinion dynamics models, which provide well-established mathematical frameworks for understanding how information, beliefs, or influences propagate through social networks (DeGroot, 1974; Friedkin and Johnsen, 1990; Rainer and Krause, 2002). These models naturally incorporate heterogeneous agent behaviors through parameters such as stubbornness (i.e., resistance to external influence) and dynamic influence modeling between neighbors. Unlike traditional diffusion processes that assume uniform propagation, opinion dynamics models explicitly account

for individual agent characteristics and temporal evolution of influence relationships (Das et al., 2014; Hassani et al., 2022). Furthermore, they possess provable convergence properties with well-characterized equilibrium states, offering theoretical guarantees that are often absent in existing diffusion GNNs. The research contributions of this chapter are:

- **Unified Opinion Dynamics Framework:** We establish a principled connection between multiple opinion dynamics models and neural diffusion, creating a single framework that generalizes French-DeGroot, Friedkin-Johnsen, and Hegselmann-Krause formulations with rigorous convergence guarantees.
- **Novel Architecture:** We propose a novel diffusion GNN that incorporates node-specific stubbornness parameters, dynamic neighborhood influence, and structural regularization while maintaining computational efficiency comparable to traditional GNNs.
- **Theoretical Analysis:** We demonstrate that our framework can exhibit diverse convergence configurations: single consensus, multi-consensus, and individualized consensus, providing greater representational flexibility than existing approaches.

1.3.5 DeepSN: A Sheaf Neural Framework for Influence Maximization

Influence maximization represents a fundamental challenge in network science, seeking to identify optimal seed sets that maximize information spread across complex networks (Li et al., 2018). Traditional approaches to this problem rely on explicit specification of diffusion models and face significant computational challenges due to the exponential growth of the search space. Recent learning-based methods have attempted to address these limitations by leveraging GNNs to automatically learn diffusion patterns from data, offering improved generalizability across different propagation models (Kumar et al., 2022; Ling et al., 2023; Panagopoulos et al., 2023).

However, existing GNN-based approaches encounter two critical limitations that constrain their effectiveness in modeling influence propagation. First, traditional GNN architectures, while successful for static tasks such as node classification, struggle to capture the dynamic and heterogeneous nature of influence diffusion processes. Their inherent inductive biases assume uniform information propagation patterns and fail to model the complex temporal dynamics characteristic of real-world influence spread. Moreover, these architectures suffer from oversmoothing issues that hamper their ability to capture global information and long-range dependencies crucial for understanding influence propagation (Xia et al., 2021). Second, the combinatorial optimization challenge inherent in seed set selection remains computationally intractable, with existing approaches often relying on expensive reinforcement learning methods or ranking-based heuristics that lack interpretability and suffer from scalability issues.

This chapter addresses these fundamental limitations through a novel observation: influence propagation can be effectively modeled as a topological phenomenon that captures both the geometric properties of network structures and the dynamic

evolution of information flow. Rather than treating diffusion as a uniform process operating over static graph connectivity, we propose that optimal influence modeling requires explicit consideration of the topological relationships between nodes and the heterogeneous dynamics governing their interactions.

Our approach leverages sheaf theory, an algebraic-topological framework that provides rigorous mathematical foundations for understanding complex network phenomena through the lens of local-to-global consistency relationships (Tennison, 1975; Bredon, 2012). Unlike traditional diffusion models that assume homogeneous propagation, sheaf-based approaches can capture diverse influence patterns by modeling private opinions at nodes and public opinions along edges, with learnable transformation maps governing information flow between these representations. The research contributions of this chapter are:

- **Sheaf Reaction-Diffusion Framework:** We introduce a novel formulation of influence propagation as a sheaf diffusion-reaction process that captures both information spread and intrinsic node transformations, enabling effective modeling of complex propagation patterns in progressive and non-progressive diffusion scenarios.
- **Novel Architecture:** We develop a novel sheaf neural architecture that extends traditional diffusion mechanisms with pointwise and coupled dynamics operators, enabling the modeling of both information spread and intrinsic node state transitions while maintaining theoretical stability guarantees.
- **Subgraph-Based Optimization:** We propose a learning-based seed selection mechanism that leverages structural insights from sheaf coefficients to partition graphs into meaningful subgraphs, significantly reducing the combinatorial search space while accounting for overlapping influence between nodes.

1.4 Thesis Roadmap

Table 1.1 provides a concise map linking each chapter to its primary method, core theoretical results, and experimental support.

Table 1.1: Chapter-to-contribution map summarising key theoretical insights and experimental support for each proposed method.

Ch.	Method	Core Contribution	Theoretical Insights	Experiments
3	GPNN	Expressivity enhancement via permutation-invariant graph partitioning	Decomposing a graph into structurally distinct partitions and modelling interactions within and across boundaries strictly increases expressivity beyond 1-WL, while maintaining efficiency than 3-WL.	Graph classification, graph regression, and node classification
4	BEC-GNN	Depth-adaptive propagation via learnable Bakry-Émery curvature	Bakry-Émery curvature captures how rapidly information diffuses around each node. High-curvature nodes reside in dense, well-connected regions where shallow aggregation suffices. Low-curvature nodes require deeper propagation to access sufficient neighbourhood context. Learnable curvature allows the framework to adapt to task-specific feature distributions, enabling principled per-node depth allocation.	Node classification, graph classification, graph regression, and influence estimation
5	AD-GNN	Node-specific adaptive aggregation depth under varying homophily	The impact of aggregation on a node varies with its local label composition and neighborhood structure. In cases of strong homophily, deeper aggregation enhances class signals, while in strong heterophily with low degree, it leads to signal cancellation. These effects accumulate across layers, suggesting the need for node-specific depth decisions.	Node classification
6	GODNF	Unified neural opinion dynamics framework for graph diffusion	Modelling diffusion as an opinion dynamics process, where each node balances its own initial state against neighbourhood influence guarantees convergence and naturally produces diverse equilibrium behaviours depending on model parameters.	Node classification, and influence estimation
7	DeepSN	Influence maximisation via sheaf reaction-diffusion	Representing influence propagation as a sheaf diffusion-reaction process enriches the representational capacity of each edge beyond scalar weights, while preserving feature separability and avoiding oversmoothing. Learned sheaf coefficients expose community structure that reduces the seed selection complexity in search space.	Influence maximization, and influence estimation

1.5 Thesis as a Unified Research Program

The five contributions in this thesis target distinct challenges but follow a natural progression across three parts. Chapter 3 addresses expressivity, showing how structural decomposition via graph partitioning enables GNNs to capture interactions beyond the limits of standard message passing. Chapters 4 and 5 then ask how aggregation should proceed given that structure, arriving at complementary answers through geometric curvature and label-theoretic homophily reasoning respectively. Chapters 6 and 7 move beyond static representation to dynamic processes, modelling how information diffuses and spreads through networks over time. The three parts are connected: the adaptive propagation principles of Chapters 4 and 5 directly motivate the node-specific diffusion mechanisms in Chapters 6 and 7, and the structural decomposition of Chapter 3 informs the community-based optimisation in Chapter 7. The thesis thus builds cumulatively toward a more complete treatment of graph neural networks spanning expressivity, adaptive propagation, and dynamic network processes.

Chapter 2

This chapter provides the foundational concepts and theoretical frameworks essential for understanding the research presented in this thesis. We begin by establishing basic graph notation and defining the primary GNN architectures, including spatial and diffusion-based approaches. We then examine key theoretical aspects of GNNs, including their expressivity limitations, performance characteristics on different graph types, and fundamental challenges such as oversmoothing. Further, this chapter explores recent advances that leverage geometric insights, opinion dynamics, and adaptive mechanisms to enhance GNN capabilities. Finally, we review how GNNs have been applied to influence maximization, a dynamic optimization problem on graphs.

2.1 Basic Notations and Definitions

We begin by introducing the fundamental notations and definitions used in this chapter. Let $\{ \cdot \}$ denote a set and $\{\!\{ \cdot \}\!\}$) denote a multiset. The cardinality of a set or multiset is denoted by $| \cdot |$.

We represent an undirected graph as $G = (V, E, X)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of n nodes, $E \subseteq V \times V$ is the set of edges, and $X \in \mathbb{R}^{n \times d}$ is the node feature matrix with d-dimensional feature vectors. For each node $v_i \in V$, we denote its feature vector as $\mathbf{x}_i \in \mathbb{R}^d$, which corresponds to the i-th row of X. The neighborhood of a node v_i is defined as $\mathcal{N}(v_i) = \{v_j \in V | (v_j, v_i) \in E\}$ representing the set of nodes that have edges to v_i.

2.2 Graph Neural Networks

In this section, we discuss two main types of GNNs relevant to this thesis: *spatial GNNs* and *diffusion GNNs*.

2.2.1 Spatial GNNs

Spatial GNNs are defined on the spatial domain using neighborhood aggregation, where these GNNs aim to find an aggregation function that iteratively updates node

features using the node's feature from the previous time step as well as its neighbors' features. Spatial GNNs are also commonly known as Message-passing Neural Networks (MPNNs) because they follow a message-passing paradigm where nodes exchange information with their direct neighbors through explicit message functions. In this framework, each node collects messages from its neighboring nodes, aggregates these messages using permutation-invariant functions, and then updates its own representation by combining the aggregated neighborhood information with its current state. This iterative process allows information to propagate through the graph structure, enabling nodes to incorporate increasingly distant neighborhood information with each passing layer, thus capturing both local connectivity patterns and broader structural properties of the graph.

Sperduti and Starita (1997) first introduced the concept of MPNNs , where they introduced recursive neural networks that extend standard neural networks to process structured data like graphs and trees by using neighbourhood aggregation, where each node's representation is computed recursively from its neighbours' features. Building upon this early work, researchers like Gori et al. (2005), Scarselli et al. (2008), and Gallicchio and Micheli (2010) extended these approaches to handle more complex graph topologies, including directed, undirected, and cyclic graphs.

Let $\mathbf{h}_i^{(t)}$ denote the embedding of node v_i at step t, initialized as $\mathbf{h}_i^{(0)} = \mathbf{x}_i$. The MPNN framework consists of two phases:

Message Passing: For $t = 1, 2, \ldots, T$ steps:

$$\mathbf{m}_i^{(t)} = \text{AGG}\left(\{\{\mathbf{h}_j^{(t-1)} : v_j \in \mathcal{N}(v_i)\}\}\right) \tag{2.1}$$

$$\mathbf{h}_i^{(t)} = \text{UPD}\left(\mathbf{h}_i^{(t-1)}, \mathbf{m}_i^{(t)}\right) \tag{2.2}$$

where AGG is a permutation-invariant function that aggregates neighbor messages since the order of neighbors should not affect the result, and UPD is a differentiable update function to enable gradient-based learning (Maron et al., 2018).

Readout: A graph-level representation is computed as:

$$\mathbf{h}_G = \text{READOUT}\left(\{\mathbf{h}_i^{(T)} | v_i \in V\}\right) \tag{2.3}$$

The READOUT function must be both differentiable for end-to-end training and permutation-invariant since graphs have no canonical node ordering (Keriven and Peyré, 2019).

Several popular MPNN variants have emerged in recent years, each with specific design choices for the aggregation and update functions. Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017) use degree-normalized aggregation where AGG computes a weighted sum of neighbor features normalized by node degrees, while the UPD function applies a linear transformation followed by a nonlinear activation. GraphSAGE (Hamilton et al., 2017) employs various aggregation schemes, including mean, max, and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) based aggregators, with an update function that concatenates the node's

previous representation with the aggregated neighborhood information. Graph Attention Networks (GATs) (Veličković et al., 2018) introduce attention mechanisms where the aggregation function computes weighted combinations of neighbor features using learned attention coefficients, allowing the model to focus on more relevant neighbors. Graph Isomorphism Networks (GINs) (Xu et al., 2019) use sum aggregation and an update function that combines the node's previous state with aggregated messages through a multi-layer perceptron (MLP) (Kruse et al., 2022), providing theoretical guarantees for distinguishing graph structures.

2.2.2 Diffusion GNNs

Diffusion GNNs extend the message-passing paradigm by modeling information propagation as a diffusion process across the graph structure. Unlike traditional spatial GNNs that rely on local neighborhood aggregation, diffusion GNNs incorporate global graph structure through mathematical operators such as the graph Laplacian or transition matrices (Gasteiger et al., 2019). This approach enables information to flow beyond immediate neighborhoods, allowing nodes to capture long-range dependencies and global structural patterns within the graph.

The core principle of diffusion GNNs lies in modeling the temporal evolution of node features through a diffusion process (Han et al., 2024). Let $\mathbf{H}(t) \in \mathbb{R}^{n \times d}$ denote the node feature matrix at time t, where each row corresponds to the d-dimensional feature vector of a node. The diffusion process is governed by a temporal operator $\mathcal{D}$ that describes how node features evolve:

$$\mathcal{D}[\mathbf{H}(t)] = \Phi(\mathbf{H}(0), \mathbf{H}(t), G; \Theta) \tag{2.4}$$

where $\mathbf{H}(0) = X$ represents the initial node features, Φ is a learnable diffusion operator parameterized by Θ, and G encodes the graph structure. The diffusion operator Φ must be differentiable to enable gradient-based learning and should respect the graph's structural properties. Diffusion GNNs can be categorised into two main classes: discrete-time and continuous-time approaches.

Discrete-Time Diffusion. Discrete diffusion GNNs propagate information through iterative steps with explicit layer updates. In discrete-time formulations, the temporal operator takes the form $\mathcal{D}[\mathbf{H}(t)] = \mathbf{H}(t+1) - \mathbf{H}(t)$, yielding the iterative update rule:

$$\mathbf{H}(t+1) = \mathbf{H}(t) + \Phi(\mathbf{H}(0), \mathbf{H}(t), G; \Theta) \tag{2.5}$$

This discrete formulation is computationally efficient and straightforward to implement, making it suitable for practical applications. Representative architectures include APPNP (Gasteiger et al., 2018), GDC (Gasteiger et al., 2019), ADGN (Zhao et al., 2021), and GPRGNN (Chien et al., 2021).

Continuous-Time Diffusion. In contrast, continuous models frame diffusion as dynamical systems governed by differential equations. The evolution is described

through an ordinary differential equation:

$$\frac{d\mathbf{H}(t)}{dt} = \Phi(\mathbf{H}(0), \mathbf{H}(t), G; \Theta) \tag{2.6}$$

The adaptive nature of continuous diffusion modeling allows these GNNs to capture intricate diffusion patterns more effectively than their discrete counterparts. Early continuous GNNs, such as GRAND (Chamberlain et al., 2021), closely mimic heat diffusion processes. Recent continuous diffusion GNNs have advanced beyond this foundation by employing techniques such as additional damping and source terms, external forces, and geometric tools to enhance their modeling capabilities. Notable examples include GRAND++ (Thorpe et al., 2022b), ACMP-GNN (Wang et al., 2022a), HiD-Net (Li et al., 2024b) and PDE-GCN (Eliasof et al., 2021).

2.3 Graph-based Downstream Tasks

In this section, we define the primary downstream tasks that are relevant to this thesis and serve as evaluation benchmarks for GNNs.

2.3.1 Node Classification

Given a graph $G = (V, E, X)$ with partially labeled nodes, the objective is to predict the labels of unlabeled nodes. Let $V_L \subset V$ denote the set of labeled nodes and $V_U = V \setminus V_L$ denote the set of unlabeled nodes. Each labeled node $v_i \in V_L$ has an associated ground truth label $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ is the label space. The goal is to learn a function $f : \mathbb{R}^d \to \mathcal{Y}$ that can accurately predict labels for nodes in V_U based on their feature representations and the graph structure.

2.3.2 Graph Classification

In this task, we are given a dataset of graphs $\mathcal{D} = \{(G_1, y_1), (G_2, y_2), \ldots, (G_m, y_m)\}$, where each graph $G_i = (V_i, E_i, X_i)$ has an associated label $y_i \in \mathcal{Y}$. The objective is to learn a function $f : \mathcal{G} \to \mathcal{Y}$ that maps graphs to their corresponding labels, where $\mathcal{G}$ represents the space of all possible graphs. This typically involves learning representations that capture the essential structural and feature information of the entire graph.

2.3.3 Graph Regression

Graph regression involves predicting continuous-valued targets for entire graphs. Consider a dataset $\mathcal{D} = \{(G_1, y_1), (G_2, y_2), \ldots, (G_m, y_m)\}$, where each $y_i \in \mathbb{R}$ (or $\mathbb{R}^d$). The objective is to learn a function $f : \mathcal{G} \to \mathbb{R}$ (or $\mathbb{R}^d$) that accurately predicts real-valued properties of these graphs.

2.3.4 Influence Maximization

Influence maximization is a significant problem in network science and graph optimization. It aims to identify a subset of nodes, referred to as the *seed set* (i.e., the initially activated nodes), within a graph that maximizes the spread of influence according to a specified diffusion function (Li et al., 2018; Ling et al., 2023). Given a graph $G = (V, E)$, the objective of the influence maximization problem is to find a subset $S^\star \subseteq V$ containing up to k nodes that maximizes the expected influence diffusion function σ.More precisely,

$$S^\star = \underset{|S| \leq k}{\operatorname{argmax}}\, \sigma(S, G; \theta).$$

Here, $S^\star$ is referred to as the optimal seed set and $\sigma(S, G; \theta)$ represents the expected influence diffusion of the seed set S in the graph G (i.e, the expected final number of activations) with model parameters θ.

The influence diffusion function $\sigma(\cdot; \theta)$ can be instantiated with various models to capture different dynamics of influence spread. In the Linear Threshold (LT) model (Granovetter, 1978), θ denotes node thresholds and edge weights, with nodes becoming active when the weighted sum of neighbors' influences exceeds their thresholds. In the Independent Cascade (IC) model (Goldenberg et al., 2001b), θ represents edge activation probabilities, where active nodes have a single chance to activate each neighbor. The Susceptible-Infected-Susceptible (SIS) model (d'Onofrio, 2008) extends this by incorporating infection and recovery rates in θ, allowing nodes to transition between susceptible and infected states over time.

2.3.5 Influence Estimation

This is a node-level regression task that involves predicting the influence probability of individual nodes within a network. Given a graph $G = (V, E)$ and a node $v_i \in V$, the objective is to learn a function $f : \mathbb{R}^d \to [0, 1]$ that estimates the probability p_i that node v_i will be influenced or activated by seed nodes in the network under a given diffusion model (such as the IC, LT, or SIS model). Unlike influence maximization problems that focus on selecting optimal seed sets, this task treats influence estimation as a regression problem where each node receives a continuous influence score representing its potential impact from information propagation throughout the graph.

2.4 Expressivity of GNNs

2.4.1 MPNNs and Weisfeiler-Lehman (1-WL) Test

The Weisfeiler-Lehman (1-WL) algorithm (Weisfeiler and Leman, 1968), also known as the color refinement algorithm, is a fundamental method for testing graph isomorphism by determining whether two graphs have identical structural properties (i.e.,

isomorphic). The algorithm operates by initially assigning each node same color, then iteratively refining these colors where each node's new color depends on its previous color and the multiset of colors in its neighborhood. This process continues until the color assignments stabilize, at which point the resulting color histograms can be compared between graphs where different histograms indicate non-isomorphic graphs.

The 1-WL algorithm can be formalized as follows:

$$h_v^{(0)} = c_0 \quad \text{for all } v \in V \tag{2.7}$$

$$h_v^{(k)} = \text{hash}(h_v^{(k-1)}, \{\{h_u^{(k-1)} | u \in \mathcal{N}(v)\}\}) \tag{2.8}$$

where c_0 is a constant initial color for all nodes. Further, $\text{hash}(\cdot)$ function is an injective mapping that assigns unique identifiers to distinct inputs, ensuring that different multisets receive different hash values. The process continues until convergence, i.e., when $h_v^{(k)} = h_v^{(k-1)}$ for all nodes $v \in V$. Despite its effectiveness and widespread use in graph analysis, the 1-WL algorithm has inherent limitations and cannot distinguish all pairs of non-isomorphic graphs, particularly those with certain regular or symmetric structures (Wijesinghe and Wang, 2022; Zhao et al., 2022a).

Xu et al. (2019) established a fundamental theoretical connection between MPNNs and the 1-WL graph isomorphism test by proving that GNNs are at most as powerful as the 1-WL test in distinguishing different graph structures. They showed that GNNs can achieve the same discriminative power as 1-WL if their neighborhood aggregation and graph readout functions are injective, leading them to propose Graph Isomorphism Networks (GIN) which provably matches 1-WL's expressive power. This theoretical framework is crucial for understanding the fundamental limitations and capabilities of GNNs, providing rigorous bounds on what graph structures these models can and cannot distinguish. In graph classification tasks like molecular property prediction, recognizing that certain GNN architectures struggle to differentiate specific molecular structures can clarify model failures (Bodnar et al., 2021; Sun et al., 2022).

2.4.2 MPNNs Beyond 1-WL Test

In recent years, researchers have increasingly focused on developing GNNs that can surpass the expressive power of 1-WL tests, addressing the fundamental limitations of standard MPNNs in distinguishing certain graph structures. Several distinct approaches have emerged to tackle this challenge, each offering unique strategies to enhance expressivity beyond the 1-WL barrier.

Structure-injecting GNNs extract structural information using a pre-processing step and inject it into a message-passing scheme as node or edge features (Horn et al., 2022; Barceló et al., 2021; Bouritsas et al., 2022; Wijesinghe and Wang, 2022). These GNNs have been empirically verified to perform well on graphs that exhibit specific structural patterns but often require hand-picking structure patterns, which limits the generalizability of the learning model. Furthermore, the substructure extraction

process can be computationally expensive, making these approaches less practical for large-scale applications.

Subgraph-based GNNs are designed to operate on subgraphs that are extracted from the original graph (Bevilacqua et al., 2022; Cotta et al., 2021; Zeng et al., 2023; Frasca et al., 2022). These approaches typically generate subgraphs according to combinatorial policies, such as extracting all possible ego networks or removing specific edges. To ensure permutation invariance of graph representations, these models often require computationally demanding solutions, such as equivariant architectures. While stochastic implementations through random sampling can improve efficiency, they cannot guarantee permutation invariance, creating a trade-off between computational feasibility and theoretical guarantees.

The k-WL algorithm (Kiefer, 2020; Cai et al., 1992) inspired the development of high-order GNNs that can capture higher-order relations by characterizing k-tuples of nodes in graphs (Maron et al., 2019b; Morris et al., 2020b; Zhao et al., 2022b). These models are built to mimic the k-WL hierarchy where k-order GNNs have expressive power comparable to the k-WL test. Despite being theoretically powerful, these models suffer from prohibitive computational complexity that scales exponentially with k, making them impractical for real-world applications with large graphs.

Another line of work focuses on enhancing MPNN expressivity through diverse architectural innovations. Expanding receptive field methods extend the traditional 1-hop message passing paradigm by allowing nodes to aggregate information from k-hop neighborhoods (Nikolentzos et al., 2020; Feng et al., 2022; Wang et al., 2022b), enabling the capture of longer-range dependencies and structural patterns without requiring multiple message passing iterations. Inductive coloring techniques assign unique identifiers or structural encodings to nodes based on their positions and roles within the graph (You et al., 2021; Huang et al., 2023), breaking symmetries that would otherwise make structurally different nodes appear identical to standard MPNNs. Homomorphism counting approaches enhance expressivity by incorporating counts of structure-preserving mappings from subgraphs to the target graph, providing rich structural information about motif occurrences and global patterns that cannot be captured by local message passing alone (Zhang et al., 2024b; Jin et al., 2024). While these methods offer complementary strengths in addressing different aspects of the 1-WL limitation, they often require careful design choices regarding which structural properties to encode, how to efficiently compute extended neighborhoods or homomorphism counts, and how to balance expressivity gains with computational overhead.

2.5 Homophilic and Heterophic Graphs

Graphs exhibit different patterns in how nodes connect based on their similarity. Homophilic graphs follow the "birds of a feather flock together" principle, where nodes with similar features, labels, or attributes preferentially connect to each other (Khanam et al., 2023). Social networks, citation networks, and collaboration graphs

often exhibit such homophilic characteristics (Bisgin et al., 2012; Ciotti et al., 2016; Zhang et al., 2018). Conversely, heterophilic graphs exhibit the opposite behavior, where nodes tend to connect to others that are different from themselves (Zhu et al., 2023a). This pattern is observed in various networks, including biological networks, financial transaction systems, and web graphs (Kimura and Hayakawa, 2008; Pei et al., 2020b; Lin et al., 2025).

For undirected graphs, several homophily measures have been proposed. For each node $v_i \in V$, let y_i denote its corresponding label. Edge homophily (Zhu et al., 2020) measures the fraction of edges connecting nodes with identical labels:

$$h_{edge} = \frac{|\{\{v_i, v_j\} \in E | y_i = y_j\}|}{|E|} \tag{2.9}$$

Node homophily (Pei et al., 2020b) adopts a node-centric approach, computing the average homophilic tendency across all nodes:

$$h_{node} = \frac{1}{|V|} \sum_{v_i \in V} \frac{|\{v_j \in \mathcal{N}(v_i) | y_i = y_j\}|}{|\mathcal{N}(v_i)|} \tag{2.10}$$

Traditional GNNs rely heavily on the fundamental assumption that neighboring nodes share similar characteristics and labels, making them inherently suited for homophilic graphs (Zhu et al., 2021; Luan et al., 2022). This design philosophy comes from their message-passing frameworks, where nodes aggregate and combine information from their immediate neighbors under the expectation that such local information sharing will enhance representation learning. However, this very strength becomes a critical weakness when applied to heterophilic graphs, where connected nodes often possess dissimilar attributes and belong to different classes. In such networks, the standard GNN approach of averaging or pooling neighboring node features can actually degrade performance by mixing contradictory signals and diluting meaningful distinctions between node classes. This severely impacts node classification tasks, where the goal is to predict node labels based on both structural and feature information (Sun et al., 2024). This performance degradation in node classification highlights the need for specialized approaches that can effectively handle graphs with varying homophily levels (Lim et al., 2021).

Geom-GCN (Pei et al., 2020b) is one of the early works that identified the performance bottleneck in heterophilic graphs. Since then, there has been a plethora of works targeting enhancing GNN performance in heterophilic graphs. Some works propose filtering neighbours to improve aggregation performance on heterophilic graphs by excluding or deprioritising dissimilar nodes (Bo et al., 2021; Luan et al., 2022; Zheng et al., 2023b; Guo et al., 2024). Furthermore, some works have shown that incorporating higher-order neighbourhoods could alleviate this issue, as it enables the aggregation process to access more homophilic information (Zhu et al., 2020; Wang et al., 2021b; Wang and Zhang, 2022b; Li et al., 2022c; Yu et al., 2024; Li et al., 2024a). There have been some works introducing structural constraints such as neighbour ordering (Song et al., 2023) and selective aggregation mechanisms (He

et al., 2022; Haruta et al., 2023) to enable more discriminative node representations by controlling how information flows between nodes of different classes. While the aforementioned works have focused on improving GNN architecture, some studies have taken an alternative path by rewiring the input graph to increase homophily (Guo et al., 2023; Li et al., 2023b; Bi et al., 2024; Bose et al., 2025).

Recent theoretical findings have shown that heterophily does not always negatively impact node classification. Ma et al. (2022b) showed that moderate levels of heterophily are more detrimental to GNNs than conditions of extreme heterophily. Further, Wang et al. (2024a) elaborated that class separability in node classification using GNNs depends on both neighborhood distribution distances and node degrees, establishing that the impact of heterophily is nuanced rather than uniformly negative.

2.6 Oversmoothing in GNNs

Traditional GNNs face a fundamental challenge known as oversmoothing, where repeated message passing operations cause node representations to converge toward uniform values, effectively erasing the distinguishing characteristics that enable meaningful node differentiation (Keriven, 2022; Qureshi et al., 2023). This phenomenon becomes particularly pronounced in deeper networks, where extensive information propagation leads to a homogenization of node embeddings across the entire graph, severely hampering the model's ability to perform downstream tasks such as node classification, and influence estimation (Xia et al., 2021; Jin and Zhu, 2025).

Mathematically, oversmoothing occurs when node representations converge as the network depth increases:

$$\lim_{t\to\infty} \|\mathbf{h}_i^{(t)} - \mathbf{h}_j^{(t)}\|_2 \to 0, \quad \forall i, j \in V \tag{2.11}$$

This convergence reduces the discriminative power of node embeddings, making initially distinct nodes indistinguishable in the embedding space.

The research community has developed diverse strategies to combat this issue. Architectural innovations have emerged as a primary defense mechanism, with residual connections (Yin et al., 2022; Yang et al., 2022b; Scholkemper et al., 2025) allowing networks to preserve original node features alongside transformed representations, while jumping knowledge mechanisms (Yang et al., 2022a; Zhu et al., 2023b) enable selective access to representations from multiple network layers. These approaches fundamentally alter how information flows through the network architecture. Complementary to these architectural modifications, regularization and normalization techniques have also proven effective in maintaining representational diversity. Dropout strategies (Luo et al., 2021; Shu et al., 2022; Hssayni et al., 2025) introduce controlled randomness during training to prevent feature convergence, while specialized normalization methods (Zhao and Akoglu, 2020; Zhou et al., 2020b) help

stabilize the learning process and preserve meaningful variation in node embeddings.

Recent advances have taken a more principled approach by leveraging geometric insights from discrete Ricci curvature to identify structural bottlenecks that cause oversmoothing (Nguyen et al., 2023; Liu et al., 2023; Fesser and Weber, 2024). These curvature-based methods not only diagnose problematic graph regions but also inform targeted interventions through graph rewiring and strategic sampling techniques that structurally address the root causes of feature oversmoothing, representing a shift toward theoretically grounded solutions for this fundamental challenge.

2.7 Curvature and GNNs

Curvature offers a geometric perspective on graph structure that can improve GNN design and performance. In this section, we review key curvature measures and their applications in GNNs.

2.7.1 Curvature Notions

Originally rooted in differential geometry, curvature measures how a space deviates from flatness by tracking the convergence or divergence of geodesics (Bochner, 1946; Ollivier, 2007). When applied to graphs, it captures the connectivity and relational properties of nodes and edges (Lin et al., 2011; Ni et al., 2015), offering a more nuanced structural analysis. Incorporating curvature into GNNs can not only deepen our understanding of local connectivity but also provide insights into information flow, helping mitigate issues like oversmoothing and oversquashing (Sun et al., 2023b; Fesser and Weber, 2024). Among the various forms of curvature, Ricci curvature (Huisken, 1985; Laugwitz, 2014) has gained prominence due to its formulation on discrete spaces, which allows for efficient computation. This efficiency makes it well-suited for analyzing the structural properties of graphs (Forman et al., 1999; Ollivier, 2007; Samal et al., 2018; Fesser and Weber, 2024). In the following, we present two notions of Ricci curvature adapted to the discrete graph setting.

2.7.1.1 Ollivier-Ricci Curvature

The Ollivier-Ricci curvature (Ollivier, 2009; Fesser et al., 2024) employs principles from optimal transport theory to quantify edge curvature. This measure evaluates the geometric properties of an edge by examining the transportation cost between probability distributions centered at the neighborhoods of connected nodes. For an edge $(v_i, v_j) \in E$, the Ollivier-Ricci curvature is expressed as:

$$\kappa_{OR}(v_i, v_j) = 1 - \frac{W_1(\mu_{v_i}, \mu_{v_j})}{\delta(v_i, v_j)} \tag{2.12}$$

where $W_1(\mu_{v_i}, \mu_{v_j})$ represents the Wasserstein distance between probability measures μ_{v_i} and μ_{v_j} supported on the neighborhoods $\mathcal{N}(v_i)$ and $\mathcal{N}(v_j)$ respectively, and

$\delta(v_i, v_j)$ denotes the shortest path distance between nodes (for edges in the graph, $\delta(v_i, v_j) = 1$). Positive curvature values indicate regions of high local connectivity, while negative values signal structural bottlenecks.

2.7.1.2 Forman-Ricci Curvature

Forman-Ricci curvature (Forman, 2003; Sreejith et al., 2016; Fesser et al., 2024) adopts a purely combinatorial framework, deriving curvature measurements directly from local topological structure. For an edge $e = (v_i, v_j) \in E$, the formulation is:

$$\kappa_{FR}(v_i, v_j) = 4 - |\mathcal{N}(v_i)| - |\mathcal{N}(v_j)| \tag{2.13}$$

where $|\mathcal{N}(v_i)|$ and $|\mathcal{N}(v_j)|$ represent the degrees of nodes v_i and v_j respectively. This topological approach characterizes how individual edges influence the local clustering properties of their surrounding network regions.

2.7.2 Curvature-based GNNs

Recent work in GNNs has leveraged various notions of curvature, most notably discrete Ricci curvature, to capture structural relationships in graphs. For instance, Ye et al. (2019) employed discrete Ricci curvature to quantify the relationship between the neighborhoods of node pairs, using these values to define edge weights and enhance message passing. Subsequent studies (Topping et al., 2022; Nguyen et al., 2023; Liu et al., 2023; Fesser and Weber, 2024) have further explored how curvature influences common GNN challenges, including oversmoothing and oversquashing. These investigations revealed that positive curvature regions often correlate with oversmoothing tendencies, where node representations become increasingly similar, while negative curvature areas are associated with oversquashing phenomena that impede long-range information propagation (Nguyen et al., 2023).

Building on these insights, Topping et al. (2022); Nguyen et al. (2023), and Fesser and Weber (2024) introduced graph rewiring strategies to improve information flow, while Liu et al. (2023) proposed an edge sampling mechanism that leverages curvature to alleviate these issues. All these approaches typically treat curvature as a static topological property computed prior to GNN training. More recently, Chen et al. (2025) proposed a curvature-based diffusion mechanism that iteratively updates edge curvature, effectively treating curvature as a learnable parameter.

2.8 Opinion Dynamics and GNNs

GNN message passing can be viewed as a form of opinion propagation, where nodes update their representations based on neighborhood information. This connection to opinion dynamics offers insights into GNN behavior and inspires new architectural designs.

2.8.1 Opinion Dynamic models

Opinion dynamics represents a fundamental area of study in social computing that examines how individual beliefs, preferences, and viewpoints evolve through social interactions within networked systems (Acemoglu and Ozdaglar, 2011; Das et al., 2014). This field has gained significant attention as researchers seek to understand phenomena ranging from political polarization and social consensus formation to the spread of misinformation and collective decision-making processes (Mueller and Tan, 2018). The integration of GNNs with opinion dynamics modelling offers promising avenues for capturing the complex, non-linear relationships that govern how information propagates and transforms across complex networks. Several classical models have emerged as cornerstone approaches for understanding these dynamics, each capturing different aspects of how individuals update their beliefs in response to social influence.

In the following, we discuss several well-known opinion dynamics models.

2.8.1.1 French-DeGroot (FD) Model

In the FD model (DeGroot, 1974), each node updates its opinion as a weighted average of its neighbors' opinions:

$$\mathbf{x}_i(t+1) = \sum_{v_j \in V} W_{ij}\mathbf{x}_j(t) \tag{2.14}$$

where $\mathbf{x}_i(t)$ denotes the opinion of node v_i at time t. The weight matrix $W \in \mathbb{R}^{n\times n}$ is row-stochastic, with $W_{ij} > 0$ if node v_i is influenced by node v_j. When the graph is connected, the opinions of all nodes converge to the same value (i.e., single consensus) (DeGroot, 1974).

2.8.1.2 Friedkin-Johnsen (FJ) Model

The FJ model (Friedkin and Johnsen, 1990) extends FD by allowing each node to retain partial attachment to its initial opinion:

$$\mathbf{x}_i(t+1) = \lambda_i\mathbf{x}_i(0) + (1-\lambda_i)\sum_{v_j \in V} W_{ij}\mathbf{x}_j(t) \tag{2.15}$$

where $\mathbf{x}_i(0)$ is the initial opinion of node v_i, and $\lambda_i \in [0,1]$ representing node v_i's level of attachment to its initial opinion. This opinion dynamics converges to an equilibrium state that reflects a balance between the influence of neighboring nodes and each node's adherence to its initial opinion, resulting in persistent diversity of opinions across the network (Biondi et al., 2023; Xu et al., 2022). Compared to the FD model, the FJ model better captures realistic scenarios where individuals maintain some degree of independence while still being susceptible to social influence.

2.8.1.3 Hegselmann–Krause (HK) Model

In the HK model (Rainer and Krause, 2002), each node updates its opinion by averaging those of other nodes whose opinions lie within a specified confidence interval:

$$\mathbf{x}_i(t+1) = \frac{1}{|\Gamma(v_i,t)|} \sum_{v_j \in \Gamma(v_i,t)} \mathbf{x}_j(t) \tag{2.16}$$

$$\Gamma(v_i,t) = \{v_j \in \mathcal{N}(v_i) \mid |\mathbf{x}_i(t) - \mathbf{x}_j(t)| \leq \epsilon\} \tag{2.17}$$

where $\epsilon > 0$ is the confidence bound. Unlike the FD and FJ models, the HK model depicts time-varying neighborhood influence as each node's influential neighbors change dynamically based on opinion proximity at each time step. This opinion dynamics typically leads to multiple opinion clusters instead of converging to a single consensus (Chen et al., 2017).

2.8.2 Opinion Dynamic-based GNNs

Several GNNs from the literature have drawn inspiration from opinion dynamics models to enhance their architectural designs. GNNs with residual connections (Gasteiger et al., 2018; Xu et al., 2018b; Liu et al., 2021a; Zhang et al., 2023) exhibit similarities to the FJ model, where the residual connections to the initial feature states mimic the concept of stubbornness in the model. Sheaf theory-based GNNs (Bodnar et al., 2022) adopt an opinion dynamics-inspired approach, where each node has both public and private opinions, and information propagation is modeled as the interplay between these two types of opinions. Zhou et al. (2024), and Vargas-Pérez et al. (2024) introduced message passing mechanisms based on the bounded confidence opinion model (Rainer and Krause, 2002), designed to enhance information propagation in GNNs. Wang et al. (2025) proposed a message-passing approach inspired by behavior opinion dynamics (Leonard et al., 2024) to address the oversmoothing issue in GNNs.

2.9 Adaptive GNNs

GNNs with adaptive architectures learn key components of their architecture during training, allowing them to optimize both structure and operations based on the graph's properties and the downstream task. Unlike conventional GNNs with fixed designs, these models incorporate learnable components that dynamically adjust aggregation strategies, assign importance to nodes/edges, and refine connectivity patterns.

Numerous studies have explored adaptive behaviors of GNNs in the literature. For example, sampling-based methods selectively extract graph structures to improve computational efficiency and representation capabilities (Younesian et al., 2023; Zhao et al., 2023; Lee et al., 2023). Similarly, several studies focus on defining adaptive neighborhoods for aggregation at each node by utilizing learnable mechanisms that

are governed by the ground truth values of the downstream task (Saha et al., 2023; Guang et al., 2024). Attention-based GNNs (Veličković et al., 2018; Brody et al., 2022) adjust edge weights adaptively by learning attention coefficients that reflect the relative importance of neighboring node during the learning process. Additionally, several studies have proposed dynamic message-passing paradigms that can adaptively determine node message-passing operations, such as message filtering and the direction of information propagation (Errica et al., 2023; Finkelshtein et al., 2024).

There have been few works exploring the impact of adaptive depth in GNNs where each node has a personalised number of message-passing iterations. Wu et al. (2024) employed reinforcement learning to design a flexible GNN architecture that adaptively searches for optimal parameters of components, including GNN depth, aggregation functions, and pooling operations for graph classification. ADMP-GNN (Abbahaddou et al., 2025) introduced a depth allocation policy for GNNs using centrality heuristics to cluster structurally similar nodes and assign optimal layer depth based on their validation performance.

2.10 GNNs for Influence Maximization

Influence maximization methods generally fall into two main categories: traditional and learning-based approaches. Traditional methods include simulation-based, proxy-based, and sketch-based techniques (Li et al., 2018). Simulation-based methods rely on Monte Carlo simulations (Harrison, 2010) for stochastic evaluation with theoretical guarantees but suffer from computational inefficiency. Proxy-based methods use heuristic approaches for efficient seed set approximation but often lack theoretical foundations. Sketch-based methods attempt to combine both simulation and proxy approaches, seeking to balance theoretical guarantees with computational efficiency (Banerjee et al., 2020).

In contrast to traditional methods that require a specific diffusion model as input, learning-based models demonstrate superior generalizability by adapting to multiple diffusion models without requiring model-specific parameter tuning. The learning-based paradigm has attracted significant research attention, with several studies leveraging deep reinforcement learning for influence maximization (Li et al., 2022b; Ma et al., 2022a; Wang et al., 2021a; Chen et al., 2023b). These reinforcement learning approaches model influence maximization as a sequential decision-making problem, where agents learn to select seed nodes through trial-and-error interactions with the network environment.

Complementary to reinforcement learning methods, several works have explored the application of GNNs in the influence maximization context (Xia et al., 2021; Kumar et al., 2022; Ling et al., 2023; Panagopoulos et al., 2023). Xia et al. (2021) proposed DeepIS, which uses GNNs to estimate node susceptibility in social networks, treating influence estimation as a node-level regression task. Kumar et al. (2022) combined graph embeddings with GNNs to learn node representations that capture influence propagation patterns, enabling more effective seed selection strategies.

Ling et al. (2023) developed a deep graph representation learning framework that jointly optimizes node embeddings and influence maximization objectives through end-to-end training. More recently, Panagopoulos et al. (2023) introduced GLIE, a GNN that learns to estimate influence spread by tightening theoretical upper bounds through supervised training, and developed corresponding influence maximization algorithms that uses learned representations for submodular seed selection. These GNN-based approaches leverage the inherent structural properties of graphs to learn effective node representations, offering promising directions for scalable influence maximization in large networks.

Chapter 3

Permutation-Invariant Graph Partitioning: How Graph Neural Networks Capture Structural Interactions?

3.1 Overview

GNNs have emerged as the *de facto* standard for tackling graph learning tasks (Horn et al., 2022; Bodnar et al., 2021; Bouritsas et al., 2022). Among the rich landscape of GNN models, Message-Passing Neural Networks (MPNNs) stand out for their simplicity and efficiency, making them a popular choice for real-world applications. This is due to their ability to leverage local neighborhood information through a message-passing scheme, which iteratively computes node representations (Gilmer et al., 2017; Kipf and Welling, 2017).

However, the representational power of MPNNs is fundamentally constrained by the 1-WL test (Weisfeiler and Leman, 1968; Xu et al., 2019; Morris et al., 2019). To overcome this limitation, researchers have proposed various approaches to enhance MPNN expressivity beyond 1-WL, including injecting structural properties (Bouritsas et al., 2022; Barceló et al., 2021; Wijesinghe and Wang, 2022; Xu et al., 2024), incorporating higher-order substructures (Morris et al., 2019, 2020b; Abu-El-Haija et al., 2019), expanding receptive fields for message passing (Nikolentzos et al., 2020; Feng et al., 2022; Wang et al., 2022b), leveraging subgraphs (Zhao et al., 2022a; Zhang and Li, 2021; Wang and Zhang, 2022a; Bevilacqua et al., 2022; Cotta et al., 2021), integrating homomorphism counts (Zhang et al., 2024b; Jin et al., 2024) and applying inductive coloring techniques (You et al., 2021; Huang et al., 2023). Despite these advancements, there is still limited understanding of how structural components within a graph, such as subgraphs representing diverse properties, interact and influence expressivity in learning tasks. Existing methods primarily focus on straightforward interactions between nodes and their neighbors, leaving the intricate interplay among structural components underexplored. Addressing these gaps is crucial for unlocking the full potential of expressive GNNs.

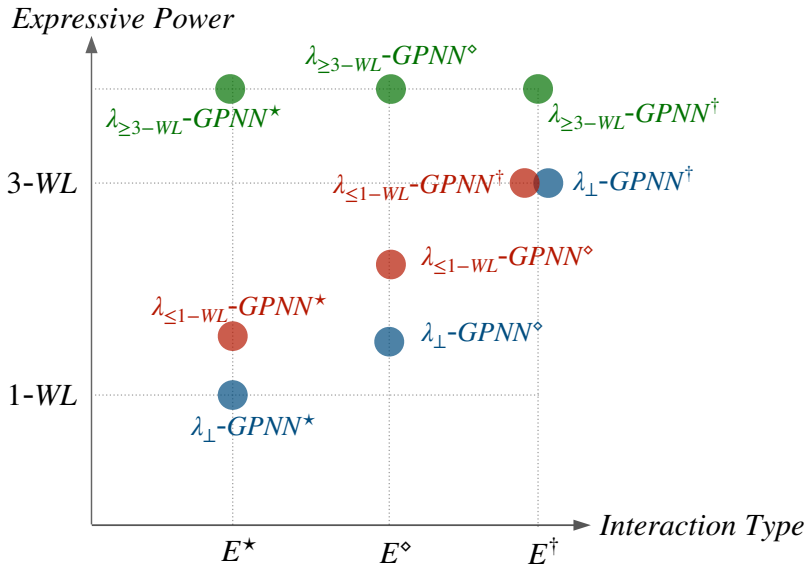


Figure 3.1: The expressivity of GPNN variants is analysed in terms of (1) *interaction type*: $E^{\star}$ (intra-edges), $E^{\diamond}$ (intra-edges and inter-edges), and $E^{\dagger}$ (all edges); (2) *expressive power*: 1-WL and 3-WL, under different permutation-invariant graph partitioning schemes $\lambda_{\perp}$, $\lambda_{\leq 1\text{-}WL}$, and $\lambda_{\geq 3\text{-}WL}$.

In this chapter, we aim to address the aforementioned limitations. Our key observations are that partitioning a graph into a set of subgraphs that preserve structural properties enables a powerful way to exploit interactions among different structural components of a graph. Particularly, graphs in real-world applications are often composed of different kinds of structural components that can be distinguished with respect to their topological properties. Exploring these structural components and their interactions can bring novel insights about the structure of a graph as well as additional power for graph representations. Unlike other data types such as images or sequences, graphs lack an inherent, consistent ordering of nodes. Therefore, a model that learns representations of graphs must treat isomorphic graphs – those that differ only by node permutations – as identical. To achieve this, ensuring permutation invariance is crucial during graph partitioning. This guarantees that graphs are divided based on their intrinsic structural properties, rather than being influenced by arbitrary node orderings. Inspired by these insights, we explore how permutation-invariant graph partitioning enhances GNN expressivity efficiently. Our key contributions are:

- **Theoretical insights:** We establish a novel connection between graph partitioning and graph isomorphism by introducing the concepts of *partition isomorphism* and *interaction isomorphism*, providing a rigorous theoretical framework for analyzing structural interactions.
- **GNN architecture:** We propose *Graph Partitioning Neural Networks* (GPNNs), a

new GNN architecture that integrates structural interactions through permutation-invariant graph partitioning to enhance graph representation learning.

- **Expressivity analysis:** We theoretically prove that GPNNs surpass the expressive power of 1-WL and efficiently approach the expressive power of 3-WL, addressing critical limitations of traditional GNNs.
- **Complexity analysis:** We show that GPNN variants achieve a balance between computational efficiency and representational power, making them practical for large-scale graph learning tasks.
- **Practical partitioning:** We explore efficient, permutation -invariant graph partitioning schemes, such as k-core and degree-based partitioning, to enable meaningful subgraph extraction and preserve structural properties.

Figure 3.1 illustrates the theoretical results on the expressivity of the proposed GPNN architecture in relation to k-WL (Cai et al., 1992) and interaction types across different graph partitioning schemes. To empirically verify the theoretical designs of GPNN, we conduct experiments on diverse graph benchmark tasks, demonstrating its superior performance over state-of-the-art models.

3.2 Graph Partitioning and Graph Isomorphism

In this section, we define permutation-invariant graph partitioning and explain its connection to graph isomorphism.

3.2.1 Preliminaries

Let G be a simple graph, and $V(G)$ and $E(G)$ refer to the node set and the edge set of G, respectively. Given $d \in \mathbb{N}$, for each $v \in V(G)$, the *d-hop neighborhood* is denoted as $N_d(v) = \{u \in V(G) \mid \delta(u, v) \leq d\}$, where $\delta(u, v)$ refers to the shortest-path distance between two nodes u and v. Note that $N_d(v)$ includes v itself (since $\delta(v, v) = 0$). For the *1-hop neighborhood*, we use the notation $N(v) = \{u \in V(G) \mid (v, u) \in E(G)\}$ to denote the set of direct neighbors of v, excluding v itself. Thus, $N_1(v) = N(v) \cup \{v\}$. An induced subgraph of G by $U \subseteq V(G)$ is the graph $G[U]$ with a node set U and edges only between nodes in U, denote as $G[U] \subseteq G$.

Two graphs G_1 and G_2 are *isomorphic*, denoted as $G_1 \simeq G_2$, if there exists a bijection $g : V(G_1) \to V(G_2)$ such that $(v, u) \in E(G_1)$ if and only if $(g(v), g(u)) \in E(G_2)$. A permutation π on a graph G is a bijection $\pi : V(G) \to V(G)$ such that $\pi(G) = (V(G), \pi(E(G)))$, where $\pi(E(G)) = \{(\pi(v), \pi(u)) | (v, u) \in E(G)\}$. A function f is *permutation-invariant* if and only if $f(G) = f(\pi(G))$ for any graph G and any permutation π on G. Let $\mathcal{G}$ be a set of graphs closed under isomorphism. A *subgraph invariant* is a function $\phi : \mathcal{G} \to \mathcal{G}$ that determines a subgraph $\phi(G) \subseteq G$ for each $G \in \mathcal{G}$, preserved under isomorphisms.

3.2.2 Graph Partitioning Scheme

We start by defining the graph partitioning scheme and relevant concepts. Then, we dive into the connection between graph partitioning and graph isomorphism by introducing the concepts of partition isomorphism and interaction isomorphism.

Definition 1 (Graph Partitioning Scheme)**.** *Let $P(\mathcal{G})$ be the powerset of $\mathcal{G}$ and $\Phi = \{\phi_j\}_{j\in[1,k]}$ be a family of subgraph invariants. A* graph partitioning scheme *is a permutation invariant function $f_\Phi : \mathcal{G} \to P(\mathcal{G})$ which partitions any graph $G \in \mathcal{G}$ into a set of subgraphs $\{S_1, \ldots, S_k\}$, where $\bigcup_{1\le i\le k} V(S_i) = V(G)$, $\bigwedge_{1\le i\ne j\le k} V(S_i) \cap V(S_j) = \emptyset$, and $\phi_j(G) = S_j$ for $j \in [1,k]$. Note that S_j can be an empty subgraph, i.e., $V(S_j) = \emptyset$ and $E(S_j) = \emptyset$.*

The permutation-invariant property of a graph partitioning scheme is crucial for exploring structural interactions. This property not only ensures that interactions among different structural components are consistently captured, under any node reordering, but also enables meaningful comparisons between subgraphs with the same indices, partitioned from different graphs by the same graph partition scheme through a family of subgraph invariants.

Based on graph partitioning schemes, we define two notions of isomorphism on graphs: one characterizes interactions within partitioned subgraphs and the other characterizes interactions across partitioned subgraphs.

Definition 2 (Partition-Isomorphism)**.** *Two graphs G and G' are* partition-isomorphic, *denoted as $G \overset{PI}{\simeq} G'$, with respect to a partitioning scheme f_Φ if there exists a bijective function $g : f_\Phi(G) \to f_\Phi(G')$ such that $f_\Phi(G) = \{S_1, \ldots, S_k\}$, $f_\Phi(G') = \{S'_1, \ldots, S'_k\}$, and $g(S_i) \simeq S'_i$ for $i = 1, \ldots, k$.*

A node $v \in V(G)$ is a *border node* with respect to f_Φ if there exist $u \in V(G)$ and two subgraphs $\{S_i, S_j\} \subseteq f_\Phi(G)$ such that $(v,u) \in E(G)$, $v \in V(S_i)$, $u \in V(S_j)$, and $S_i \ne S_j$. We use $V_B(G)$ to denote the set of all border nodes in G.

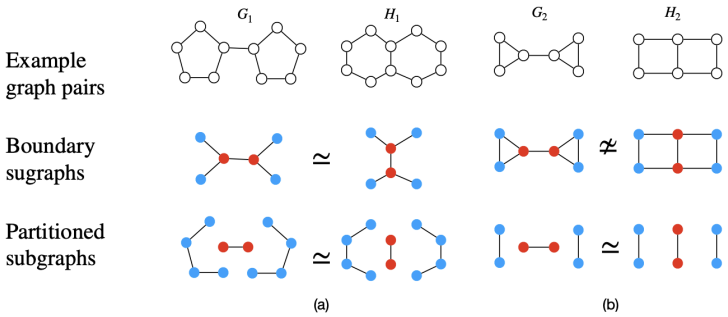
Definition 3 (Boundary Subgraph)**.** *The* boundary subgraph $B(G) = (V, E)$ *of a graph G with respect to f_Φ is defined by the node set $V = V_B(G)$ and the edge set $E = \{(v,u) \in E(G) \mid v, u \in V_B(G)\}$, where each edge in E connects two border nodes.*

Boundary subgraphs underpin structural interaction equivalence in partitioned graphs.

Definition 4 (Interaction-Isomorphism)**.** *Two graphs G and G' are* interaction-isomorphic, *denoted $G \overset{II}{\simeq} G'$, with respect to a partitioning scheme f_Φ, if and only if:*

- *$G \overset{PI}{\simeq} G'$, i.e., graphs are partition-isomorphic under f_Φ;*
- *$B(G) \simeq B(G')$, i.e., boundary subgraphs are isomorphic.*

The following establishes the connection between partition-isomorphism, interaction-isomorphism, and graph isomorphism. One direction of the proof follows directly from the definitions, while the other is shown using counterexample graphs in Figure 3.2.



Type	Pairs of Graphs (a)	Pairs of Graphs (b)
Graph Isomorphism	✗	✗
Interaction Isomorphism	✓	✗
Partition Isomorphism	✓	✓

Figure 3.2: Two pairs of non-isomorphic graphs, (G_1, H_1) and (G_2, H_2), are partitioned by node degrees, grouping nodes with the same degree. Boundary and partitioned subgraphs are shown, with different subgraphs highlighted in red and blue. (a) $G_1 \overset{PI}{\simeq} H_1$ and $G_1 \overset{II}{\simeq} H_1$; (b) $G_2 \overset{PI}{\simeq} H_2$, but $G_2 \overset{II}{\not\simeq} H_2$.

Theorem 1. *Fix a graph partitioning scheme. We have: (a) If $G \simeq G'$, then $G \overset{II}{\simeq} G'$, but not vice versa; (b) If $G \overset{II}{\simeq} G'$, then $G \overset{PI}{\simeq} G'$, but not vice versa.*

Proof. Suppose that the graph partitioning scheme is f_Φ. We first prove one direction of Statement (a), i.e., if $G \simeq G'$, then $G \overset{II}{\simeq} G'$. From the definition of graph isomorphism, we know that if $G \simeq G'$, then there is a bijective mapping $f : V(G) \to V(G')$ such that for any $(u, v) \in E(G)$, $(f(u), f(v)) \in E(G')$, and vice verse. By definition 2, we know that if $G \overset{PI}{\simeq} G'$, then there exists a bijective mapping $g : f_\Phi(G) \to f_\Phi(G')$ such that $g(S_i) \simeq S_i'$ for every $i \in [1, k]$. Thus, if $G \simeq G'$, we know that the same bijective mapping from graph isomorphism $f : V(S_i) \to V(S_i')$ satisfies that for any $(u, v) \in E(S_i)$, $(f(u), f(v)) \in E(S_i')$ for any $i \in [1, k]$. Then, we consider the boundary subgraphs $B(G) = (\mathcal{V}, \mathcal{E})$ and $B(G') = (\mathcal{V}', \mathcal{E}')$. By definition 4, it requires to show a bijective function $g : B(G) \to B(G')$ such that $(v, u) \in \mathcal{E}$ iff $(g(v), g(u)) \in \mathcal{E}'$. This can also be satisfied by f where $f : \mathcal{V} \to \mathcal{V}'$ such that for any $(u, v) \in \mathcal{E}$, $(f(u), f(v)) \in \mathcal{E}'$. Therefore, such a bijective mapping f of graph isomorphism satisfies both of the conditions of interaction-isomorphism. This direction is proven. However, the converse direction of Statement (a) does not hold. The pair of graphs shown in Figure 3.2(a) is interaction-isomorphic, but not isomorphic.

For Statement (b), the first direction (i.e., if $G \overset{II}{\simeq} G'$, then $G \overset{PI}{\simeq} G'$) can easily follow from the definition of interaction-isomorphism. The converse do not hold, which can be proven by the pair of graphs shown in Figure 3.2(b).

The proof is complete. □

3.3 Graph Partitioning Neural Networks

In this section, we introduce *Graph Partitioning Neural Networks* (GPNN), a novel GNN model which can integrate structural interactions into representation learning.

We begin by defining partitioning colouring, where nodes and edges are assigned colours to reflect permutation-invariant partitions generated by a graph partition scheme.

Definition 5 (Partition Colouring)**.** *Let C_V be a set of node colours, C_E be a set of edge colours containing $\{c_e, c_a, c_n\}$, $C_V \cap C_E = \emptyset$, $\{S_1, \ldots, S_k\}$ be a set of subgraphs generated by a graph partitioning scheme over G, and $\pi(\cdot)$ be an injective hashing function. A* partition colouring *$\lambda = (\lambda_V, \lambda_E)$ consists of the following:*

– node colouring: *Each node $v \in V(G)$ is assigned a* node colour*, $\lambda_V : V(G) \to C_V$ such that $\lambda_V(v) = \lambda_V(u)$ if and only if $v \in V(S_i)$ and $u \in V(S_i)$.*

– Edge colouring: *Each pair of distinct nodes $\{v, u\} \subseteq V(G)$ is assigned an* edge colour*, $\lambda_E : V(G) \times V(G) \to C_E$ such that*

$$\lambda_E(v,u) = \begin{cases} \pi(\lambda_V(v), c_e, \lambda_V(u)) & (v,u) \in E(G), \lambda_V(v) = \lambda_V(u); \\ \pi(\lambda_V(v), c_a, \lambda_V(u)) & (v,u) \in E(G), \lambda_V(v) \neq \lambda_V(u); \\ \pi(\lambda_V(v), c_n, \lambda_V(u)) & (v,u) \notin E(G). \end{cases}$$

Here, $\{c_e, c_a, c_n\}$ represents three types of interactions: *inter-interaction*, *intra-interaction*, and *non-interaction*, respectively. Intuitively, inter-interactions refer to interactions between nodes within the same partitioned subgraph, intra-interactions refer to interactions between nodes across different partitioned subgraphs, and non-interactions indicate that no direct interactions occurs between nodes.

Definition 6 (Coloured Neighbourhood)**.** *For each node $v \in V(G)$, the neighbourhood $N_d(v)$ consists of a set of neighbouring node subsets $\{N_d^1(v), \ldots, N_d^k(v)\}$, each of which is coloured with a distinct colour under λ_V such that, for any two nodes $\{u, w\} \subseteq N_d(v)$, $\lambda_V(u) = \lambda_V(w)$ if and only if $\{u, w\} \subseteq N_d^j(v)$ for exactly one $j \in [1, k]$.*

Below we present the message-passing scheme of GPNN. Given a partition colouring $\lambda = (\lambda_V, \lambda_E)$, let $\beta_v^{(0)} = \lambda_V(v)$ and $\gamma_v^{(0)} = \lambda_V(v)$ for any $v \in V(G)$ and $\alpha_{vu}^{(0)} = \lambda_E(v,u)$ for any $v, u \in V(G)$. We learn the node embedding $\beta_v^{(\ell+1)}$ of each node v at the $(\ell+1)$-th iteration as

$$\beta_v^{(l+1)} = \textsc{Upd}\left(\gamma_v^{(l)}, \textsc{Agg}\{\!\{\gamma_u^{(l)} | u \in N(v)\}\!\}\right). \tag{3.1}$$

$\textsc{Upd}(.)$ and $\textsc{Agg}(.)$ are injective and permutation-invariant functions that represent update and aggregate operations in GNNs, respectively. Then, we define the interaction embedding $\alpha_{vu}^{(\ell+1)}$ of (v,u) at the $(\ell+1)$-th iteration as

$$\alpha_{vu}^{(\ell+1)} = \textsc{Upd}\Big(\alpha_{vu}^{(\ell)}, \textsc{Agg}\Big(\{\!\{(\alpha_{vw}^{(\ell)}, \alpha_{uw}^{(\ell)}) | w \in N_d(v)\}\!\}\Big)\Big). \tag{3.2}$$

We then aggregate the embeddings of neighboring nodes and their corresponding interaction embeddings from different partitions (i.e., the coloured neigbourhood

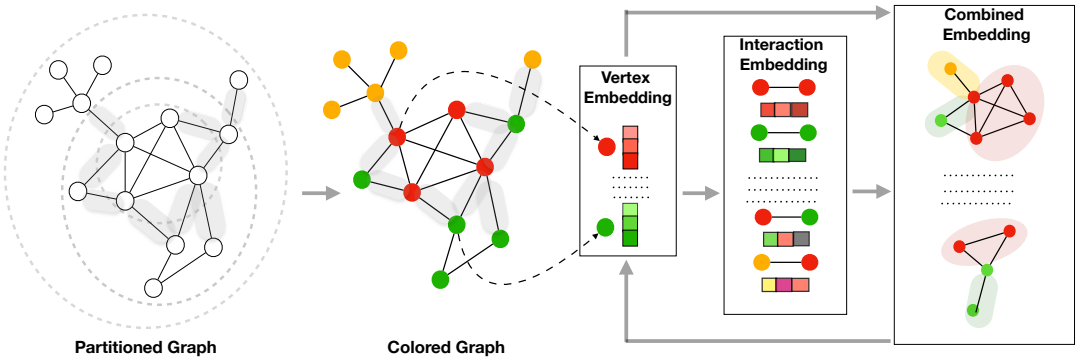


Figure 3.3: A high-level workflow of λ_{core}-GPNN for an input graph. The intra-edges are marked using grey color. GPNN generates node representations by considering interactions within and between partitions.

$\{N_d^1(v), \ldots, N_d^k(v)\}$) into a combined embedding γ_v as

$$\gamma_v^{(\ell+1)} = \textsc{Cmb}\Big(\textsc{Agg}\left(\{\!\{(\beta_u^{(\ell+1)}, \alpha_{vu}^{(\ell+1)}) | u \in N_d^1(v)\}\!\}\right), \\ \ldots, \textsc{Agg}\left(\{\!\{(\beta_u^{(\ell+1)}, \alpha_{vu}^{(\ell+1)}) | u \in N_d^k(v)\}\!\}\right)\Big). \tag{3.3}$$

$\textsc{Cmb}(.)$ is an injective combining function, closed under permutation. Note that (v, u) of $\alpha_{vu}^{(\ell+1)}$ may correspond to inter-interaction, intra-interaction, or non-interaction, depending on not only how v and u are connected in G but also how they are partitioned by a graph partitioning scheme.

Remark 1. *Our model can be integrated into any existing GNN models as a plugin. Let h_v^{gnn} be a node embedding from an existing GNN model (e.g., GIN (Xu et al., 2019) and GCN (Kipf and Welling, 2017). By combining h_v^{gnn} with $\gamma_v^{(\ell+1)}$, we obtain the representation for each node v as*

$$h_v = \textsc{Cmb}\left(h_v^{gnn}, \gamma_v^{(\ell+1)}\right). \tag{3.4}$$

An illustration of the high-level workflow of GPNN is provided in Figure 3.3.

3.4 Theoretical Analysis

3.4.1 Expressivity Analysis

In this section, we discuss the expressivity of GPNN from two aspects: (1) How does a partition colouring, determined by a chosen graph partitioning scheme, affect the expressivity of GPNN? (2) How do different types of interactions affect the expressivity of GPNN?.

We use λ-GPNN$^{\delta}$ to refer to the variants of GPNN with a partition colouring λ, $d = 1$ (i.e., $N_d(v)$ contains direct neighbours of v and v itself), and $\delta \in \{\star, \diamond, \dagger\}$, where

$\star$, $\diamond$, and † denote that (v, u) in Eq. 3.2 considers intra-interactions ($E^{\star}$), both inter-interactions and intra-interactions ($E^{\diamond}$), and all interactions ($E^{\dagger}$), respectively. We also introduce the notion of λ-equivalence to compare the expressivity of different graph partitioning schemes.

Definition 7 (λ-Equivalence)**.** *Let $\lambda = (\lambda_V, \lambda_E)$ be a partition colouring. Two graphs G and H are* λ-equivalent*, denoted as $G \equiv_{\lambda} H$, if they are indistinguishable by λ, i.e.,*

- *for each $c \in C_V$, $|\{v \in V(G) | \lambda_V(v) = c\}| = |\{v' \in V(H) | \lambda_V(v') = c\}|$;*
- *for each $c \in C_E$, $|\{\{v, u\} \subseteq V(G) | \lambda_E(v, u) = c\}| = |\{\{v', u'\} \subseteq V(H) | \lambda_E(v', u') = c\}|$.*

Given two partition colourings λ_1 and λ_2, we say that λ_1 *is at least as expressive as* λ_2, denoted as $\lambda_1 \sqsupseteq \lambda_2$, if and only if for any two graphs $\{G, H\} \subseteq \mathcal{G}$, $G \equiv_{\lambda_2} H$ whenever $G \equiv_{\lambda_1} H$. Similarly, we use $\lambda_1 \sqsupset \lambda_2$ and $\lambda_1 \equiv \lambda_2$ to indicate that λ_1 is *strictly more expressive than* or *equally expressive as* λ_2, respectively. These notations can also be extended to k-WL and GPNNs.

A partition colouring is *trivial*, denoted as $\lambda_{\perp}$, if $f_{\Phi}(G) = \{G\}$ for any $G \in \mathcal{G}$, i.e., all nodes assigned the same colour. A partition colouring is *complete*, denoted as $\lambda_{\top}$, if $f_{\Phi}(G) = f_{\Phi}(H) \Leftrightarrow G \simeq H$ for any $\{G, H\} \subseteq \mathcal{G}$, i.e., orbit colouring (McKay and Piperno, 2014). The expressivity of a partition colouring λ is lower bounded by $\lambda_{\perp}$ and upper bounded by $\lambda_{\top}$.

The following proposition describes how the expressivity of λ-GPNN$^{\delta}$ compares to 1-WL and 3-WL when λ is trivial. The expressive power of $\lambda_{\perp}$-GPNN$^{\delta}$ is at least as expressive as 1-WL but upper bounded by 3-WL.

Proposition 1. *The following hold: (1) $\lambda_{\perp}$-GPNN$^{\star} \equiv$ 1-WL; (2) $\lambda_{\perp}$-GPNN$^{\star} \sqsubseteq \lambda_{\perp}$-GPNN$^{\diamond} \sqsubseteq \lambda_{\perp}$-GPNN†; (3) $\lambda_{\perp}$-GPNN$^{\dagger} \sqsubseteq$ 3-WL.*

Proof. We first prove Statement (1) $\lambda_{\perp}$-GPNN$^{\star} \equiv$ 1-WL. According to the model design, $\lambda_{\perp}$-GPNN* starts with Eq. 3.1. Since $\lambda_{\perp}$ is trivial, $E^{\star} = \emptyset$, meaning that all nodes are initially assigned the same colour. In Eq. 3.1, a node embedding is updated based on its previous embedding and the embeddings of its direct neighbours. This implies that for, without any interation embedding, for any iteration of Eq. 3.1 has expressive power upper bounded by 1-WL, as demonstrated in Theorem 3 of Xu et al. (2019). $\lambda_{\perp}$-GPNN* does not account for any structural interactions. Therefore, Eq. 3.2 does not contribute additional expressivity. Given that Eq. 3.3 is an injective combination of Eq.. 3.1 and Eq. 3.2 from the neighbours of each node and the node itself, the expressive power of the final embedding in any iteration is also upper-bounded by 1-WL. This completes the first part of the proof.

We then prove Statement (2) $\lambda_{\perp}$-GPNN$^{\star} \sqsubseteq \lambda_{\perp}$-GPNN$^{\diamond} \sqsubseteq \lambda_{\perp}$-GPNN†. Given that $E^{\star} \subseteq E^{\diamond} \subseteq E^{\dagger}$, it is trivial to see that λ-GPNN$^{\star} \sqsubseteq \lambda$-GPNN$^{\diamond} \sqsubseteq \lambda$-GPNN†.

Finally, we prove Statement (3) $\lambda_{\perp}$-GPNN$^{\dagger} \sqsubseteq$ 3-WL. When accounting for all of inter-interactions, intra-interactions and non-interactions in a graph, the expressive power of Eq. 3.2 is upper-bounded by 2-FWL algorithm (Huang and Villar, 2021; Zhou et al., 2023) (i.e., equivalent to 3-WL in terms of their ability in distinguishing

non-isomorphic graphs). Since λ is trivial, the initial node colours by $\lambda_\perp$ do not provide any additional power beyond that of 1-WL. Therefore, for any iteration, the final embeddings generated by any GPNN variant are upper bounded by 3-WL, meaning that λ-GPNN$^\dagger \sqsubseteq$ 3-WL. □

When considering the same interaction type, GPNN$^\delta$ maintains the expressivity order of their respective partition colourings, as stated in the theorem below.

Theorem 2. *Let λ_1 and λ_2 be two partition colourings with $\lambda_1 \sqsupseteq \lambda_2$. Then λ_1-GPNN$^\delta \sqsupseteq \lambda_2$-GPNN$^\delta$ for any $\delta \in \{\star, \diamond, \dagger\}$.*

Proof. By the definition of λ-GPNN$^\delta$, GPNN starts with a graph in which nodes are coloured by λ. Since $\lambda_1 \sqsupseteq \lambda_2$, we know that, for any two nodes $\{v, u\} \subseteq V(G)$, if v and u are assigned with the same colour by λ_1, then they must be assigned with the same colour by λ_2; the converse does not necessarily hold. Then, by the definition of GPNN in terms of Equations 3.1, 3.2, and 3.3, we know that if two nodes $\{v, u\} \subseteq V(G)$ are assigned with different colours by λ, then they would always have different colours after applying λ-GPNN$^\delta$ with any number of layers. That is, λ-GPNN$^\delta$ always *refines* a partitioning colouring λ. Since the same equations (Equations 3.1, 3.2, and 3.3) are applied by λ_1-GPNN$^\delta$ and λ_2-GPNN$^\delta$ on nodes in G, which only differ in partitioning colourings, and also the functions $\textsc{Agg}(.), \textsc{Cmb}(.)$ and $\textsc{Upd}(.)$ used in GPNN are injective, any two nodes $v, u \in V(G)$ must have different colours by applying λ_1-GPNN$^\delta$ whenever they have different colours by applying λ_2-GPNN$^\delta$, but not vice versa. □

The following theorem establishes expressivity bounds of λ-GPNN$^\delta$ in terms of the partition colouring λ and k-WL. For clarity, we define $max(\lambda, \text{k-WL}) = \text{k-WL}$ if $\lambda \sqsubseteq$ k-WL, or $max(\lambda, \text{k-WL}) = \lambda$ if $\lambda \sqsupseteq$ k-WL.

Theorem 3. *Let λ be a partition colouring satisfying $\lambda \sqsubseteq k$-WL or $\lambda \sqsupseteq k$-WL for some $k \in \mathbb{N}$. Then $max(\lambda, 1\text{-WL}) \sqsubseteq \lambda$-GPNN$^\delta$ and λ-GPNN$^\delta \sqsubseteq max(\lambda, 3\text{-WL})$ for any $\delta \in \{\star, \diamond, \dagger\}$.*

Proof. From Theorem 1, we know that $\lambda_\perp$-GPNN$^\star \equiv$ 1-WL. By Theorem 2, for any non-trivial λ, λ-GPNN* is at least as expressive as $\lambda_\perp$-GPNN*. Hence, we can deduce that λ-GPNN* is at least lower bounded by 1-WL. When λ is strictly more expressive than 1-WL, any iteration of GPNN will be strictly more expressive than the corresponding 1-WL iteration as Eq. 3.1 and Eq. 3.3 together can only further refine initial node colours. In this case, λ-GPNN* is lower bounded by λ. Therefore, we can conclude that, for any λ, $\max(\lambda, \text{1-WL}) \sqsubseteq \lambda$-GPNN*. Given that $E^\star \subseteq E^\diamond \subseteq E^\dagger$, we can deduce that λ-GPNN$^\star \sqsubseteq \lambda$-GPNN$^\diamond \sqsubseteq \lambda$-GPNN†. Therefore, this relationship also holds for the other two GPNN variants, as they are at least as expressive as the corresponding λ-GPNN*.

From Theorem 1, we also know that $\lambda_\perp$-GPNN$^\delta$ is upper bounded by 3-WL. Additionally, if $\lambda \sqsubseteq$ 3-WL, then for any non-trivial λ, λ-GPNN$^\delta$ is also upper bounded by 3-WL. This is because initial node coloring is upper bounded by 3-WL, and any

interaction learning in GPNNs is similarly constrained by 3-WL, leading to final embeddings that are upper bounded by 3-WL. When λ is strictly more powerful than 3-WL, GPNN is as expressive as λ, since no iteration of GPNN can further refine the color beyond λ. Therefore, for any λ, we have λ-GPNN$^{\dagger} \sqsubseteq \max(\lambda, 3\text{-WL})$. □

We show that for a partition coloring λ below 3-WL, the expressive power of λ-GPNN$^{\delta}$ is non decreasing when more interactions are captured into representations. However, when a partition coloring λ is at least as expressive as 3-WL, the expressive power of λ-GPNN$^{\delta}$ remains unchanged.

Lemma 1. *When $\lambda \sqsubset$ 3-WL, λ-GPNN$^{\dagger} \sqsupseteq \lambda$-GPNN$^{\diamond} \sqsupseteq \lambda$-GPNN*. When $\lambda \sqsupseteq$ 3-WL, λ-GPNN$^{\dagger} \equiv \lambda$-GPNN$^{\diamond} \equiv \lambda$-GPNN*.*

Proof. First, we consider the case where $\lambda \sqsubset$ 3-WL. Since $E^{\star} \subseteq E^{\diamond} \subseteq E^{\dagger}$, for any λ, we can derive the following: λ-GPNN$^{\star} \sqsubseteq \lambda$-GPNN$^{\diamond} \sqsubseteq \lambda$-GPNN†.

Next, we consider the case where $\lambda \sqsupseteq$ 3-WL. In this case, by $max(\lambda, 1\text{-WL}) \sqsubseteq \lambda$-GPNN$^{\delta}$, as stated in Theorem 3, we know that the expressive power of λ-GPNN$^{\delta}$ is lower-bounded by λ. Similarly, by λ-GPNN$^{\delta} \sqsubseteq max(\lambda, 3\text{-WL})$ from Theorem 3, we know that the expressive power of λ-GPNN$^{\delta}$ is also upper-bounded by λ. Therefore, all GPNN variants (λ-GPNN$^{\delta}$) will have the same expressive power, determined by the initial partition coloring λ, regardless of the interaction types $\delta \in \{\star, \diamond, \dagger\}$ considered. Thus, λ-GPNN$^{\dagger} \equiv \lambda$-GPNN$^{\diamond} \equiv \lambda$-GPNN* holds. □

Based on Lemma 1, the following theorem compares the expressivity of different GPNN variants.

Theorem 4. *For any partition colouring λ, λ-GPNN$^{\dagger} \sqsupseteq \lambda$-GPNN$^{\diamond} \sqsupseteq \lambda$-GPNN*.*

Proof. Given that λ determines the initial node colouring in GPNNs, any pair of graphs distinguished by λ will also be distinguished by any GPNN variant. Additionally, since $E^{\star} \subseteq E^{\diamond} \subseteq E^{\dagger}$, it follows that for any λ, GPNN$^{\star} \sqsubseteq$ GPNN$^{\diamond} \sqsubseteq$ GPNN†.This completes the proof. □

Remark 2. *The expressivity of λ-GPNN$^{\delta}$ stems from two key sources: the capacity of a chosen graph partitioning scheme, as reflected by the partition colouring λ, and the ability to capture different types of structure interactions via $\delta \in \{\star, \diamond, \dagger\}$. The former sets the lower bound of λ-GPNN$^{\delta}$ in its ability to distinguish non-isomorphic graphs, ranging from a trivial colouring to a complete colouring. For the latter, λ-GPNN$^{\delta}$ addresses partition isomorphism and interaction isomorphism by considering various types of interactions. Inter-interactions enable λ-GPNN$^{\delta}$ to handle interactions within partitions, as required by partition isomorphism, while intra-interactions capture interactions across partitions, as considered by interaction isomorphism. The model capacity of GPNN$^{\delta}$ alone is bounded between 1-WL as a lower bound and 3-WL as an upper bound. Consequently, non-isomorphic graph pairs that cannot be separated by* $\max(\lambda, 3\text{-WL})$ *remain indistinguishable by λ-GPNN$^{\delta}$, regardless of the interaction type chosen. This includes graph pairs that are structurally identical under the chosen partitioning scheme, such as highly regular graphs where all nodes share the same local structure and the partition colouring provides no meaningful separation.*

3.4.2 Complexity Analysis

Generally, 1-WL and traditional GNNs that are upper-bounded by 1-WL, such as GIN (Xu et al., 2019), have a time complexity of $O(|E|)$. In contrast, 3-WL and existing GNNs with provable 3-WL expressive power are known to be computationally expensive, with a time complexity of at least $O(|V|^3)$ (Maron et al., 2019a). As a result, while these GNNs can offer strong expressive power, they are often impractical for real-world applications. The time complexities of our GPNNs fall between those of 1-WL and 3-WL, offering a balanced trade-off between computational efficiency and representational power. Specifically, the time complexities of GPNN*, GPNN$^\diamond$, and GPNN† are $O(|E| + d_{\max}|E^\star|)$, $O(|E| + d_{\max}|E^\diamond|)$, and $O(|E| + d_{\max}|V|^2)$, respectively, where $d_{\max} = \textsc{Max}(\{|N_d(v)||v \in V(G)\})$. Table 3.1 provides a detailed time complexity analysis of GPNNs, covering both node embedding and interaction embedding computations, and compares them with traditional GNNs and 3-WL.

	GIN	GCN	GPNN*	GPNN$^\diamond$	GPNN†	3-WL
VE	$O(\|E\|)$	$O(\|E\|)$	$O(\|E\|)$	$O(\|E\|)$	$O(\|E\|)$	-
IE	-	-	$O(d_{\max}\|E^\star\|)$	$O(d_{\max}\|E^\diamond\|)$	$O(d_{\max}\|V\|^2)$	$O(\|V\|^3)$

Table 3.1: Time complexity analysis of different embedding computations, where **VE** refers to node embeddings and **IE** refers to interaction embeddings.

As shown in Table 3.1, GPNNs are more computationally efficient than 3-WL with a time complexity $O(|V|^3)$. This is due to two reasons. Firstly, real-world graphs are often sparse, and thus $|E^\star|$ and $|E^\diamond|$ are significantly smaller than $|V|^2$, i.e., $|E^\star| < |E^\diamond| \ll |V^2|$. Secondly, GPNNs only consider local neighbourhoods for learning interaction embeddings, i.e., nodes within a d-hop neighborhood, while 3-WL adopts a global neighbourhood, where the neighbours of each edge involve every node in the entire graph.

It is worth noting that the graph partitioning scheme used by GPNNs should be computationally efficient, i.e., in linear or polynomial time in terms of the size of an input graph. This ensures that graph partitioning can be processed efficiently as a preprocessing step. In our experiments, we use a graph partitioning scheme with a time and space complexity of $O(|E|)$, which will be discussed in the next section.

3.5 Practical Choices of Partitioning Schemes

One might ask how to choose a graph partitioning scheme. In practice, there are many design options available. However, two important criteria must be met: *permutation invariance* and *computational efficiency*. To address these criteria, we first consider a graph partitioning scheme based on the k-core property (Malliaros et al., 2020), which is both permutation invariant and computationally efficient.

Definition 8 (k-Core Property)**.** *Let* $\textsc{Deg}_S(v)$ *denote the degree of node* v *in a subgraph* S*. A subgraph* S *has the* k-core property *on a graph* G *if* S *is the largest induced subgraph of* G *satisfying:* $\forall v \in V(S)\ \ \textsc{Deg}_S(v) \geq k$.

Let φ_j denote the j-core property and $\Phi_{core} = \{\phi_j\}_{j\in[0,k]}$ where $\phi_j = \varphi_j \wedge \neg\varphi_{j+1}$ satisfying the j-core property but not the j+1-core property. The graph partitioning scheme $f_{\Phi_{\text{core}}}$ corresponds to the shell decomposition (Alvarez-Hamelin et al., 2005), which decomposes a graph G into a set of subgraphs, namely *shells*. Let $\phi \circ \psi$ denote the sequential composition of two properties ϕ and ψ. Then Φ_{core} can be further refined:

- $\Phi_{\text{core-degree}} = \{\phi'_j\}_{j\in[0,2k]}$ where $\phi'_{2j-1} = \phi_j \circ \psi_j$, $\phi'_{2j} = \phi_j \circ \neg\psi_j$, and $\psi_j = \forall v \in V(S_j)\ \textsc{Deg}_{S_j}(v) = j$.
- $\Phi_{\text{core-onion}} = \{\phi'_{ji}\}_{j\in[0,k]}$ where $\phi'_{ji} = \phi_j \circ \psi^1_j \circ, \ldots, \circ\psi^i_j$, and ψ^i_j refers to the i-th iteration to remove nodes with the lowest degree in the j-core subgraph.

Alternatively, we may consider graph partitioning schemes based on other properties such as node degrees. That is $\Phi_{\text{degree}} = \{\phi''_j\}_{j\in[0,k]}$ where ϕ''_j is defined as the set of nodes with a degree of j, and k is the maximum node degree in the graph. When computational resources are sufficient, more computationally demanding schemes can be considered, such as Φ_{triangle} where the partitioning is based on the number of triangles in which each node participates.

A question that might arise is how a partition colouring λ determined by such f_Φ relates to k-WL. The following theorem states that the expressive power of the partition colourings based on the k-core and degree properties are upper bounded by 1-WL.

Theorem 5. *Let λ be a partition colouring based on f_Φ where $\Phi \in \{\Phi_{core}, \Phi_{degree}\}$. Then $G \equiv_\lambda H$ whenever $G \equiv_{1\text{-}WL} H$.*

Proof. We first prove the theorem for Φ_{core}. Let $\lambda_{1\text{-}WL}$ and λ_{core} be two partition colorings generated by 1-WL and core decomposition based on the k-core property (Definition 8), respectively. 1-WL is a coloring algorithm that iteratively captures the neighborhood degree sequence of each node in a graph (Smith, 2019). If $\lambda_{1-WL}(u) = \lambda_{1-WL}(v)$, then the nodes u and v must have identical neighborhood degree sequence. Core decomposition is an iterative peeling algorithm that, given an input graph G, the peeling process starts from nodes of the lowest degree in the graph G. Assume that there are two nodes u and v s.t. $\lambda_{1-WL}(u) = \lambda_{1-WL}(v)$ and $\lambda_{core}(u) \neq \lambda_{core}(v)$. Since $\lambda_{core}(u) \neq \lambda_{core}(v)$, u and v must appear in different peeling iterations; otherwise, they would have the same color from the core decomposition. Let P_u and P_v be two sets containing all nodes removed in the peeling iterations before u and v are removed by the core decomposition algorithm, respectively. Since $\lambda_{core}(u)$ and $\lambda_{core}(v)$ are different, K-shell numbers (Alvarez-Hamelin et al., 2005) of u and v must be distinct. Without loss of generality, we assume that the K-shell value of v is larger than the K-shell value of u. Then, P_u must be a proper subset of P_v (i.e. $P_u \subset P_v$). In other words, P_u cannot be the same as P_v. Thus, P_u and P_v should have different degree sequences. Since, u and v have different degree sequences around them, their coloring received from 1-WL should be different. This contradicts the above assumption. Therefore, $\lambda_{1-WL}(u) = \lambda_{1-WL}(v) \implies \lambda_{core}(u) = \lambda_{core}(v)$. This

also implies that if the colors of any two nodes in two graphs G and H are the same by 1-WL, then their node colors by core decomposition must also be the same. Therefore, under an injective graph readout function, if $G \equiv_{\text{1-WL}} H$, then $G \equiv_{\text{core}} H$.

Then, we prove the theorem for Φ_{degree}. Let $\lambda_{1\text{-}WL}$ and λ_{degree} be two partition colourings generated by 1-WL and degrees of nodes, respectively. Since the 1^{st} iteration of 1-WL captures the 1-hop neighborhood degree sequence of each node, the colouring produced by the 1^{st} iteration of 1-WL must be equivalent to the colouring of λ_{degree}. Further iterations of 1-WL would only refine the colours of nodes. This implies that if the colours of any two nodes in two graphs G and H are the same by 1-WL, then their node colours by λ_{degree} must also be the same, i.e. $\forall u, v \in V(G), \lambda_{1-WL}(u) = \lambda_{1-WL}(v) \implies \lambda_{degree}(u) = \lambda_{degree}(v)$, although the converse does not generally hold. Therefore, under an injective graph readout function, if $G \equiv_{\text{1-WL}} H$, then $G \equiv_{\text{degree}} H$. □

$\Phi_{\text{core-degree}}$ refines Φ_{core} by node degrees, while $\Phi_{\text{core-onion}}$ refines Φ_{core} based on degree correlations within each shell. Theorem 5 applies to both schemes since the refinements are based on node degrees. However, it does not hold for Φ_{triangle}, which can distinguish graphs that are indistinguishable by 1-WL.

3.6 Experimental Design

We conduct experiments on three widely used benchmark tasks: graph classification, graph regression, and node classification.

3.6.1 Datasets

We evaluate GPNNs on 18 benchmark datasets. For graph classification, we use six small-scale real-world datasets from TU Datasets (Morris et al., 2020a) and three large-scale molecular datasets from Open Graph Benchmark (OGB) (Hu et al., 2020). We also consider ZINC (Dwivedi et al., 2023) for graph regression, along with five widely used benchmark datasets for node classification including both hemophilic and heterophilic graph data (Sen et al., 2008; Craven et al., 2000).

Table 3.2 and Table 3.3, provide a summary of statistics for datasets that we have considered in all our experiments. Table 3.2 consists of the dataset statistics for graph classification and graph regression, whereas Table 3.3 contains the dataset statistics for node classification.

3.6.2 Baselines

For graph classification, we choose GIN (Xu et al., 2019) as the base model for GPNNs and comparing them with GNNs that has expressive power equivalent or beyond 1-WL: GIN (Xu et al., 2019), GraphSNN (Wijesinghe and Wang, 2022), GIN-AK+ (Zhao et al., 2022a), and KP-GIN (Feng et al., 2022), CIN (Bodnar et al., 2021), PPGN (Maron et al., 2019a), and ESAN (Bevilacqua et al., 2022). In graph regression, we

Table 3.2: Dataset statistics for graph classification and regression

Datasets	Task Type	# Graphs	Avg #Nodes	Avg #Edges	# Classes
MUTAG	Graph Classification	188	17.93	19.79	2
PTC_MR	Graph Classification	344	14.29	14.69	2
PTC_FM	Graph Classification	349	14.11	14.48	2
BZR	Graph Classification	405	35.75	38.36	2
COX2	Graph Classification	467	41.22	43.45	2
ENZYMES	Graph Classification	600	32.63	62.14	6
DHFR	Graph Classification	756	42.43	44.54	2
IMDB-B	Graph Classification	1,000	19.77	96.53	2
PROTEINS	Graph Classification	1,113	39.06	72.82	2
ogbg-moltox21	Graph Classification	7,831	18.6	19.3	2
ogbg-moltoxcast	Graph Classification	8,576	18.8	19.3	2
ogbg-molhiv	Graph Classification	41,127	25.5	27.5	2
ZINC	Graph Regression	12,000	23.1	49.8	1

Table 3.3: Dataset statistics for node classification

Datasets	# Nodes	# Edges	# Features	# Classes
Cora	2708	5278	1433	7
CiteSeer	3327	4552	3703	6
Wisconsin	251	466	1703	5
Cornell	183	277	1703	5
Texas	183	279	1703	5

employ GIN (Xu et al., 2019), GCN (Kipf and Welling, 2017), PPGN (Maron et al., 2019a), DGN (Beaini et al., 2021), Deep LRP (Chen et al., 2020b), and CIN (Bodnar et al., 2021) as baselines. For node classification, we evaluate performance changes by incorporating our GPNN variants into standard GNNs, using GIN (Xu et al., 2019), GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017) as base models.

3.6.3 Experimental Setups

In graph classification, we adopt the setup outlined by Hu et al. (2020) for the OGB datasets. For TU datasets, we present results under two distinct configurations: Setting 1 as described by Xu et al. (2019) and Setting 2 by Feng et al. (2022). In the first setting, we report the average test accuracy across 10 folds and the standard deviation for the epoch with the highest mean accuracy. In the second setting, we provide the mean and standard deviation of the best accuracy across all test folds. In graph regression, we follow the experimental setup described by Dwivedi et al. (2023). For node classification, we follow the setup suggested in Pei et al. (2020b), where nodes in the datasets are randomly split into 60%, 20%, and 20% for training, validation, and testing, respectively.

3.6.4 Hyper-parameters

In our experiments, the hyper-parameters of GPNN are searched in the following ranges: the number of layers is within $\{1, 2, 3, 4, 5\}$, $d \in \{1, 2, 3, 4, 5\}$, dropout $\in \{0.0, 0.1, 0.2, 0.5, 0.6\}$, learning rate $\in \{0.009, 0.01\}$, batch size $\in \{32, 64, 128\}$, hidden units $\in \{32, 64, 100\}$, and the number of epochs $\in \{100, 200, 300, 500\}$. We use the Adam algorithm (Kingma, 2015) as our optimizer. For ZINC and OGB datasets, the initial learning rate decays with a factor of 0.5 after every 10 epochs. We do not use any learning rate decay technique for TU datasets.

3.7 Results and Discussion

3.7.1 Graph Classification

Table 4.5 reports the graph classification results. GPNN* and GPNN$^\diamond$ consistently outperform or match the baseline models. Compared to GPNN$^\diamond$, GPNN* handles fewer interactions but performs on par or exceeds the performance of GPNN$^\diamond$. This underscores that the presence of meaningful interactions are more paramount to learn graph structure than exhaustively exploring all interactions at the expense of computational efficiency. Unlike many existing models (e.g., Bodnar et al. (2021)), GPNNs do not rely on hand-crafted, domain-specific structural information, such as cycles, which are often correlated with molecular datasets like OGB.

Methods	MUTAG	PTC-MR	PROTEINS	IMDB-B	BZR	COX2	ogbg-molhiv	ogbg-moltox21
[1]GIN	89.40 ±5.6	64.60 ±7.0	75.90 ±2.8	75.10 ±5.1	85.60 ±2.0	82.44 ±3.0	75.58 ±1.4	74.91 ±0.5
[1]GraphSNN	91.24 ±2.5	66.96 ±3.5	76.51 ±2.5	**76.93** ±3.3	**88.69** ±3.2	82.86 ±3.1	78.51 ±1.7	75.45 ±1.1
[1]ESAN	91.00 ±7.1	**69.20** ±6.5	77.10 ±4.6	**77.10** ±3.0	N/A	N/A	76.43 ±2.1	75.12 ±0.50
[1]CIN	**92.70** ±6.1	68.20 ±5.6	77.00 ±4.3	75.60 ±3.7	N/A	N/A	**80.94** ±0.6	N/A
[1]GIN-AK+	91.30 ±7.0	**68.20** ±5.6	77.10 ±5.7	75.60 ±3.7	N/A	N/A	**79.61** ±1.1	N/A
[1]KP-GIN	92.20 ±6.5	66.80 ±6.8	75.80 ±4.6	76.60 ±4.2	N/A	N/A	N/A	N/A
[1]PPGN	90.55 ±8.7	66.17 ±6.5	**77.20** ±4.7	73.00 ±5.8	N/A	N/A	N/A	N/A
[2]GIN	92.80 ±5.9	65.60 ±6.5	78.80 ±4.1	78.10 ±3.5	91.05 ±3.4	88.87 ±2.3	75.58 ±1.4	74.91 ±0.5
[2]GraphSNN	94.70 ±1.9	70.58 ±3.1	78.42 ±2.7	78.51 ±2.8	91.12 ±3.0	86.28 ±3.3	78.51 ±1.7	75.45 ±1.1
[2]GIN-AK+	95.00 ±6.1	74.10 ±5.9	78.90 ±5.4	77.30 ±3.1	N/A	N/A	79.61 ±1.1	N/A
[2]KP-GIN	95.60 ±4.4	76.20 ±4.5	79.50 ±4.4	**80.70** ±2.6	N/A	N/A	N/A	N/A
[1]$\lambda_{\text{core-degree}}$-GPNN*	91.02 ±7.1	66.20 ±11.2	**77.18** ±4.6	75.60 ±2.7	88.60 ±4.6	**82.88** ±4.6	78.12 ±1.91	**76.13** ±0.68
[1]$\lambda_{\text{core-degree}}$-GPNN$^\diamond$	**92.60** ±4.8	65.95 ±8.5	76.82 ±3.9	74.40 ±2.4	**89.12** ±2.3	**83.09** ±3.1	77.63 ±1.80	**75.89** ±0.30
[2]$\lambda_{\text{core-degree}}$-GPNN*	**97.89** ±2.6	**78.17** ±6.2	**81.40** ±3.5	**80.10** ±3.1	**94.05** ±2.6	**89.09** ±3.3	78.12 ±1.91	76.13 ±0.68
[2]$\lambda_{\text{core-degree}}$-GPNN$^\diamond$	**97.37** ±4.9	**79.61** ±7.3	**85.53** ±6.2	78.10 ±3.1	**91.84** ±1.6	**89.72** ±2.3	77.63 ±1.80	75.89 ±0.30

Table 3.4: Graph classification performance is reported as accuracy (%) for TU datasets and ROC (%) for OGB datasets. For the TU datasets, settings 1 and 2 correspond to the configurations used in Xu et al. (2019) and Feng et al. (2022), respectively. For OGB datasets, we employ the setup introduced by Hu et al. (2020). The best results for each setting are highlighted in **blue** for setting 1 and **black** for setting 2. Baseline results are sourced from Feng et al. (2022), Wijesinghe and Wang (2022), and Zhao et al. (2022a).

3.7.2 Graph Regression

We provide experiments related to graph regression task. We employ a 12K subset of the ZINC (250K) dataset (Irwin et al., 2012) with and without edge features. The results are summarized in Table 3.5.

Methods	ZINC	
	Without Edge Features	With Edge Features
GCN	0.459 ±0.006	0.321 ±0.009
PPGN	0.407 ±0.028	N/A
GIN	0.407 ±0.028	0.163 ±0.004
DGN	0.219 ±0.010	0.168 ±0.003
Deep LRP#	0.223 ±0.008	N/A
CIN#	**0.115** ±0.003	**0.079** ±0.006
$\lambda_{\text{core-degree}}$-GPNN*	**0.214** ±0.018	0.148 ±0.015
$\lambda_{\text{core-degree}}$-GPNN$^{\diamond}$	0.222 ±0.021	**0.141** ±0.011

Table 3.5: Graph regression MAE for ZINC 12K dataset.The best results are highlighted in **black** and the second best results are highlighted in **blue**. For the baseline results, we refer to Bodnar et al. (2021). The methods marked by # are based on pre-selected domain-specific structural information.

From the results in Table 3.5, we can see that both GPNN variants show competitive performance in the regression task. Different from models like CIN, GPNN does not process any hand-crafted domain-specific structural information that is known to be highly correlated to molecular prediction tasks. Therefore, our model performance is less than such models.

3.7.3 Node Classification

Methods	Cora	CiteSeer	Wisconsin	Texas	Cornell
GIN	80.6 ± 0.4	73.8 ± 0.4	70.5 ± 1.6	61.6 ± 1.1	28.9 ± 2.6
GPNN$^{\star}{}_{GIN}$	82.0 ± 1.4	**74.8 ± 0.3**	71.3 ± 1.7	**62.3 ± 1.9**	**69.4 ± 13.3**
GPNN$^{\diamond}{}_{GIN}$	**82.3 ± 0.9**	74.2 ± 0.4	**71.4 ± 1.5**	61.5 ± 1.5	67.2 ± 13.9
GCN	82.6 ± 1.5	78.1 ± 0.2	55.8 ± 19.5	53.1 ± 21.6	54.7 ± 21.9
GPNN$^{\star}{}_{GCN}$	**83.2 ± 1.5**	**78.8 ± 0.3**	**56.5 ± 17.7**	51.8 ± 23.0	**61.1 ± 18.9**
GPNN$^{\diamond}{}_{GCN}$	82.8 ± 1.7	78.6 ± 0.3	56.1 ± 18.2	**57.7 ± 22.5**	55.1 ± 21.3
GAT	83.9 ± 0.8	77.7 ± 0.6	58.6 ± 1.6	62.0 ± 2.2	64.5 ± 1.4
GPNN$^{\star}{}_{GAT}$	84.2 ± 0.8	**78.1 ± 0.6**	**60.3 ± 2.5**	64.3 ± 3.1	**66.0 ± 1.6**
GPNN$^{\diamond}{}_{GAT}$	**84.5 ± 0.4**	77.8 ± 0.8	59.9 ± 2.5	**64.4 ± 3.4**	65.3 ± 1.4
GraphSAGE	84.5 ± 0.3	77.9 ± 0.5	87.0 ± 1.7	81.6 ± 2.3	80.9 ± 1.9
GPNN$^{\star}{}_{GraphSAGE}$	**84.9 ± 0.3**	**78.2 ± 0.6**	**87.4 ± 1.2**	79.5 ± 2.0	82.6 ± 1.9
GPNN$^{\diamond}{}_{GraphSAGE}$	84.7 ± 0.5	78.1 ± 0.5	87.1 ± 1.6	**82.0 ± 1.3**	**84.7 ± 1.9**

Table 3.6: node Classification Accuracy (%) averaged over 10 random splits. The best results are highlighted in **black**.

Table 3.6 presents the results on node classification, demonstrating that GPNNs consistently enhance the performance of all standards methods across benchmark

datasets, including both homophilic and heterophilic ones. This enhancement is attributed to GPNNs' ability to incorporate structural interactions as additional features into representations, which standard GNNs often overlook. This allows GPNNs to capture nuanced relationships between nodes, leading to enhanced performance compared to methods that rely solely on inter-interactions between nodes.

3.7.4 Impact of Different Graph Partitioning Schemes

Choosing a graph partitioning scheme is one of the key design decisions of GPNN. We thus experiment to understand how a graph partitioning scheme may affect the learning of structural interactions in GPNN. Five graph partitioning schemes are considered in this experiment: Φ_{core}, $\Phi_{\text{core-degree}}$, $\Phi_{\text{core-onion}}$, Φ_{degree}, and Φ_{triangle}. We use three molecular datasets from Open Graph Benchmark (OGB) (Hu et al., 2020) and three datasets from TU dataset benchmark (Morris et al., 2020a). The results are shown in Table 3.7. The corresponding statistics of structural interactions by these graph partitioning schemes are provided in the appendix Table 3.8, and Figures 3.4, and 3.5.

From the results, we can see that $\Phi_{\text{core-degree}}$ and Φ_{degree} schemes consistently perform better than the other schemes. This is primarily due to two reasons. Firstly, these two schemes generate a considerable amount of structural interactions from intra-edges which helps GPNN to learn the structural behavior of a graph. Secondly, the number of graph partitions generated by these two schemes is sufficient but not excessive, which thereby does not add much computational burden to the training process. For $\Phi_{\text{core-onion}}$, it generates the highest number of intra-interactions and partitions as shown in the interaction statistics, and the training complexities induced by the large number of learning parameters lead to its subpar performance. Although the number of intra-edges generated by Φ_{triangle} on these datasets is small, it can distinguish some graphs that cannot be distinguished by 1-WL. Thus, $\lambda_{\text{triangle}}$-GPNN* outperforms GIN on all datasets.

Methods	DHFR	ENZYMES	PTC_FM	ogbg-moltoxcast	ogbg-moltox21	ogbg-molhiv
GIN	80.04 ± 4.9	34.17 ± 4.2	70.50 ± 4.7	63.41 ± 0.7	74.91 ± 0.5	75.58 ± 1.4
λ_{core}-GPNN*	80.43 ± ±3.1	35.33 ± ±2.9	**74.48 ±** ±4.2	**63.88** ±0.8	**75.34** ±0.4	**76.52** ±1.1
λ_{core}-GPNN$^\diamond$	**80.56 ±** ±2.6	**44.67 ±** ±5.6	72.48 ± ±6.4	63.59 ±0.6	75.28 ±0.5	75.91 ±1.2
$\lambda_{\text{core-degree}}$-GPNN*	82.01 ± ±2.3	38.50 ± ±3.8	73.92 ± ±6.0	**64.70 ±** ±0.4	**76.13 ±** ±0.7	**78.12 ±** ±1.9
$\lambda_{\text{core-degree}}$-GPNN$^\diamond$	**82.15 ±** ±2.9	**39.83 ±** ±4.1	**73.92 ±** ±4.9	64.48 ± ±0.5	75.98 ± ±0.4	77.70 ± ±2.2
$\lambda_{\text{core-onion}}$-GPNN*	81.61 ± ±2.4	36.50 ± ±3.2	72.79 ± ±6.4	**62.91** ±0.8	75.60 ±0.5	76.57 ±1.1
$\lambda_{\text{core-onion}}$-GPNN$^\diamond$	**82.14 ±** ±3.2	**36.83 ±** ±3.2	**73.36 ±** ±4.9	61.86 ±0.4	**76.07** ±0.5	**77.03** ±1.3
λ_{degree}-GPNN*	**82.41 ±** ±3.7	35.50 ± ±4.7	73.05 ± ±3.5	**65.28 ±** ±0.5	75.15 ± ±0.7	**78.98 ±** ±1.5
λ_{degree}-GPNN$^\diamond$	**82.41 ±** ±2.9	**43.33 ±** ±4.8	**73.19 ±** ±3.7	64.81 ±0.7	**75.89** ±0.3	77.63 ±1.8
$\lambda_{\text{triangle}}$-GPNN*	79.37 ± ±2.5	**41.33 ±** ±3.0	70.48 ± ±4.5	**63.99 ±** ±0.4	**75.20 ±** ±0.9	**76.42 ±** ±1.2
$\lambda_{\text{triangle}}$-GPNN$^\diamond$	**80.96 ±** ±3.1	40.16 ± ±3.5	**71.92 ±** ±4.6	63.41 ±0.7	74.99 ±0.6	75.92 ±1.2

Table 3.7: Ablation study accuracy (%) ± standard deviation (%). Highest results in each partitioning scheme are highlighted in **black**.

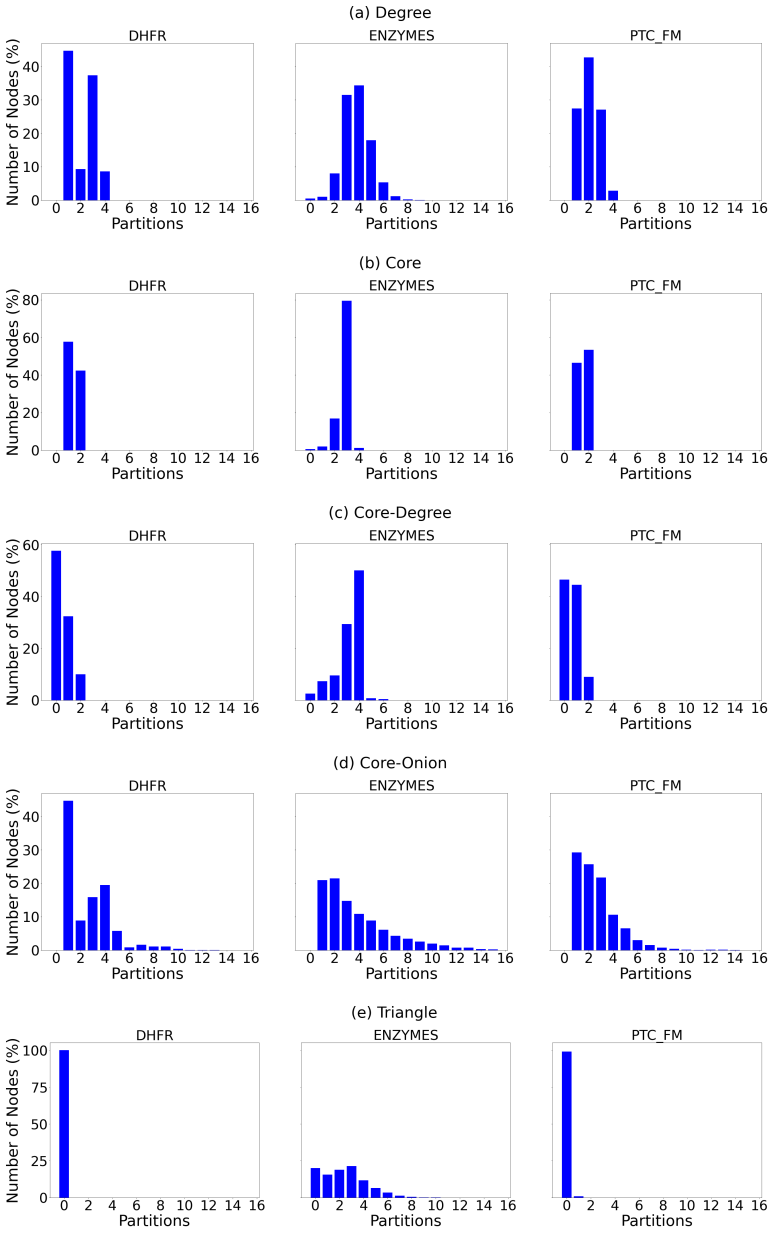


Figure 3.4: Node distributions concerning partitions under five graph partitioning schemes for TU Datasets.

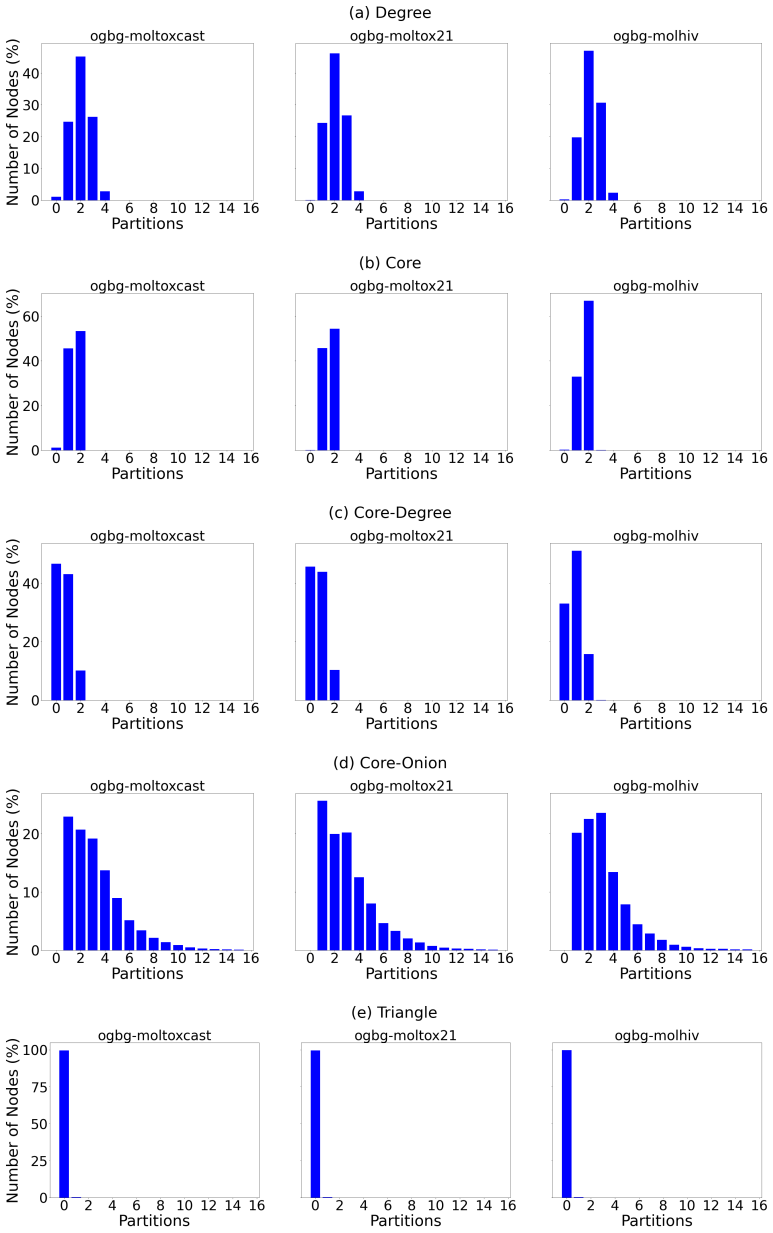


Figure 3.5: Node distributions concerning partitions under five graph partitioning schemes for OGB Datasets.

Dataset	Partitioning Scheme	# Partitions	Avg # Intra-Edges	Avg # Inter-Edges	# Intra-Edges / # Inter-Edges
DHFR	Φ_{core}	3	12.38	32.16	28 / 72
	$\Phi_{\text{core-degree}}$	5	21.79	22.75	49 / 51
	$\Phi_{\text{core-onion}}$	27	33.88	10.66	76 / 24
	Φ_{degree}	7	34.72	9.82	78 / 22
	Φ_{triangle}	1	0.00	44.54	0 / 100
ENZYMES	Φ_{core}	5	2.59	59.55	4 / 96
	$\Phi_{\text{core-degree}}$	7	23.33	38.81	38 / 62
	$\Phi_{\text{core-onion}}$	22	44.78	17.36	72 / 28
	Φ_{degree}	10	42.44	19.70	68 / 32
	Φ_{triangle}	14	41.80	20.34	67 / 33
PTC_FM	Φ_{core}	3	2.14	12.34	15 / 85
	$\Phi_{\text{core-degree}}$	5	4.70	9.78	32 / 68
	$\Phi_{\text{core-onion}}$	15	8.87	5.61	61 / 39
	Φ_{degree}	5	9.55	4.93	66 / 34
	Φ_{triangle}	2	0.05	14.43	1 / 99
ogbg-moltoxcast	Φ_{core}	3	5.08	33.44	13 / 87
	$\Phi_{\text{core-degree}}$	5	13.05	25.47	34 / 66
	$\Phi_{\text{core-onion}}$	42	24.53	13.99	64 / 36
	Φ_{degree}	7	24.56	13.96	64 / 36
	Φ_{triangle}	2	0.09	38.43	1 / 99
ogbg-moltox21	Φ_{core}	3	5.12	33.47	13 / 87
	$\Phi_{\text{core-degree}}$	5	13.18	25.41	34 / 66
	$\Phi_{\text{core-onion}}$	42	24.54	14.05	64 / 36
	Φ_{degree}	7	24.50	14.08	63 / 37
	Φ_{triangle}	2	0.09	38.50	1 / 99
ogbg-molhiv	Φ_{core}	4	7.09	47.84	13 / 87
	$\Phi_{\text{core-degree}}$	8	23.30	31.64	42 / 58
	$\Phi_{\text{core-onion}}$	88	33.53	21.41	61 / 39
	Φ_{degree}	11	33.52	21.42	61 / 39
	Φ_{triangle}	11	0.13	54.80	1 / 99

Table 3.8: Statistics for five graph partitioning schemes used in experiments.

3.7.5 Impact of Different Structural Interactions

Methods	MUTAG	PTC_MR	COX2	DHFR
GIN	92.80 ±5.9	65.60 ±6.5	88.87 ±2.3	80.04 ±4.9
$\lambda_{\perp}$-GPNN*	94.74 ±4.7	69.75 ±5.1	87.37 ±4.1	80.03 ±4.0
$\lambda_{\perp}$-GPNN$^{\diamond}$	**96.32** ±4.1	**71.77** ±4.3	89.37 ±2.2	**80.30** ±3.0
$\lambda_{\perp}$-GPNN†	95.76 ±3.2	71.43 ±4.7	**89.45** ±2.1	78.97 ±4.0
$\lambda_{\text{core-degree}}$-GPNN*	97.89 ±2.6	78.17 ±6.2	89.09 ±3.3	82.01 ±2.3
$\lambda_{\text{core-degree}}$-GPNN$^{\diamond}$	**97.37** ±4.9	79.61 ±7.3	**89.72** ±2.3	**82.15** ±2.9
$\lambda_{\text{core-degree}}$-GPNN†	96.29 ±3.4	**80.05** ±7.1	89.21 ±2.8	80.16 ±2.6

Table 3.9: Comparison of GPNN variants in graph classification accuracy (%) with different interaction types.

We examine how trivial and non-trivial partitioning schemes, as well as various types of interactions, impact GPNNs' performance. For the partitioning scheme, we use $\Phi_{\text{core-degree}}$, which balances interaction exploration with computational efficiency. We use four datasets from TU dataset benchmark (Morris et al., 2020a) following the evaluation setup introduced by Feng et al. (2022). Results are reported in Table 3.9. From the results, we can see that $\lambda_{\text{core-degree}}$-GPNN$^{\delta}$ consistently outperforms the corresponding $\lambda_{\perp}$-GPNN$^{\delta}$. All GPNNs show considerable improvements over GIN while $\lambda_{\perp}$-GPNN* has the closest results, empirically validating Theorem 1.

Although GPNN† theoretically possess higher expressive power, compared GPNN* and GPNN$^{\diamond}$, it does not always perform well empirically We believe this is due to increased computational burden, as large numbers of model parameters that negatively affect model training and optimization.

3.7.6 Impact of Individual GPNN Components

To identify which components of GPNN contribute most to its performance, we conduct a systematic ablation study using two TU benchmark (Morris et al., 2020a) datasets where we remove one component at a time while keeping all other elements fixed. We evaluate five variants of our model. The *Full GPNN*$^{\diamond}$ includes all components as described in Equations 1-3. In the *w/o node Embedding* variant, we remove Equation 1, relying solely on interaction embeddings and the combined neighborhood representation. The *w/o Interaction Embedding* variant excludes Equation 2, eliminating the explicit modeling of edge interactions between nodes. In the *w/o Colored Neighborhood* variant, we ignore partition assignments when aggregating neighbor information, treating all neighbors uniformly regardless of which partition they belong to. Finally, the *w/o Partition Scheme* variant uses trivial partitioning ($\lambda_{\perp}$), where all nodes receive the same initial color, effectively removing structural partitioning from the model.

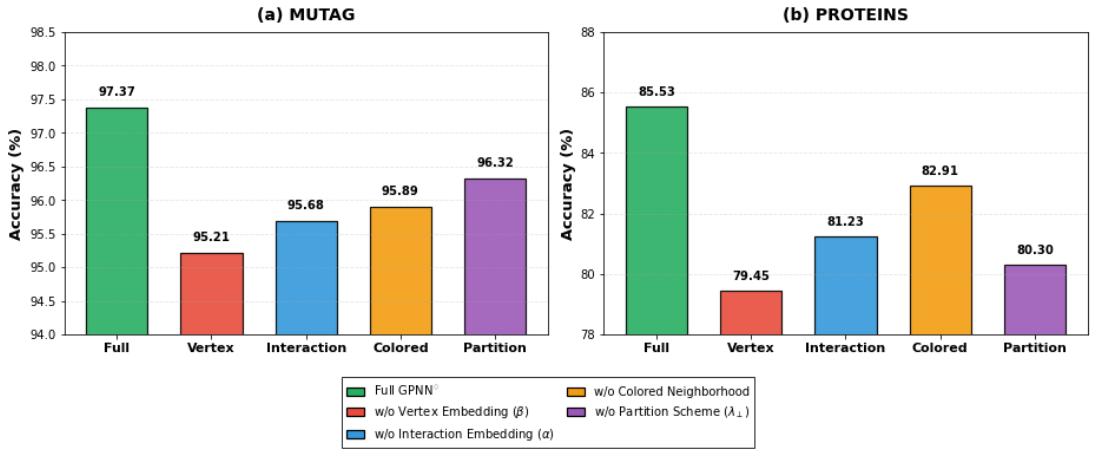


Figure 3.6: Component-wise ablation study on MUTAG and PROTEINS datasets. Each bar represents model performance when a specific component is removed.

Figure 3.6 shows that all components contribute to performance, with node embeddings being the most critical, followed by interaction embeddings. Removing either causes substantial accuracy drops, confirming their complementary roles in representation learning. The colored neighborhood aggregation and structural partitioning scheme each provide moderate but consistent improvements, demonstrating that partition-aware message passing enhances the model's ability to capture structural patterns. Overall, GPNN's effectiveness stems from the synergistic integration of all components rather than any single element.

3.8 Summary

In this chapter, we introduced the notion of permutation-invariant graph partitioning to explore complex interactions among structural components. Building on this, we developed a novel GNN architecture to integrate structural interactions into graph representation learning. Our theoretical analysis shows how these interactions can enhance the expressive power of GNN models, establishing strong connections to the 1-WL and 3-WL tests, as well as their complexities. Empirical results validate the effectiveness of the proposed model.

Chapter 4

Depth-Adaptive Graph Neural Networks via Learnable Bakry-Émery Curvature

4.1 Overview

In recent times, researchers have started leveraging geometric tools, such as curvature (Bochner, 1946), to enhance the representational power of GNNs (Topping et al., 2022; Ye et al., 2019). Originally rooted in differential geometry, curvature measures how a space deviates from flatness by tracking the convergence or divergence of geodesics (Bochner, 1946; Ollivier, 2007). When applied to graphs, it captures the connectivity and relational properties of nodes and edges (Lin et al., 2011; Ni et al., 2015), offering a more nuanced structural analysis. Incorporating curvature into GNNs can not only deepen our understanding of local connectivity but also provide insights into information flow, helping mitigate issues like oversoothing and oversquashing (Sun et al., 2023b; Fesser and Weber, 2024). Among the various forms of curvature, Ricci curvature has gained prominence due to its formulation on discrete spaces, which allows for efficient computation. This efficiency makes it well-suited for analyzing the structural properties of graphs (Forman et al., 1999; Ollivier, 2007; Samal et al., 2018; Fesser and Weber, 2024).

Despite their promise, current approaches compute curvature as a pre-processing step for GNNs, limiting its integration to decoupled techniques, such as graph rewiring and sampling (Topping et al., 2022; Fesser and Weber, 2024; Liu et al., 2023; Nguyen et al., 2023; Ye et al., 2019; Li et al., 2022a), which do not interact dynamically with the learning process. Consequently, these methods often overlook the interplay between node features, structured data, and the evolving dynamics of information diffusion that directly impact downstream tasks. However, integrating curvature dynamically during training presents significant challenges. Traditional curvature measures rely on the static, discrete graph structure, limiting their flexibility in learning contexts. Moreover, dynamic integration requires efficient curvature computation at every training step and the ability to adapt to the continuously changing training process.

In this chapter, we address these challenges by leveraging Bakry-Émery curvature (Cushing et al., 2020; Mondal et al., 2024; Pouryahya et al., 2016). Unlike traditional curvature measures, Bakry-Émery curvature captures intrinsic graph structures while accounting for diffusion dynamics, node function behavior, and gradient flows. Essentially, it extends Ricci curvature by incorporating a broader range of graph features, offering a more nuanced view of structural properties and information propagation (Wei and Wylie, 2009). Notably, although Bakry-Émery curvature is computed locally, its lower bounds provide theoretical guarantees for global properties such as Markov chain convergence rates and functional inequalities (Weber and Zacher, 2021). These features make it particularly well-suited for GNN architectures, where understanding the interplay between structure and diffusion can lead to more effective learning strategies. Among the alternatives, both Ollivier-Ricci and Forman-Ricci curvature are purely structural measures defined on edges or node degrees, and do not account for diffusion dynamics or feature behaviour on the graph. This makes them unsuitable for capturing the interplay between graph structure and information propagation.

However, integrating Bakry-Émery curvature into the GNN architecture is nontrivial. One key difficulty arises from its definition (Ambrosio et al., 2015), which requires evaluating curvature over the entire function space, a computationally intractable task for large, complex graphs. To overcome this, we propose a learnable strategy that selects a task-specific subset of functions, enabling efficient curvature estimation by focusing on the most relevant structural and diffusion characteristics. Moreover, we introduce an adaptive mechanism that leverages the estimated node curvature to dynamically adjust the depth of message-passing layers for each node. This approach not only enhances the feature distinctiveness of GNN architectures but also captures finer structural variations within the graph.

To summarize, our main contributions are as follows:

- **A New Perspective on Curvature in GNNs:** We propose Bakry-Émery curvature as a geometric tool for measuring node curvature in GNNs, capturing both structural properties and diffusion dynamics of graphs.
- **Efficient Curvature Approximation:** We develop a learnable strategy that selects a task-specific subset of functions, enabling efficient estimation of Bakry-Émery curvature on large-scale graphs.
- **Depth-Adaptive Mechanism for GNNs:** We introduce an adaptive mechanism that uses estimated Bakry-Émery curvature to dynamically determine the message-passing depth for each node, thereby enhancing the feature distinctiveness of existing GNN architectures.
- **Theoretical Insights:** We theoretically link Bakry-Émery curvature to feature distinctiveness in GNNs, showing that high-curvature nodes (fast diffusion, short mixing times) need fewer layers for distinct representations, while low-curvature nodes (slow diffusion, long mixing times) benefit from deeper architectures.

- **Enhanced GNN Performance:** Extensive experiments show that integrating our depth-adaptive mechanism consistently improves the performance of existing GNN architectures across a range of downstream tasks.

4.2 Bakry-Émery Curvature

4.2.1 Preliminaries

Let $G = (V, E, w)$ be an undirected weighted graph, where V is the set of nodes, E is the set of edges, and $w : E \to \mathbb{R}_+$ is the edge weight function. For any node $v \in V$, we denote by $N(v)$ the set of nodes adjacent to v.

4.2.2 Formulation

Bakry–Émery curvature provides a way to capture the local geometric behavior of functions on a graph (Ambrosio et al., 2015). We begin by defining a few operators that are fundamental to this concept.

For a function $f : V \to \mathbb{R}$, its weighted Laplacian at v is defined as

$$\Delta f(v) = \sum_{u \in N(v)} w(v,u)(f(u) - f(v)). \tag{4.1}$$

This operator captures the weighted difference between $f(v)$ and the values of f at its neighbors.

To formalize the Bakry–Émery curvature, we introduce two operators that capture the local behavior of functions on the graph. The first operator, $\Gamma(f,f)(v)$, analogous to the squared gradient, quantifies the local variability of f at v relative to its neighbors:

$$\Gamma(f,f)(v) = \frac{1}{2}\left(\sum_{u \in N(v)} w(v,u)(f(u) - f(v))^2\right) \tag{4.2}$$

The second operator, $\Gamma_2(f,f)(v)$, quantifies the convexity of f at v as follows:

$$\Gamma_2(f,f)(v) = \frac{1}{2}\Delta\Gamma(f,f)(v) - \Gamma(f,\Delta f)(v) \tag{4.3}$$

which can be equivalently expanded as,

$$\begin{aligned}\Gamma_2(f,f)(v) =& \frac{1}{2}\Big(\sum_{u \in N(v)} w(v,u)\big(\Gamma(f,f)(u) - \Gamma(f,f)(v)\big)\Big) \\ &- \sum_{u \in N(v)} w(v,u)(f(u) - f(v))\big(\Delta f(u) - \Delta f(v)\big).\end{aligned} \tag{4.4}$$

The Bakry–Émery curvature $\kappa(v)$ at node v is defined as the largest constant such

that the curvature–dimension inequality

$$\Gamma_2(f,f)(v) \geq \kappa(v)\,\Gamma(f,f)(v). \tag{4.5}$$

holds for all functions $f : V \to \mathbb{R}$. Equivalently, $\kappa(v)$ represents the tightest lower bound on the ratio

$$\frac{\Gamma_2(f,f)(v)}{\Gamma(f,f)(v)}, \tag{4.6}$$

for all functions f with $\Gamma(f,f)(v) \neq 0$. This lower bound reflects the local geometric and functional properties of the graph at the node v.

4.3 Curvature and Information Propagation

In this section, we examine the influence of Bakry–Émery curvature on local information diffusion in a graph. In particular, we establish a relationship between curvature and the local mixing time, which quantifies the rate at which information equilibrates around a node.

Given a node $v \in V$ with Bakry–Émery curvature $\kappa(v)$, we define the local mixing time $\tau_v(\epsilon)$ for any tolerance $\epsilon > 0$ as

$$\tau_v(\epsilon) = \sup\left\{t > 0 : |\nabla f_t|^2(v) \geq \epsilon\,|\nabla f_0|^2(v) \text{ for all } f_0 \in \ell^2(V)\right\} \tag{4.7}$$

Here, $\ell^2(V)$ denotes the space of square-summable functions on V, $|\nabla f_0|^2(v)$ is bounded for all $v \in V$, and $f_t = e^{-tL} f_0$ describes the evolution of f_0 under the heat semigroup associated with the Laplacian L. The operator e^{-tL} is defined via the matrix exponential.

The local gradient at v is defined by

$$|\nabla f|(v) := \sqrt{\Gamma(f,f)(v)}. \tag{4.8}$$

Theorem 6 (Mixing Time Bound)**.** *Let $G = (V, E, w)$ be an undirected, weighted graph with bounded degree and Laplacian L. If the local curvature-dimension inequality holds for every $f \in \ell^2(V)$ at node v, then for any $\epsilon \in (0,1)$, the local mixing time satisfies*

$$\tau_v(\epsilon) \leq \frac{\log(1/\epsilon)}{2\kappa(v)}.$$

Proof. First, we explicitly define the operators (Lin and Yau, 2010):

$$\Gamma(f,g)(v) = \frac{1}{2}\left(L(fg) - fLg - gLf\right)(v)$$

$$\Gamma_2(f,f)(v) = \frac{1}{2}L\Gamma(f,f)(v) - \Gamma(f,Lf)(v)$$

Under the heat semigroup evolution $f_t = e^{-tL} f_0$ (Bailleul and Bernicot, 2016), for any

initial function $f_0 \in \ell^2(V)$, we have:

$$\frac{d}{dt}|\nabla f_t|^2(v) = -2\Gamma(f_t, Lf_t)(v)$$

Here, we differentiate the squared gradient with respect to time. The operator $\Gamma(f_t, Lf_t)(v)$ represents the interaction between the graph Laplacian and the gradient of the function at each point in time. Next, we apply the local Γ_2-criterion, and get the following bound:

$$\Gamma_2(f_t, f_t)(v) \geq \kappa(v)|\nabla f_t|^2(v)$$

This criterion tells us that the curvature $\kappa(v)$ at each node provides a lower bound for the Γ_2 operator of the function f_t in terms of the gradient squared. Using the Bochner formula for functions evolving under the heat equation (Wei and Wylie, 2007), and considering that the Bakry-Émery curvature provides a lower bound for Ricci curvature along with the maximum principle (Lin and Yau, 2010; Protter and Weinberger, 2012), we get the following inequality:

$$\frac{d}{dt}\left|\nabla f_t\right|^2(v) \leq -2\kappa(v)\left|\nabla f_t\right|^2(v)$$

Applying Grönwall's inequality (Evans, 2022) to above, we derive the following:

$$|\nabla f_t|^2(v) \leq e^{-2\kappa(v)t}|\nabla f_0|^2(v)$$

The exponential decay reflects the influence of the curvature $\kappa(v)$ on the gradient. To achieve $|\nabla f_t|^2(v) \geq \epsilon|\nabla f_0|^2(v)$ for any $\epsilon > 0$, we solve for t to get:

$$t \leq \frac{\log(1/\epsilon)}{2\kappa(v)}$$

The time t depends on the local curvature at each node. Finally, we conclude that the local mixing time $\tau_v(\epsilon)$, which represents the time required for the gradient to decay by a factor of ϵ, satisfies the inequality:

$$\tau_v(\epsilon) \leq \frac{\log(1/\epsilon)}{2\kappa(v)}$$

□

This theorem indicates that higher Bakry–Émery curvature at a node accelerates the local decay of gradients, thereby enabling faster information diffusion within its neighborhood. In essence, nodes with high curvature act as efficient hubs, quickly equilibrating information and promoting rapid propagation throughout the graph. Conversely, low-curvature nodes, which are less effective at mixing, lead to slower information spread. Since the mixing time is inversely proportional to curvature, enhancing curvature in the network can be seen as a mechanism for improving diffusion efficiency.

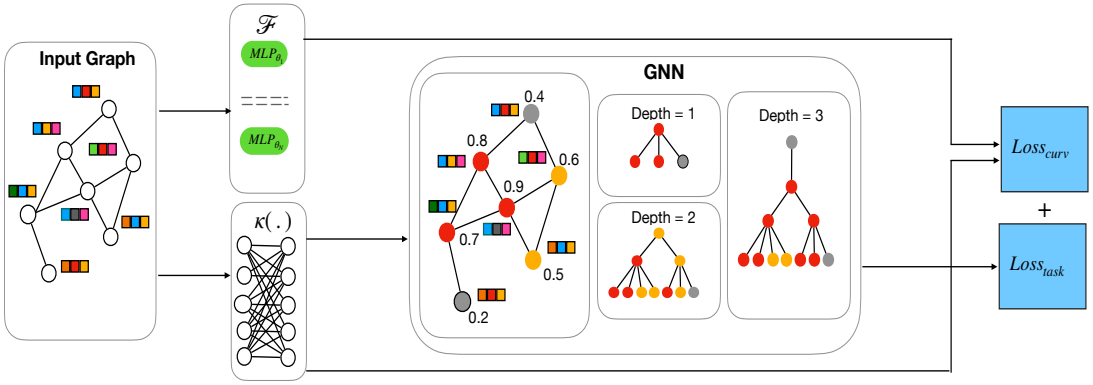


Figure 4.1: High-level overview of the proposed model: nodes are clustered based on learned curvature, where message-passing depth is adaptively adjusted according to curvature. Both the curvature functions and the GNN are jointly optimized via a loss function.

4.4 Depth-Adaptive Graph Neural Networks

We introduce a novel mechanism that leverages Bakry–Émery curvature to adaptively control the message-passing depth of graph neural networks (GNNs). In our framework, nodes with higher curvature (which typically indicate rapid local diffusion) terminate message passing earlier, while nodes with lower curvature continue aggregating messages to capture a broader neighborhood. The theoretical justification for this design choice is provided in Section 4.5.

4.4.1 Adaptive Layer Depth Mechansim

Let $t \in \mathbb{N}$ denote the iteration index in the message-passing process. To assign a stopping depth $T(v) \in \mathbb{N}$ to each node v, we rank the nodes based on their Bakry–Émery curvature $\kappa(v)$. For a given threshold $k\%$, we define

$$T(v) := \min \left\{ t \in \mathbb{N} \,\middle|\, \frac{1}{|V|} \sum_{w \in V} \mathbb{I}\Big(\kappa(w) \geq \kappa(v)\Big) \leq \frac{k\,t}{100} \right\}, \tag{4.9}$$

where $\mathbb{I}(\cdot)$ is the indicator function which is 1 if $\kappa(w) \geq \kappa(v)$ and 0 otherwise. This ensures that a node v is assigned a stopping depth $T(v) = t$ when the proportion of nodes with curvature at least $\kappa(v)$ falls below the threshold $\frac{k\,t}{100}$.

During message passing, for each node v and iteration $t \leq T(v)$, we update its embedding by aggregating messages from its neighbors. To incorporate the adaptive depth, the message from a neighbor u is taken from the layer $\gamma = \min\{t-1, T(u)\}$.

Formally, the update equations are:

$$m_v^{(t)} = \mathrm{AGG}\Big(\{\!\{h_u^{(\min\{t-1,\,T(u)\})} \mid u \in N(v)\}\!\}\Big), \tag{4.10}$$

$$h_v^{(t)} = \mathrm{UPD}\Big(h_v^{(t-1)},\, m_v^{(t)}\Big). \tag{4.11}$$

After $t = T(v)$ iterations, the final representation $h_v^{(T(v))}$ is fed into a task-specific module (e.g., a classifier or regressor) $\mathcal{C}$:

$$\hat{y}_v = \mathcal{C}\big(h_v^{(T(v))}\big) \tag{4.12}$$

where $\hat{y}_v$ represents the predicted output for node v.

This adaptive scheme is model-agnostic and can be integrated into existing GNN architectures such as GIN (Xu et al., 2019), GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017). The high-level architecture of our approach is depicted in Figure 4.1.

4.4.2 Learning to Estimate Curvature

Accurate estimation of $\kappa(v)$ is essential for determining the stopping depth $T(v)$. We cast curvature estimation as an optimization problem that enforces the Bakry–Émery curvature-dimension inequality. Let $\hat{\kappa}(v)$ denote the estimated curvature of node v. For any function $f : V \to \mathbb{R}$, we define the penalty

$$\mathcal{L}(\hat{\kappa}(v), f) \coloneqq \max\big\{0,\ \hat{\kappa}(v)\,\Gamma(f,f)(v) - \Gamma_2(f,f)(v)\big\} \tag{4.13}$$

This penalty quantifies the violation of the inequality

$$\Gamma_2(f,f)(v) \geq \hat{\kappa}(v)\,\Gamma(f,f)(v). \tag{4.14}$$

When the inequality holds at node v for f, we have $\mathcal{L}(\hat{\kappa}(v), f) = 0$; otherwise, the penalty is positive. Ideally, the true curvature $\kappa_{\text{true}}(v)$ is the largest value such that

$$\mathcal{L}(\kappa_{\text{true}}(v), f) = 0 \quad \text{for all } f : V \to \mathbb{R}. \tag{4.15}$$

Since optimizing over the entire function space is intractable, we restrict our search to a finite, parameterized set $\mathcal{F}$. In practice, we employ a Multi-Layer Perceptron (MLP) as a flexible function approximator. Let $f_\theta : V \to \mathbb{R}$ denote the function represented by an MLP with parameters θ. By sampling parameter configurations $\{\theta_i\}_{i=1}^{N}$, we obtain a candidate set

$$\mathcal{F} = \{f_{\theta_1}, f_{\theta_2}, \ldots, f_{\theta_N}\}. \tag{4.16}$$

We then approximate the estimated curvature by

$$\hat{\kappa}(v) = \sup_{\kappa \in \mathbb{R}} \inf_{f \in \mathcal{F}} \mathcal{L}(\kappa, f). \tag{4.17}$$

This optimization process seeks the maximum $\hat{\kappa}(v)$ such that the curvature-dimension inequality is "nearly" satisfied for all functions $f \in \mathcal{F}$.

We highlight two key properties of our curvature estimation approach:

1. **Upper bound:** Since $\mathcal{F}$ is a proper subset of the space of all functions $f : V \to \mathbb{R}$, the restricted optimization can only overestimate the true curvature. Formally, if

$$\kappa_{\text{true}}(v) = \sup\{\kappa \in \mathbb{R} : \mathcal{L}(\kappa, f) = 0 \text{ for all } f : V \to \mathbb{R}\}, \tag{4.18}$$

   then

$$\hat{\kappa}(v) = \sup_{\kappa \in \mathbb{R}} \inf_{f \in \mathcal{F}} \mathcal{L}(\kappa, f) \geq \kappa_{\text{true}}(v). \tag{4.19}$$

2. **Smoothness:** In order for the differential operators Γ and Γ_2 to be well-defined, the candidate functions in $\mathcal{F}$ must be smooth (Bakry and Émery, 2006). However, standard MLPs do not inherently guarantee smooth approximations, as many common activation functions (e.g., ReLU) lack smoothness. To ensure smooth outputs, we employ an MLP with C^∞ activation functions (e.g., the sigmoid function in our implementation), thereby guaranteeing that each function $f_\theta \in \mathcal{F}$ is infinitely differentiable.

This approach leverages the expressive power of neural networks to approximate a rich subset of functions, facilitating a robust, task-specific estimation of node curvature.

4.4.3 GNN Training Loss Function

We jointly optimize the GNN for the downstream task and for compliance with the curvature-dimension inequality by combining two loss terms: the task loss and the curvature loss.

Let $\mathcal{L}_{\text{task}}$ denote the loss for the downstream task (e.g., cross-entropy for classification or mean squared error for regression). To enforce the curvature-dimension inequality, we define the curvature loss as

$$\mathcal{L}_{\text{curv}} = \sum_{v \in V} \left(\sum_{f \in \mathcal{F}} \mathcal{L}\big(\hat{\kappa}(v), f\big) - \lambda\, \hat{\kappa}(v) \right). \tag{4.20}$$

Here, λ is a regularization constant controlling the tradeoff between maximizing $\hat{\kappa}(v)$ and ensuring the curvature-dimension inequality. The overall training loss is then defined as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{curv}}. \tag{4.21}$$

optimizing GNN performance while enforcing adherence to the graph's geometric structure via curvature.

4.5 Theoretical Analysis

To gain deeper insight into the computational and representational properties of our method, we first analyze its complexity, then explore the connection between feature distinctiveness and curvature.

4.5.1 Complexity Analysis

We first analyze the time and space complexity of our optimization process. For each node $v \in V$, let d_v denote its degree, and let $d_{\max}$ denote the maximum node degree in the graph. The computational cost of each operation depends on the local structure of the graph, particularly the degree distribution.

The time complexity of Equation (4.1), the weighted Laplacian $\Delta f(v)$, and Equation (4.2), the squared gradient operator $\Gamma(f,f)(v)$, is $O(d_v)$. In contrast, computing the convexity operator $\Gamma_2(f,f)(v)$ (Equation (4.3)) requires $O(d_v^2)$ operations per node. Thus, the overall computational cost over all nodes is $\mathcal{O}(\sum_{v\in V} d_v^2)$, which can be upper-bounded by $O(|V| \cdot d_{\max}^2)$. In our approach, the optimization is performed over a finite set $\mathcal{F}$ of candidate functions, with $|\mathcal{F}| = N$. Consequently, the total time complexity becomes

$$O\big(N \cdot |V| \cdot d_{\max}^2\big).$$

Since real-world graphs are typically sparse (i.e., $d_{\max} \ll |V|$) and N is small (in our experiments, $N \leq 5$), the overall time complexity is manageable. The space complexity is $\mathcal{O}(|V| + |E|)$, accounting for the storage of node and edge data.

4.5.2 Feature Distinctiveness and Curvature

We establish a connection between a node's feature distinctiveness in a GNN and its Bakry–Émery curvature. Here, feature distinctiveness is quantified by the variation in node features across layers. In particular, we show that the rate at which feature distinctiveness decays is governed by the Bakry–Émery curvature, reflecting how quickly information diffuses over the graph. This analysis assumes that node features evolve approximately according to heat flow driven by the graph Laplacian.

Theorem 7 (Feature Decay Bound)**.** *Let $G = (V, E, w)$ be an undirected, weighted graph with bounded degree. Consider a GNN with l layers, where each layer approximates the heat flow for a time step Δt. For a node $v \in V$ with Bakry-Émery curvature $\kappa(v)$, we define the feature distinctiveness $D(v, l)$ after l layers as*

$$D(v,l) = \frac{|\nabla f_l|^2(v)}{|\nabla f_0|^2(v)},$$

where f_l represents the node features at layer l. Then, the following holds:

$$D(v, l) \leq e^{-2\kappa(v) l \Delta t}.$$

For any $\epsilon > 0$, to maintain $D(v, l) \geq \epsilon$, it suffices that the number of layers satisfy:

$$l \leq \frac{\log(1/\epsilon)}{2\kappa(v)\Delta t}.$$

Proof. From the definition of $\tau_v(\epsilon)$ in Theorem 6, we know that:

$$|\nabla f_t|^2(v) \geq \epsilon |\nabla f_0|^2(v) \quad \text{for all} \quad t \leq \tau_v(\epsilon)$$

Using the upper bound for $\tau_v(\epsilon)$, we have:

$$\tau_v(\epsilon) \leq \frac{\log(1/\epsilon)}{2\kappa(v)}.$$

Now, to express this in the desired form, using Theorem 6, observe that the exponential decay bound for $|\nabla f_t|^2(v)$ is given by:

$$|\nabla f_t|^2(v) \leq e^{-2\kappa(v)t} |\nabla f_0|^2(v)$$

Each layer in the GNN approximates the heat flow for a time step Δt, so after l layers, the time is $t = l\Delta t$. Substituting this into the definition of $D(v, l)$, we get:

$$D(v, l) = \frac{|\nabla f_l|^2(v)}{|\nabla f_0|^2(v)} \leq e^{-2\kappa(v) l \Delta t}$$

For the second part, to maintain $D(v, l) \geq \epsilon$, we require:

$$e^{-2\kappa(v) l \Delta t} \geq \epsilon.$$

Taking the logarithm of both sides, we get $2\kappa(v) l \Delta t \leq \log(1/\epsilon)$. This is simplified to:

$$l \leq \frac{\log(1/\epsilon)}{2\kappa(v)\Delta t}.$$

□

In a nutshell, the higher curvature nodes, with stronger local connectivity, lose feature distinctiveness more quickly in a GNN due to faster information propagation. Therefore, fewer layers are required to maintain a given level of distinctiveness ϵ for high curvature nodes. In contrast, low curvature nodes have weaker local connections, causing feature distinctiveness to decay more slowly. To preserve ϵ for low curvature nodes, more layers are needed. Thus, the number of layers required to maintain feature distinctiveness tends to decrease with increasing node curvature.

4.6 Experimental Design

We conduct experiments on four widely used benchmark tasks: node classification, node regression, graph classification, and graph regression, utilizing a total of 21 datasets and comparing against 10 baselines to evaluate the effectiveness of our approach.

4.6.1 Datasets

For node classification, we use three widely adopted graphs exhibiting strong homophily: Cora, PubMed, and Citeseer (Sen et al., 2008; Namata et al., 2012), as well as five datasets with heterophily: Texas, Wisconsin, Actor, Squirrel, and Cornell (Rozemberczki et al., 2021; Tang et al., 2009). Additionally, we include a large-scale dataset from the Open Graph Benchmark, ogbn-arxiv (Hu et al., 2020). For node regression, we utilize four real-world datasets (Jazz, Network Science, Cora-ML, and Power Grid) (Rossi and Ahmed, 2015; McCallum et al., 2000). For graph classification, we use six small-scale real-world datasets from TU Datasets (MUTAG, PTC-MR, COX2, BZR, PROTEINS, and IMDB-BINARY) (Morris et al., 2020a), as well as a large-scale molecular dataset from the Open Graph Benchmark, ogbg-moltox21 (Hu et al., 2020). Finally, we employ the ZINC 12k dataset (Dwivedi et al., 2023) for graph regression.

Tables 4.1 and 4.2 present an overview of the statistics for the datasets used in our experiments. Specifically, Table 4.1 covers statistics for node classification and regression, while Table 4.2 provides dataset statistics related to graph classification and regression tasks.

Datasets	Task Type	# nodes	# Edges	# Classes
Jazz	node Regression	198	2,742	1
Network Science	node Regression	1,565	13,532	1
Cora-ML	node Regression	2,810	7,981	1
Power Grid	node Regression	4,941	6,594	1
Cora	node Classification	2,708	5,429	7
Citeseer	node Classification	3,327	4,732	6
Pubmed	node Classification	19,717	44,338	3
Actor	node Classification	7,600	33,544	5
Squirrel	node Classification	5,201	217,073	5
Wisconsin	node Classification	251	499	5
Cornell	node Classification	183	295	5
Texas	node Classification	183	309	5
ogbn-arxiv	node Classification	169,343	1,166,243	40

Table 4.1: Dataset statistics for node classification and regression.

Datasets	Task Type	# Graphs	Avg #nodes	Avg #Edges	# Classes
MUTAG	Graph Classification	188	17.93	19.79	2
PTC_MR	Graph Classification	344	14.29	14.69	2
BZR	Graph Classification	405	35.75	38.36	2
COX2	Graph Classification	467	41.22	43.45	2
IMDB-BINARY	Graph Classification	1,000	19.77	96.53	2
PROTEINS	Graph Classification	1,113	39.06	72.82	2
ogbg-moltox21	Graph Classification	7,831	18.6	19.3	2
ZINC	Graph Regression	12,000	23.1	49.8	1

Table 4.2: Dataset statistics for graph classification and regression.

4.6.2 Baselines

For node classification and regression baselines, we use three classical GNNs: GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017). For node classification, we further employ three enhanced GNNs: SGC (Wu et al., 2019), MixHop (Abu-El-Haija et al., 2019), and DIR-GNN (Rossi et al., 2024). For graph classification and regression baselines, we select two GNNs whose expressive power is known to be upper-bounded by the 1-dimensional Weisfeiler-Lehman (1-WL) algorithm (Zhang et al., 2024a): GIN (Xu et al., 2018a) and GCN (Kipf and Welling, 2017). Additionally, we include three GNNs designed to possess expressive power beyond 1-WL: ID-GNN (You et al., 2021), GraphSNN (Wijesinghe and Wang, 2022), and NC-GNN (Liu et al., 2024).

4.6.3 Evaluation settings

For node classification, we use the feature vectors and class labels with 10 random splits for all datasets except the OGB dataset. In these splits, 48% of the nodes are assigned for training, 32% for validation, and 20% for testing, following Pei et al. (2020a). For the OGB dataset, we adopt the setup described by Hu et al. (2020). For the node regression task, we focus on influence estimation (Xia et al., 2021; Ling et al., 2023), a key problem in network science, where the goal is to predict each node's influence based on graph structure and diffusion dynamics. We consider the Linear Threshold (LT) (Granovetter, 1978) and Independent Cascade (IC) (Goldenberg et al., 2001a) models, which capture different information spread mechanisms, with an initial activation set comprising 10% of nodes. Following Ling et al. (2023), we use their data splits and adopt 10-fold cross-validation, as in Errica et al. (2020). In graph classification, we adopt the setup outlined by Hu et al. (2020) for the OGB dataset. For the TU datasets, we follow the experimental setup provided by Feng et al. (2022), reporting the mean and standard deviation of the best accuracy across 10 test folds. For graph regression, we adhere to the setup provided by Dwivedi et al. (2023). We report baseline results from previous works using the same experimental setup, where available. If such results are unavailable, we generate baseline results by adopting the hyperparameter configurations specified in the original papers.

Since the benchmark graphs are unweighted and the Bakry-Émery curvature op-

erators (Equations (4.1), (4.2), and (4.3)) require edge weights, we employ learnable parameters for these weights. Additionally, the functions used to evaluate the Bakry-Émery curvature inequality take node features as inputs.

4.6.4 Model Hyper-Parameters

In our experiments, we search for hyper-parameters within the following ranges: the number of layers is selected from $\{2, 3, 4, 5\}$, k from $\{5, 10, 15, 20, 30, 40\}$, N from $\{1, 2, 3, 4, 5\}$, dropout from $\{0.0, 0.1, 0.2, 0.5, 0.6, 0.9\}$, learning rate from $\{0.005, 0.009, 0.01, 0.1\}$, batch size from $\{32, 64\}$, hidden units from $\{32, 64, 128, 256, 300\}$, weight decay from $\{5e-4, 9e-3, 1e-2, 1e-1\}$, and the number of epochs from $\{100, 200, 500, 1000\}$. The Adam optimizer (Kingma, 2015) is used for optimization. Further, we employ $\lambda = 1$ for all the experiments. For the ZINC and ogbg-moltox21 datasets, the learning rate decays by a factor of 0.5 after every 10 epochs.

4.7 Results, and Discussion

We use GNN+BEC to represent the GNN model incorporating our Bakry-Émery curvature-based adaptive layer depth mechanism, and $\Delta \uparrow$ indicates the performance improvement of our approach compared to the baseline.

4.7.1 Node Classification

Methods	Cora	Citeseer	Pubmed	Actor	Squirrel	Cornell	Wisconsin	Texas	ogbn-arxiv
GCN	87.28 ±1.26	76.68 ±1.64	87.38 ±0.66	30.26 ±0.79	36.89 ±1.34	57.03 ±4.67	59.80 ±6.99	59.46 ±5.25	71.74 ±0.29
GCN+BEC	88.50 ±1.32	80.43 ±0.88	88.55 ±0.68	31.66 ±0.58	37.82 ±0.60	61.62 ±4.64	69.41 ±2.85	75.68 ±2.18	71.91 ±0.17
$\Delta \uparrow$	+1.22	+3.75	+1.17	+1.40	+0.93	+4.59	+9.61	+16.22	+0.17
GAT	82.68 ±1.80	75.46 ±1.72	84.68 ±0.44	26.28 ±1.73	30.62 ±2.11	58.92 ±3.32	55.29 ±8.71	58.38 ±4.45	71.54 ±0.30
GAT+BEC	85.39 ±0.69	76.12 ±3.23	86.71 ±0.46	31.22 ±0.56	31.61 ±0.47	61.62 ±5.64	63.23 ±3.28	71.56 ±5.56	71.84 ±0.31
$\Delta \uparrow$	+2.71	+0.66	+2.03	+4.94	+0.99	+2.70	+7.94	+13.18	+0.30
GraphSAGE	86.90 ±1.04	76.04 ±1.30	88.45 ±0.50	34.23 ±0.99	41.61 ±0.74	75.95 ±5.01	81.18 ±5.56	82.43 ±6.14	71.49 ±0.27
GraphSAGE+BEC	87.08 ±0.42	78.01 ±1.33	89.00 ±0.31	35.56 ±0.87	44.53 ±0.81	81.08 ±3.82	88.97 ±1.88	89.41 ±3.06	71.83 ±0.14
$\Delta \uparrow$	+0.18	+1.97	+0.55	+1.33	+2.92	+5.13	+7.79	+6.98	+0.34
SGC	84.97 ±1.95	75.66 ±1.37	87.15 ±0.47	25.83 ±1.09	41.78 ±2.88	55.41 ±5.29	57.84 ±4.83	58.11 ±6.27	68.74 ±0.12
SGC+BEC	85.49 ±0.65	78.32 ±0.66	87.94 ±0.23	27.97 ±2.18	42.03 ±0.62	65.13 ±3.29	63.97 ±9.30	69.80 ±1.79	71.39 ±0.30
$\Delta \uparrow$	+0.52	+2.66	+0.79	+2.14	+0.25	+9.72	+6.13	+11.69	+2.65
MixHop	87.61 ±0.85	76.26 ±1.33	85.31 ±0.61	32.22 ±2.34	43.80 ±1.48	73.51 ±6.34	75.88 ±4.90	77.84 ±7.73	70.83 ±0.30
MixHop+BEC	87.89 ±0.67	76.79 ±0.95	88.18 ±0.37	36.91 ±0.93	43.97 ±0.67	78.37 ±2.70	93.82 ±1.10	88.23 ±1.75	72.04 ±0.26
$\Delta \uparrow$	+0.28	+0.53	+2.87	+4.69	+0.17	+4.86	+17.94	+10.39	+1.21
DIR-GNN	82.89 ±0.44	76.55 ±0.72	88.84 ±0.13	30.12 ±0.65	42.50 ±0.80	80.00 ±3.60	79.60 ±3.80	81.40 ±2.40	70.72 ±0.26
DIR-GNN+BEC	84.12 ±0.76	77.67 ±1.08	89.40 ±0.19	32.46 ±0.43	46.19 ±1.26	85.94 ±3.78	89.55 ±1.22	87.84 ±3.01	71.55 ±0.26
$\Delta \uparrow$	+1.23	+1.12	+0.56	+2.34	+3.69	+5.94	+9.95	+6.44	+0.83

Table 4.3: Node classification accuracy ± std (%). The baseline results are sourced from Zhu et al. (2020); Sun et al. (2024); Brody et al. (2022); Han et al. (2023b).

Table 4.3 presents the results for the node classification task. Our approach consistently outperforms the baselines in both homophilic and heterophilic settings. Notably, our adaptive layer mechanism achieves greater performance improvements on

heterophilic graphs compared to homophilic ones. We attribute this to the ability of our method to learn data-driven curvature values that leverage heterophilic label patterns, effectively capturing the complex and diverse relationships between nodes with dissimilar labels, which leads to enhanced aggregation and feature extraction. Additionally, our approach improves the performance of GNNs designed with inductive biases for heterophilic graph settings, such as MixHop, and DIR-GNN. This demonstrates that our method complements these models by capturing additional structural and label-dependent information that might otherwise be overlooked.

We also observe that our depth-adaptive mechanism significantly reduces the standard deviation of accuracy across baseline models, enhancing stability and consistency. This reduced variance ensures reliable and repeatable results, essential for real-world applications.

4.7.2 Node Regression

Table 4.4 presents the results for the influence estimation (Xia et al., 2021) task, formulated as a node regression problem. The results indicate that our approach significantly enhances baseline performance in predicting influence by effectively adapting to various underlying diffusion processes. By dynamically adjusting curvature based on the propagation mechanisms of different diffusion models, our method enhances generalization of the baseline GNN models across diverse network conditions.

Methods	Jazz		Cora-ML		Network Science		Power Grid	
	IC	LT	IC	LT	IC	LT	IC	LT
GCN	0.233 ±0.010	0.199 ±0.006	0.277 ±0.007	0.255 ±0.008	0.270 ±0.019	0.190 ±0.012	0.313 ±0.024	0.335 ±0.023
GCN+BEC	0.202 ±0.007	0.124 ±0.002	0.258 ±0.006	0.194 ±0.010	0.233 ±0.010	0.123 ±0.026	0.329 ±0.019	0.298 ±0.039
Δ ↑	+0.031	+0.075	+0.019	+0.061	+0.037	+0.067	-0.016	+0.037
GAT	0.342 ±0.005	0.156 ±0.100	0.352 ±0.004	0.192 ±0.010	0.274 ±0.002	0.114 ±0.008	0.331 ±0.002	0.280 ±0.015
GAT+BEC	0.238 ±0.035	0.123 ±0.002	0.269 ±0.014	0.184 ±0.015	0.233 ±0.016	0.126 ±0.010	0.291 ±0.016	0.277 ±0.013
Δ ↑	+0.104	+0.033	+0.083	+0.008	+0.041	-0.012	+0.040	+0.003
GraphSAGE	0.201 ±0.028	0.120 ±0.004	0.255 ±0.010	0.203 ±0.019	0.241 ±0.010	0.112 ±0.005	0.313 ±0.024	0.341 ±0.018
GraphSAGE+BEC	0.190 ±0.015	0.060 ±0.010	0.246 ±0.008	0.176 ±0.008	0.231 ±0.010	0.073 ±0.012	0.302 ±0.028	0.205 ±0.013
Δ ↑	+0.011	+0.060	+0.009	+0.027	+0.010	+0.039	+0.011	+0.136

Table 4.4: Influence estimation performance MAE ± std under IC and LT diffusion models.

4.7.3 Graph Classification

We present graph classification results in Table 4.5. These results show that our adaptive layer mechanism achieves notable to moderate improvements over baseline methods across all datasets. Significant gains are observed with the GIN model, which is commonly regarded as the standard baseline for graph classification tasks. Moreover, our approach enhances GNNs with higher expressive power than 1-WL by improving feature distinctiveness, thus offering a more effective mechanism for distinguishing between graph structures.

Methods	MUTAG	PTC-MR	IMDB-BINARY	COX2	BZR	PROTEINS	ogbg-moltox21	ZINC
GIN	92.8 ±5.9	65.6 ±6.5	78.1 ±3.5	88.9 ±2.3	91.1 ±3.4	78.8 ±4.1	74.9 ±0.5	0.387 ±0.015
GIN+BEC	96.1 ±3.6	72.9 ±5.7	80.8 ±3.3	89.3 ±3.1	92.4 ±3.6	79.1 ±3.7	75.7 ±0.7	0.308 ±0.009
Δ ↑	+3.3	+7.3	+2.7	+0.4	+1.3	+0.3	+0.8	+0.079
GCN	92.2 ±4.4	68.8 ±6.2	79.8 ±2.3	88.5 ±3.8	92.6 ±4.8	78.8 ±3.9	75.3 ±0.7	0.459 ±0.006
GCN+BEC	95.0 ±3.0	70.6 ±5.1	79.9 ±3.4	89.3 ±2.8	93.6 ±3.8	79.6 ±4.4	75.5 ±0.5	0.424 ±0.049
Δ ↑	+2.8	+1.8	+0.1	+0.8	+1.0	+0.8	+0.2	+0.035
ID-GNN	97.8 ±3.7	74.4 ±4.2	79.3 ±2.9	87.8 ±3.1	93.3 ±4.8	78.1 ±3.9	74.7 ±0.5	0.366 ±0.014
ID-GNN+BEC	98.3 ±3.6	75.6 ±4.0	81.5 ±2.4	89.3 ±3.1	93.8 ±4.5	78.6 ±3.8	75.4 ±0.7	0.350 ±0.013
Δ ↑	+0.5	+0.8	+2.2	+1.5	+0.5	+0.5	+0.7	+0.016
GraphSNN	94.7 ±1.9	70.6 ±3.1	78.5 ±2.3	86.3 ±3.3	91.1 ±3.0	78.4 ±2.7	75.5 ±1.1	0.297 ±0.004
GraphSNN+BEC	96.1 ±2.5	72.9 ±7.4	79.4 ±2.7	88.9 ±2.0	92.1 ±4.9	79.2 ±3.9	75.3 ±0.7	0.265 ±0.005
Δ ↑	+1.4	+2.3	+0.9	+2.6	+1.0	+0.8	-0.2	+0.032
NC-GNN	92.8 ±5.0	71.8 ±6.2	78.4 ±4.0	88.4 ±3.3	92.6 ±4.3	78.4 ±3.1	75.1 ±0.4	0.448 ±0.095
NC-GNN+BEC	96.0 ±2.4	75.0 ±4.6	79.6 ±3.6	88.9 ±2.4	93.6 ±4.9	79.6 ±2.9	75.6 ±0.5	0.356 ±0.017
Δ ↑	+3.2	+3.2	+1.2	+0.5	+1.0	+1.2	+0.5	+0.092

Table 4.5: Graph classification accuracy ± std (%) for TU datasets, ROC ± std (%) for the OGB dataset, and graph regression MAE ± std for ZINC dataset, with baseline results sourced from Wijesinghe and Wang (2022), Feng et al. (2022), and Hu et al. (2020).

4.7.4 Graph Regression

The regression results for ZINC 12K dataset is depicted in Table 4.5. Similar to the graph classification, the proposed adaptive layer mechanism consistently improves performance over baseline methods. These results underscore the effectiveness of our approach in enhancing feature representation and regularization, thereby improving the overall performance of graph regression tasks. Although node-specific, our curvature provides insights into the global graph structure, enhancing performance in tasks that depend on holistic properties of graphs.

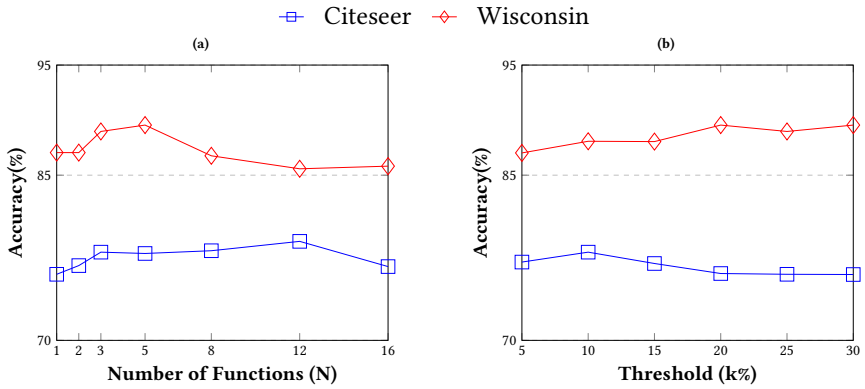


Figure 4.2: Model sensitivity to hyper-parameters

4.7.5 Hyper-Parameter Analysis

In this experiment, we analyze the model's sensitivity to its two primary hyper-parameters; number of curvature approximation functions, and the threshold k. Us-

ing GraphSAGE as the base model, we evaluate node classification performance on two datasets. The results are depicted in Figure 4.2.

Figure 4.2 (a) illustrates how performance varies with the number of functions used to approximate curvature. As expected, increasing the number of functions enhances performance by enabling the model to capture multiple aspects of information propagation. However, this improvement is only observed up to a certain threshold, beyond which the additional complexity introduced by an increased number of learnable parameters begins to outweigh the benefits, ultimately leading to performance degradation. Notably, the model achieves good performance with a relatively small number of functions, which is advantageous from a computational efficiency perspective. Figure 4.2 (b) illustrates how model performance varies with the threshold. For Citeseer, optimal performance is achieved at a lower threshold (5–10%), whereas Wisconsin benefits from a higher threshold. This difference stems from dataset characteristics: Citeseer, being homophilic, has adjacent nodes with the same labels, allowing deeper aggregation to enhance information mixing. In contrast, Wisconsin, a heterophilic dataset, benefits from shallow depth aggregation to prevent mixing information from differently labeled neighboring nodes.

4.7.6 Oversmoothing Analysis

We analyze the impact of our adaptive layer depth approach on oversmoothing in the node classification task by progressively increasing the layer depth up to 64 while measuring classification accuracy. Notably, we set $k = 1$ to allow the aggregation stopping mechanism to remain effective even at large depths. As shown in Figure 4.3, our method exhibits substantial robustness against oversmoothing, achieving a significant performance margin over baseline models across both datasets.

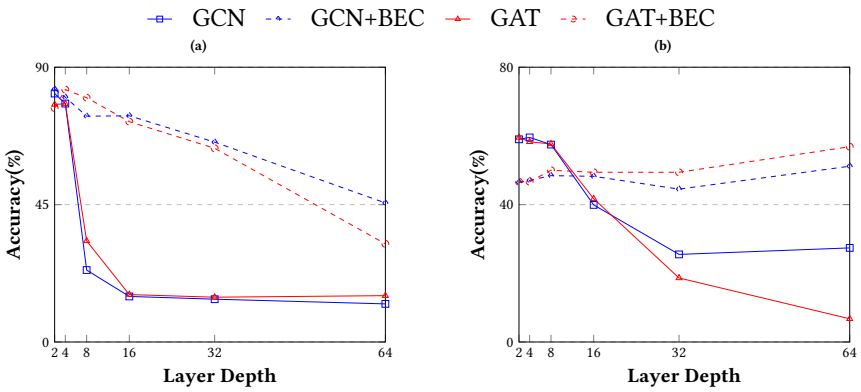


Figure 4.3: Oversmoothing comparison: (a) Cora; (b) Texas.

4.7.7 Comparison with Existing Curvature Notions

In this experiment, we integrate existing curvature notions by replacing our learnable curvature with these predefined curvatures in our adaptive layer depth approach

and evaluate their performance. We consider four curvatures from GNN literature: Ollivier-Ricci **(ORC)** (Southern et al., 2023; Topping et al., 2022), Forman Ricci **(FRC)** (Fesser and Weber, 2024), Jost and Liu **(JLC)** (Giraldo et al., 2023), and Effective Resistance **(ERC)** (Devriendt and Lambiotte, 2022) along with two structural descriptors, betweenness centrality **(BC)** and degree. These curvatures are precomputed before training and interpreted consistently: higher values indicate denser local structures, while lower values correspond to sparser regions. Note that for any edge-based curvature, node curvature is computed as the average of adjacent edge curvatures. The results are summarized in Table 4.6.

Methods	node Classification		node Regression	
	Citeseer	Cornell	Network Science	
			IC	LT
GCN	76.68 ±1.64	57.03 ±4.67	0.270 ±0.019	0.190 ±0.012
GCN+ORC	78.49 ±0.35	57.83 ±4.22	0.251 ±0.006	0.170 ±0.023
GCN+FRC	78.51 ±0.47	58.92 ±4.80	0.253 ±0.005	0.178 ±0.024
GCN+JLC	77.03 ±0.56	58.64 ±3.20	0.239 ±0.010	0.155 ±0.011
GCN+ERC	78.70 ±0.66	59.72 ±2.54	0.249 ±0.007	0.155 ±0.029
GCN+BC	78.10 ±0.26	45.68 ±3.51	0.248 ±0.005	0.147 ±0.013
GCN+Degree	77.87 ±0.45	44.59 ±2.76	0.243 ±0.006	0.159 ±0.019
GCN+BEC	**80.43** ±0.88	**61.62** ±4.64	**0.233** ±0.010	**0.123** ±0.026

Table 4.6: Ablation study on curvature notions. For node classification, accuracy ± std(%) is reported; for node regression, MAE ± std is used. The best results are highlighted in **bold**.

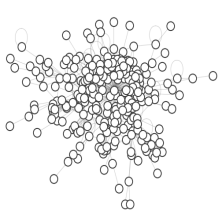
Our approach, which learns curvature in a data-driven manner, significantly outperforms all precomputed curvatures in both downstream tasks by capturing task-specific information.

4.7.8 Runtime and Parameter Complexity analysis

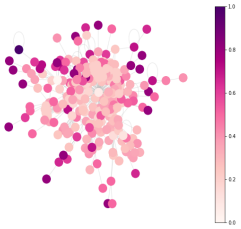
We evaluate scalability on large datasets, fixing the model depth at 4 and the hidden layer size at 256 for all methods. For our approach, we consider N in the set $\{1, 3, 5\}$. We adopt GCN and GIN as the base models for ogbn-arxiv and ogbg-moltox21, respectively. Each model is assessed on parameter count, average runtime (over 100 iterations), and accuracy/RoC, as shown in Table 4.7.

Methods	ogbn-arxiv			ogbg-moltox21		
	#Param (k)	Time (s)	Acc. (%)	#Param (k)	Time (s)	Roc. (%)
Baseline	176	0.30	70.87	1118	4.83	74.84
+BEC (N=1)	179	0.39	71.14	1123	6.90	75.22
+BEC (N=3)	184	0.52	71.76	1134	8.98	75.53
+BEC (N=5)	189	0.65	71.82	1144	10.39	75.76

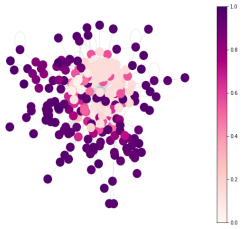
Table 4.7: Runtime and parameter complexity comparison for node and graph classification tasks.



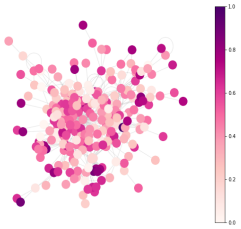
((a)) Original Graph



((b)) Ollivier Ricci Curvature



((c)) Forman Ricci Curvature



((d)) Bakry-Émery Curvature

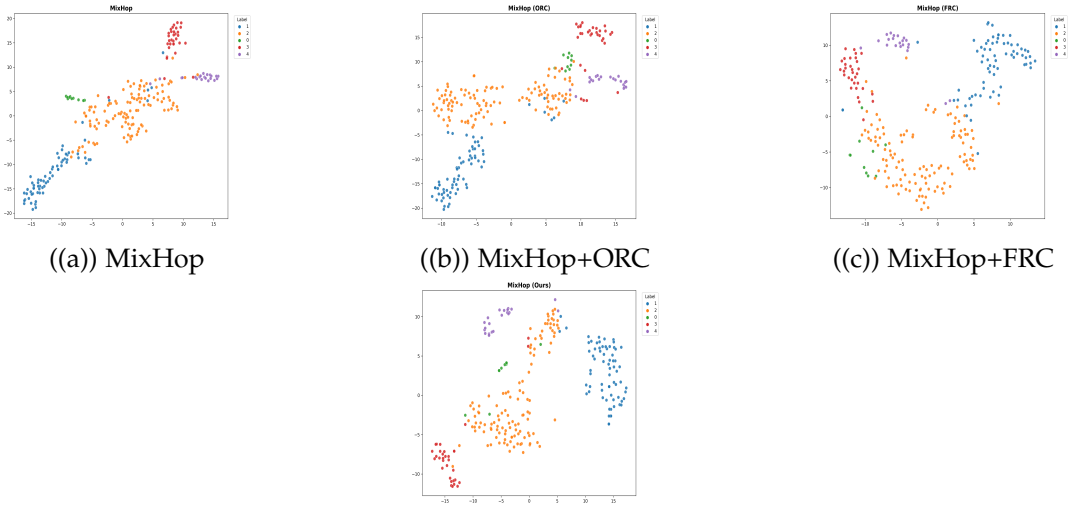
Figure 4.4: Curvature visualization

As N increases, parameter count and runtime grow moderately, ensuring computational feasibility. Our approach keeps parameter growth minimal by using MLPs to estimate curvature instead of expanding the GNN's parameter space. This design choice ensures that parameter learning of GNN remains efficient. With the same model capacity (i.e., fixed hidden layer size), our approach enhances representation power and outperforms the baseline, offering a meaningful improvement without excessive computational cost.

4.7.9 Visualization Analysis

We present a visualization analysis of node curvature for different curvature notions in Figure 4.4 using the Wisconsin dataset. In our approach, curvature is learned with MixHop as the base model. For the two Ricci curvature methods, node curvature is computed as the average of adjacent edge curvatures. All the node curvature values are normalized between 0 and 1 for visualization purposes. The comparison reveals that our method captures heterogeneous curvature patterns with greater granularity compared to other approaches. This level of detail is well justified, as Wisconsin is a heterophilic dataset where neighbouring nodes have diverse characteristics.

Figure 4.5 provides node embedding visualization using t-SNE (Van der Maaten and Hinton, 2008). MixHop is employed as the base model, and we evaluate its node classification performance on the Wisconsin dataset. The colors represent different class labels in the dataset. The comparison of embeddings shows significant improvements in our approach. The most notable enhancement is the formation of well-separated clusters. Our method improves cluster boundary definition, reducing overlap between class labels. Overall, our method captures both fine-grained rela-



((a)) MixHop ((b)) MixHop+ORC ((c)) MixHop+FRC

((d)) MixHop+BEC

Figure 4.5: 2D node embedding visualization

tionships and broader community structures in embedding, leading to better classification. This is well supported by the curvature visualization, as our approach effectively captures heterogeneous curvature patterns that other methods fail to detect.

4.8 Summary

In this chapter, we propose a novel mechanism for GNNs that learns node-wise curvature in a data-driven manner. Our framework builds on Bakry-Émery curvature, which theoretically captures geometric intricacies and information diffusion in graphs from a functional perspective. By establishing a theoretical link between feature distinction in GNNs and node curvature, we introduce an adaptive layer depth mechanism that leverages node curvature to enhance the representation capacity of GNNs. Integrated with existing GNN architectures, the proposed approach consistently improves performance across diverse downstream tasks.

Chapter 5

Beyond Fixed Depth: Adaptive Graph Neural Networks for Node Classification Under Varying Homophily

5.1 Overview

Traditional GNNs are built upon the homophily assumption, where connected nodes tend to share similar labels or characteristics (McPherson et al., 2001; Zhu et al., 2020). While this assumption has proven effective for many real-world networks, such as citation networks and knowledge graphs (Yang et al., 2016), it significantly limits GNNs' performance on heterophilic graphs, where connected nodes often have different labels or properties (Zheng et al., 2023a). Such heterophilic patterns are common in many real-world scenarios, including social networks and web graphs (Pei et al., 2020b; Zhu et al., 2021; Platonov et al., 2023).

To address this limitation, researchers have begun exploring specialized heterophily-aware GNN architectures that explicitly account for the dissimilarity between connected nodes and develop alternative message-passing strategies (Zhu et al., 2021; Zheng et al., 2022; Luan et al., 2022; Zhu et al., 2023a; Wang et al., 2024a). Nonetheless, certain limitations still exist. First, categorizing graphs as homophilic or heterophilic based on simple neighborhood label ratios oversimplifies the complex nature of information propagation in graphs. Real-world graphs are intricate structural entities with different regions or nodes exhibiting varying levels of homophily. Different nodes have varying sensitivity to information aggregation, depending on their structural context, features, and neighbor label composition. A more nuanced understanding of this phenomenon is required to design GNN architectures that accommodate these differences. Second, methods specifically designed for heterophilic graphs often lack the flexibility to effectively handle both homophilic and heterophilic contexts within the same architecture, limiting their applicability and generalizability across diverse graph types.

We aim to address these limitations through a fundamental observation: optimal

information propagation strategies are inherently different across nodes within the same graph. The uniform nature of information aggregation in current GNNs (i.e., fixed layer depth applied to all nodes) limits node classification performance, as different nodes have varying information aggregation needs based on their local neighbourhood properties, including degree distribution, label homophily ratio, and feature similarity patterns within their close neighbourhoods. This motivates us to theoretically connect structural and label characteristics to information propagation dynamics at the node level, gaining a more nuanced understanding of node-specific aggregation needs beyond global graph homophily characteristics.

Building on these theoretical insights, we derive node-specific metrics that quantify individual node aggregation requirements without being constrained by global graph homophily measures. We then utilize these insights to develop dynamic GNN architectures that adaptively determine aggregation depth for each node, improving their ability to handle node-specific information propagation needs. The main contributions of our work are summarized as follows:

- **Theoretical Foundation**: We establish theoretical connections between structural and label characteristics and information propagation dynamics, deriving node-specific metrics that quantify optimal aggregation depth requirements in GNNs.
- **Novel Adaptive Depth Architecture**: We propose a novel GNN architecture that dynamically adjusts aggregation depth for individual nodes based on their specific information propagation needs, with two distinct variants to accommodate different computational requirements.
- **Unified Homophilic/Heterophilic Handling**: Our architecture provides a single framework that seamlessly accommodates both homophilic and heterophilic graph patterns without requiring separate architectures or preprocessing steps.
- **Empirical Validation**: We provide comprehensive experiments demonstrating that our adaptive depth architecture consistently improves node classification performance of mainstream GNN backbones across diverse graph structures and homophily levels.

This chapter addresses the same core challenge as Chapter 4, namely determining the optimal aggregation depth per node, but approaches it from a complementary angle. Chapter 4 uses geometric curvature to guide depth allocation, where this chapter derives depth decisions from the label-theoretic properties of each node's local neighbourhood, specifically its homophily pattern and neighbourhood structural distribution. The two frameworks are independent.

5.2 Theoretical Analysis

We develop a theoretical framework to understand how neighborhood profile impacts node classification performance in GNNs under different homophily condi-

tions, establishing the basis for our adaptive depth allocation strategy. For node v, we define its profile as the tuple (d_v^+, d_v^-, d_v) describing the distribution of same-label neighbours, opposite-label neighbours, and total degree within its neighbourhood.

5.2.1 Preliminaries

We consider an undirected graph $G = (V, E, \mathbf{X}, \mathbf{y})$, where V represents the set of nodes, E is the set of edges, $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ is the matrix of initial node feature representations with a feature dimension of d, and $\mathbf{y} \in \{0,1\}^{|V|}$ is the vector of node labels. Each node $v \in V$ is associated with a feature vector $\mathbf{x}_v \in \mathbb{R}^d$ (i.e., $\mathbf{x}_v = \mathbf{X}_{v,:}$), and has a label $y_v \in \{0,1\}$ (i.e., $y_v = \mathbf{y}_v$). For theoretical tractability, we focus on the binary classification setting. Let $\mathcal{N}(v)$ be the neighborhood of node v, and $d_v = |\mathcal{N}(v)|$ its degree. We define same-label neighbors as $\mathcal{N}_v^+ = \{u \in \mathcal{N}(v) : y_u = y_v\}$ and opposite-label neighbors as $\mathcal{N}_v^- = \{u \in \mathcal{N}(v) : y_u \neq y_v\}$. Their cardinalities are $d_v^+ = |\mathcal{N}_v^+|$ and $d_v^- = |\mathcal{N}_v^-|$, with $d_v = d_v^+ + d_v^-$.

5.2.2 Class Concepts

To establish the assumptions for our analysis, we formally define the concepts of class prototypes, signal variance, and noise variance.

Definition 9 (Class Concepts)**.** *We define the following graph-wide parameters:*

- *Class prototypes:* $\boldsymbol{\mu}^0, \boldsymbol{\mu}^1 \in \mathbb{R}^d$ *where* $\boldsymbol{\mu}^c = \mathbb{E}[\mathbf{x}_u | y_u = c]$ *for* $c \in \{0,1\}$
- *Signal variance:* $\Delta^2 = \|\boldsymbol{\mu}^0 - \boldsymbol{\mu}^1\|^2$
- *Noise variance:*$\sigma_{intra}^2 = Var[\mathbf{x}_u | y_u = c]$ *for any* $c \in \{0,1\}$

Signal variance assesses the separation between different classes, while noise variance evaluates the variation within a single class. Note that, under the homoscedasticity assumption (Yang et al., 2019), we assume equal noise variance across all classes. Higher signal variance leads to better class separation, while lower noise variance ensures tight clustering within classes, resulting in better classification quality (Fisher, 1936).

5.2.3 GNN Architecture

We utilize Graph Convolutional Network (GCN) (Kipf and Welling, 2017) to examine homophily/heterophily effects, aligning with numerous theoretical analyses in this area (Ma et al., 2022b; Wang et al., 2024b). The fundamental node update mechanism of GCN is:

$$\mathbf{X}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X}^{(l)}\mathbf{W}^{(l)}) \tag{5.1}$$

where $\mathbf{W}^{(l)}$ is the learnable parameter matrix at layer l, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops, $\tilde{\mathbf{D}}$ is the degree matrix corresponding to $\tilde{\mathbf{A}}$, and $\mathbf{X}^{(0)} =$

$\mathbf{X}$ are the initial node features. For analytical tractability, we analyze a simplified aggregation scheme that captures the essential neighborhood averaging behavior. Following Wu et al. (2019), which shows that most of the benefit in GCNs comes from local averaging rather than from nonlinear activation functions, we focus on the uniform averaging operation obtained through row normalization $\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$.

$$\mathbf{h}_v = \frac{1}{d_v + 1}\left(\mathbf{x}_v + \sum_{u \in \mathcal{N}(v)} \mathbf{x}_u\right) \tag{5.2}$$

5.2.4 Assumptions

We consider a contextual stochastic block model (CSBM) (Deshpande et al., 2018) where nodes are partitioned into two classes. Each node's feature follows a class-specific Gaussian distribution: $\mathbf{x}_v = \boldsymbol{\mu}^{y_v} + \boldsymbol{\epsilon}_v$ with $\boldsymbol{\epsilon}_v \sim \mathcal{N}(\mathbf{0}, \sigma^2_{\text{intra}}\mathbf{I}_d)$, where $\mathbf{I}_d$ is the $d \times d$ identity matrix. Under this model, for any node v, same-label neighbors $u \in \mathcal{N}_v^+$ have features $\mathbf{x}_u = \boldsymbol{\mu}^{y_v} + \boldsymbol{\epsilon}_u^+$, while opposite-label neighbors $u \in \mathcal{N}_v^-$ have features $\mathbf{x}_u = \boldsymbol{\mu}^{1-y_v} + \boldsymbol{\epsilon}_u^-$, where $1 - y_v$ denotes the opposite class label in the binary setting. All noise terms $\boldsymbol{\epsilon}_v, \{\boldsymbol{\epsilon}_u^+\}_{u \in \mathcal{N}_v^+}, \{\boldsymbol{\epsilon}_u^-\}_{u \in \mathcal{N}_v^-}$ are independent and follow $\mathcal{N}(\mathbf{0}, \sigma^2_{\text{intra}}\mathbf{I}_d)$. This CSBM setup is consistent with established theoretical frameworks for analyzing heterophily in GNNs (Ma et al., 2022b).

5.2.5 Aggregation Effect Analysis

In this section, we analyse how the neighbourhood aggregation impacts the node classification quality under varying homophily/heterophily conditions. To analyze this effect, we decompose GNN aggregation based on neighbourhood labels.

Definition 10 (Label-Based Aggregation)**.** *We decompose the GNN aggregation operation based on neighbor labels as:*

$$\mathbf{h}_v = \frac{1}{d_v + 1}\left(\mathbf{x}_v + \sum_{u \in \mathcal{N}_v^+} \mathbf{x}_u + \sum_{u \in \mathcal{N}_v^-} \mathbf{x}_u\right)$$

To quantify how neighbourhood aggregation affects the original class signal, we introduce signal preservation factor.

Definition 11 (Signal Preservation Factor)**.** *For node v, the signal preservation factor is:*

$$\alpha_v = \frac{1 + d_v^+ - d_v^-}{d_v + 1}$$

We present the following theorem, which characterises how neighbourhood label composition affects representation quality after aggregation.

Theorem 8 (Label Aggregation Effect)**.** *For any node v in the graph, under the stated graph-wide assumptions, the aggregated representation has expected value:*

$$\mathbb{E}[\mathbf{h}_v|y_v] = \frac{1+d_v^+}{d_v+1}\boldsymbol{\mu}^{y_v} + \frac{d_v^-}{d_v+1}\boldsymbol{\mu}^{1-y_v}$$

The signal variance after aggregation is:

$$\|\mathbb{E}[\mathbf{h}_v|y_v=0] - \mathbb{E}[\mathbf{h}_v|y_v=1]\|^2 = \alpha_v^2\Delta^2$$

The noise variance is: $Var[\mathbf{h}_v|y_v] = \frac{\sigma_{intra}^2}{d_v+1}$
The node-specific classification quality is: $Q_v = \frac{\alpha_v^2(d_v+1)\Delta^2}{\sigma_{intra}^2}$

Proof. We start by proving the Expected Representation. Starting from the aggregation formula:

$$\mathbf{h}_v = \frac{1}{d_v+1}\left(\mathbf{x}_v + \sum_{u\in\mathcal{N}_v^+}\mathbf{x}_u + \sum_{u\in\mathcal{N}_v^-}\mathbf{x}_u\right)$$

Taking expectations conditioned on the node's label y_v:

$$\begin{aligned}\mathbb{E}[\mathbf{h}_v|y_v] = \frac{1}{d_v+1}\Big(&\mathbb{E}[\mathbf{x}_v|y_v] + \sum_{u\in\mathcal{N}_v^+}\mathbb{E}[\mathbf{x}_u|y_v] \\ &+ \sum_{u\in\mathcal{N}_v^-}\mathbb{E}[\mathbf{x}_u|y_v]\Big)\end{aligned}$$

By the feature assumptions, $\mathbb{E}[\mathbf{x}_v|y_v] = \boldsymbol{\mu}^{y_v}$, $\mathbb{E}[\mathbf{x}_u|y_v] = \boldsymbol{\mu}^{y_v}$ for $u \in \mathcal{N}_v^+$, and $\mathbb{E}[\mathbf{x}_u|y_v] = \boldsymbol{\mu}^{1-y_v}$ for $u \in \mathcal{N}_v^-$. Substituting:

$$\begin{aligned}\mathbb{E}[\mathbf{h}_v|y_v] &= \frac{1}{d_v+1}\Big(\boldsymbol{\mu}^{y_v} + d_v^+\cdot\boldsymbol{\mu}^{y_v} + d_v^-\cdot\boldsymbol{\mu}^{1-y_v}\Big) \\ &= \frac{1+d_v^+}{d_v+1}\boldsymbol{\mu}^{y_v} + \frac{d_v^-}{d_v+1}\boldsymbol{\mu}^{1-y_v}\end{aligned}$$

Then, we prove the Signal Variance. For class 0 nodes ($y_v = 0$):

$$\mathbb{E}[\mathbf{h}_v|y_v=0] = \frac{1+d_v^+}{d_v+1}\boldsymbol{\mu}^0 + \frac{d_v^-}{d_v+1}\boldsymbol{\mu}^1$$

For class 1 nodes ($y_v = 1$):

$$\mathbb{E}[\mathbf{h}_v|y_v=1] = \frac{1+d_v^+}{d_v+1}\boldsymbol{\mu}^1 + \frac{d_v^-}{d_v+1}\boldsymbol{\mu}^0$$

Computing the difference:

$$\begin{aligned}
&\mathbb{E}[\mathbf{h}_v|y_v=0]-\mathbb{E}[\mathbf{h}_v|y_v=1]\\
&=\frac{1+d_v^+}{d_v+1}\boldsymbol{\mu}^0+\frac{d_v^-}{d_v+1}\boldsymbol{\mu}^1-\frac{1+d_v^+}{d_v+1}\boldsymbol{\mu}^1-\frac{d_v^-}{d_v+1}\boldsymbol{\mu}^0\\
&=\frac{1+d_v^+-d_v^-}{d_v+1}(\boldsymbol{\mu}^0-\boldsymbol{\mu}^1)
\end{aligned}$$

Taking the squared norm:

$$\begin{aligned}
&\|\mathbb{E}[\mathbf{h}_v|y_v=0]-\mathbb{E}[\mathbf{h}_v|y_v=1]\|^2\\
&=\left(\frac{1+d_v^+-d_v^-}{d_v+1}\right)^2\|\boldsymbol{\mu}^0-\boldsymbol{\mu}^1\|^2\\
&=\alpha_v^2\Delta^2
\end{aligned}$$

Then, we prove the Noise Variance. The noise variance is:

$$\mathrm{Var}[\mathbf{h}_v|y_v]=\mathrm{Var}\left[\frac{1}{d_v+1}\Big(\boldsymbol{\epsilon}_v+\sum_{u\in\mathcal{N}_v^+}\boldsymbol{\epsilon}_u^+ +\sum_{u\in\mathcal{N}_v^-}\boldsymbol{\epsilon}_u^-\Big)\right]$$

Since there are $1+d_v^++d_v^-=d_v+1$ independent noise terms, each with variance σ^2_{intra}:

$$\begin{aligned}
\mathrm{Var}[\mathbf{h}_v|y_v]&=\frac{1}{(d_v+1)^2}\cdot(d_v+1)\sigma^2_{\text{intra}}\\
&=\frac{\sigma^2_{\text{intra}}}{d_v+1}
\end{aligned}$$

Finally, we derive the Classification Quality. By definition, classification quality is the ratio of signal variance to noise variance:

$$\begin{aligned}
Q_v&=\frac{\alpha_v^2\Delta^2}{\sigma^2_{\text{intra}}/(d_v+1)}\\
&=\frac{\alpha_v^2(d_v+1)\Delta^2}{\sigma^2_{\text{intra}}}
\end{aligned}$$

This completes the proof. □

Employing Theorem 8, we derive the following observations.

Corollary 1 (Strong Homophily)**.** *When $d_v^+ \gg d_v^-$, the signal preservation factor $\alpha_v \approx 1$, and classification quality scales linearly with degree: $Q_v \propto d_v$.*

In strongly homophilic neighborhoods, aggregation generally benefits nodes regardless of their degree, since $\alpha_v \approx 1$ guarantees that the expected aggregated representation preserves class signal. However, higher-degree nodes achieve better per-

formance since increased sample (neighborhood) size decreases the probability of "error".

Corollary 2 (Strong Heterophily)**.** *When $d_v^- \gg d_v^+$, α_v transitions from 0 at low degrees to -1 at high degrees. Classification quality shows poor performance at low degrees due to signal cancellation ($Q_v \approx 0$ when d_v is small), but achieves performance comparable to homophilic cases at high degrees ($Q_v \propto d_v$ when d_v is large).*

Strong heterophily isn't inherently bad. It becomes problematic when there aren't enough neighbours to establish a reliable pattern. With sufficient degree, heterophilic nodes can perform as well as homophilic ones by learning from the opposite relationships.

Corollary 3 (Mixed Homophily/Heterophily)**.** *When $d_v^+ \approx d_v^-$ (balanced same/opposite-label neighbors), $\alpha_v \approx \frac{1}{d_v+1}$, causing signal cancellation that worsens with degree, leading to poor classification performance.*

Nodes with balanced neighbourhoods experience signal cancellation, as competing signals from same-class and opposite-class neighbours neutralize each other. This effect is particularly severe for high-degree nodes, where $\alpha_v \to 0$ as d_v increases, leaving such nodes with insufficient directional information for reliable classification.

5.2.6 Multi-Layer Analysis

We extend Theorem 8 to multi-layer setting to analyze the iterative aggregation effect in GNNs.

Theorem 9 (Iterative Aggregation Effect)**.** *Assume a n-layer GNN where: (1) label-conditioned features remain independent across layers, and (2) each layer performs the same neighborhood aggregation pattern with degree d_v. Then, for node v, after n layers:*

- *Signal variance:* $\alpha_v^{2n}\Delta^2$
- *Noise variance:* $\frac{\sigma_{intra}^2}{(d_v+1)^n}$
- *Classification quality:* $Q_v^n = \frac{\alpha_v^{2n}(d_v+1)^n\Delta^2}{\sigma_{intra}^2}$

Proof. We proceed by induction on the number of layers n.

First, consider the base case ($n = 1$). From Theorem 8, for a single aggregation layer:

$$\mathbb{E}[\mathbf{h}_v^{(1)}|y_v] = \frac{1+d_v^+}{d_v+1}\boldsymbol{\mu}^{y_v} + \frac{d_v^-}{d_v+1}\boldsymbol{\mu}^{1-y_v}$$

$$\mathrm{Var}[\mathbf{h}_v^{(1)}|y_v] = \frac{\sigma_{\text{intra}}^2}{d_v+1}$$

The signal variance is:

$$\begin{aligned}&\|\mathbb{E}[\mathbf{h}_v^{(1)}|y_v=0]-\mathbb{E}[\mathbf{h}_v^{(1)}|y_v=1]\|^2\\&=\left(\frac{1+d_v^+-d_v^-}{d_v+1}\right)^2\Delta^2=\alpha_v^2\Delta^2\end{aligned}$$

And classification quality:

$$Q_v^{(1)}=\frac{\alpha_v^2\Delta^2}{\sigma_{\text{intra}}^2/(d_v+1)}=\frac{\alpha_v^2(d_v+1)\Delta^2}{\sigma_{\text{intra}}^2}$$

Assume that after k layers:

$$\begin{aligned}\|\mathbb{E}[\mathbf{h}_v^{(k)}|y_v=0]-\mathbb{E}[\mathbf{h}_v^{(k)}|y_v=1]\|^2&=\alpha_v^{2k}\Delta^2\\\text{Var}[\mathbf{h}_v^{(k)}|y_v]&=\frac{\sigma_{\text{intra}}^2}{(d_v+1)^k}\\Q_v^{(k)}&=\frac{\alpha_v^{2k}(d_v+1)^k\Delta^2}{\sigma_{\text{intra}}^2}\end{aligned}$$

Consider the step ($k\to k+1$): At layer $k+1$, we apply the same aggregation operation to the outputs of layer k. Let $\mathbf{r}_0^{(k)}=\mathbb{E}[\mathbf{h}_v^{(k)}|y_v=0]$ and $\mathbf{r}_1^{(k)}=\mathbb{E}[\mathbf{h}_v^{(k)}|y_v=1]$. By hypothesis:

$$\|\mathbf{r}_0^{(k)}-\mathbf{r}_1^{(k)}\|^2=\alpha_v^{2k}\Delta^2$$

Applying one more aggregation layer:

$$\begin{aligned}\mathbb{E}[\mathbf{h}_v^{(k+1)}|y_v=0]&=\frac{1}{d_v+1}\left(\mathbf{r}_0^{(k)}+\sum_{u\in\mathcal{N}_v^+}\mathbf{r}_0^{(k)}+\sum_{u\in\mathcal{N}_v^-}\mathbf{r}_1^{(k)}\right)\\&=\frac{1+d_v^+}{d_v+1}\mathbf{r}_0^{(k)}+\frac{d_v^-}{d_v+1}\mathbf{r}_1^{(k)}\end{aligned}$$

Similarly:

$$\mathbb{E}[\mathbf{h}_v^{(k+1)}|y_v=1]=\frac{1+d_v^+}{d_v+1}\mathbf{r}_1^{(k)}+\frac{d_v^-}{d_v+1}\mathbf{r}_0^{(k)}$$

The signal difference becomes:

$$\begin{aligned}&\mathbb{E}[\mathbf{h}_v^{(k+1)}|y_v=0]-\mathbb{E}[\mathbf{h}_v^{(k+1)}|y_v=1]\\&=\frac{1+d_v^+-d_v^-}{d_v+1}(\mathbf{r}_0^{(k)}-\mathbf{r}_1^{(k)})\\&=\alpha_v(\mathbf{r}_0^{(k)}-\mathbf{r}_1^{(k)})\end{aligned}$$

Taking the squared norm:

$$\begin{aligned}
&\|\mathbb{E}[\mathbf{h}_v^{(k+1)}|y_v = 0] - \mathbb{E}[\mathbf{h}_v^{(k+1)}|y_v = 1]\|^2 \\
&= \alpha_v^2 \|\mathbf{r}_0^{(k)} - \mathbf{r}_1^{(k)}\|^2 \\
&= \alpha_v^2 \cdot \alpha_v^{2k}\Delta^2 = \alpha_v^{2(k+1)}\Delta^2
\end{aligned}$$

For the noise variance, each aggregation layer independently reduces variance by factor $(d_v + 1)$:

$$\begin{aligned}
\mathrm{Var}[\mathbf{h}_v^{(k+1)}|y_v] &= \frac{1}{d_v+1}\mathrm{Var}[\mathbf{h}_v^{(k)}|y_v] \\
&= \frac{1}{d_v+1} \cdot \frac{\sigma^2_{\text{intra}}}{(d_v+1)^k} \\
&= \frac{\sigma^2_{\text{intra}}}{(d_v+1)^{k+1}}
\end{aligned}$$

Finally, the classification quality becomes:

$$\begin{aligned}
Q_v^{(k+1)} &= \frac{\alpha_v^{2(k+1)}\Delta^2}{\sigma^2_{\text{intra}}/(d_v+1)^{k+1}} \\
&= \frac{\alpha_v^{2(k+1)}(d_v+1)^{k+1}\Delta^2}{\sigma^2_{\text{intra}}}
\end{aligned}$$

This completes the induction and proves all three statements. □

Theorem 9 shows how single-layer effects compound over multiple layers. Signal preservation factor α_v is raised to the power $2n$, meaning if $|\alpha_v| < 1$ (signal degradation), the effect becomes exponentially worse with depth. Conversely, noise reduction compounds beneficially, but the net effect depends on whether signal preservation dominates.

5.2.7 Limitations of our Theoretical Framework

While our theoretical framework offers a new perspective on GNN performance under varying homophily conditions, our analysis has several limitations. In this section, we discuss these limitations.

Simplified Aggregation Analysis We primarily focus on the uniform averaging operation, omitting several key components of real GCN architectures, including a symmetric normalisation scheme, weight transformation and non-linear activation functions. This is done to make our analysis more tractable. Our experimental results demonstrate that the insights derived under these relaxations are applicable to real-world settings.

Layer Independence Assumption Theorem 9 assumes independence between features across layers, which might not hold due to feature correlations and oversmoothing.Nonetheless, this assumption becomes more reasonable for shallow networks (i.e., 2-3 layers), which are generally optimal for node classification tasks. In such cases, the independence assumption serves as a useful first-order approximation.

Contextual Stochastic Block Model Structure Our analysis assumes a CSBM structure, which allows us to maintain Gaussian feature distributions and clear class separations. This aligns with existing theoretical frameworks built for GNNs that prioritise analytical tractability.

Binary Classification Restriction Our current theoretical analysis focuses on binary classification, which provides analytical feasibility. Our experimental results strongly validate the fact that our findings hold in multi-class settings. This transferability can be justified by several reasons.

- Regardless of distinguishing between binary or multi-classes, nodes still benefit from same-label neighbours and are negatively impacted by opposite-label neighbours, making the high-level concept of the homophily-heterophily tradeoff identical.
- The presented tradeoff between signal preservation and noise reduction scales naturally to multi-class scenarios. In signal preservation, same-label neighbours reinforce the correct class prototype, while opposite-label neighbours (regardless of their specific labels) hinder the signal. The principle of reducing noise through neighbourhood averaging remains consistent, irrespective of the number of classes.
- Our key insight about balanced neighbourhoods causing signal cancellation extends naturally to a multi-class setting. Nodes with equal proportions of different label neighbours experience diluted class-specific signals, whether those different classes represent a single alternative (binary) or multiple alternatives (multi-class).

Architecture Generalizability Our current theoretical analysis is based on GCN with uniform neighbour aggregation. However, our experimental analysis incorporates multiple GNN backbones with diverse architectures, demonstrating that our insights also hold for these.

5.3 Adaptive Depth Graph Neural Networks

We introduce AD-GNN, a novel architecture that leverages our theoretical findings to enhance message propagation in GNNs by allocating adaptive depth. The key insight is that different nodes benefit differently from deeper layers based on their local neighbourhood structure and multi-layer aggregation effects.

Let Q_v^n denote the classification quality for node v after n layers of message passing, and Q_v^0 represents the initial classification quality before any message passing, given by $Q_v^0 = \frac{\Delta^2}{\sigma_{\text{intra}}^2}$. For node v and target depth n, we define the Depth Benefit Metric:

Definition 12 (Depth Benefit Metric)**.** *For node v and target depth n, we define the Depth Benefit Metric:*

$$\varepsilon_v^n = \frac{Q_v^n}{Q_v^0} = \left(\alpha_v^2 \cdot (d_v + 1)\right)^n \tag{5.3}$$

The metric represents the multiplicative improvement in classification quality where $\varepsilon_v^n > 1$ indicates benefit, $\varepsilon_v^n = 1$ indicates no change, and $\varepsilon_v^n < 1$ indicates degradation.

5.3.1 Stopping Depth Assignment

Let $t_{\max}$ be the maximum allowable depth in the network. For each node $v \in V$, we define its stopping depth $T(v)$ as:

$$T(v) = \max\left\{t \in \{1, 2, \ldots, t_{\max}\} \,\middle|\, \tilde{\varepsilon}_v^{t_{\max}} \geq \tau_{\boldsymbol{\theta}}(t)\right\} \tag{5.4}$$

where $\tilde{\varepsilon}_v^{t_{\max}}$ is the depth benefit score normalized to $[0, 1]$ using min-max scaling across all nodes, and $\tau_{\boldsymbol{\theta}} : \mathbb{N} \to [0, 1]$ is a monotonically increasing adaptive threshold function parameterized by $\boldsymbol{\theta}$. The monotonically increasing property ensures progressive filtering where nodes with lower depth benefit scores are gradually excluded as network depth increases, preventing nodes that cease to benefit from message passing at layer t from re-entering at deeper layers $t' > t$. The threshold function is defined as:

$$\tau_{\boldsymbol{\theta}}(t) = \lambda + (1 - \lambda) \cdot \theta(t) \tag{5.5}$$

where $\lambda \in [0, 1]$ is a hyperparameter, and $\theta(t) : \mathbb{N} \to [0, 1]$ is a learnable monotonically increasing function. The parameter λ establishes a minimum threshold ensuring $\tau_{\boldsymbol{\theta}}(t) \in [\lambda, 1]$, while the learnable component $\theta(t)$ adaptively optimizes the threshold across layers.

5.3.2 Message Passing Framework

For each node v and step $t \leq T(v)$, the message passing consists of aggregation and update steps:

$$\mathbf{m}_v^{(t)} = \text{AGG}\left(\left\{\!\!\left\{\mathbf{h}_u^{(\min\{t-1, T(u)\})} \mid u \in \mathcal{N}(v)\right\}\!\!\right\}\right) \tag{5.6}$$

$$\mathbf{h}_v^{(t)} = \text{UPD}\left(\mathbf{h}_v^{(t-1)}, \mathbf{m}_v^{(t)}\right) \tag{5.7}$$

For any $t > T(v)$, we have $\mathbf{h}_v^{(t)} = \mathbf{h}_v^{(t-1)}$. The adaptive depth mechanism can enhance existing GNN architectures such as GCN (Kipf and Welling, 2017), GAT (Veličković

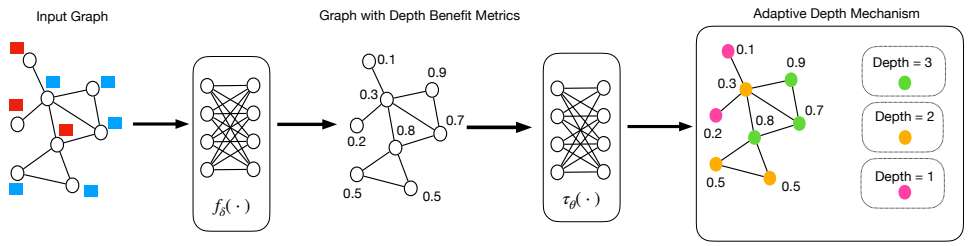


Figure 5.1: High-level workflow of AD-GNN: First, the depth benefit metric for each node is computed using their node profile and learned label probabilities. Then, the depth for each node is determined based on a learnable threshold mechanism. Nodes that have higher depth benefit metrics will receive more aggregation layers compared to others.

et al., 2018), and GraphSAGE (Hamilton et al., 2017) by utilizing their original aggregation and update functions along with node-specific depth allocation based on the depth benefit metric.

5.3.3 Depth Benefit Metric Calculation

Computing the depth benefit metric requires estimating α_v for each node, which is challenging due to incomplete label information during semi-supervised training. We use feature similarity to estimate same-label probabilities between adjacent nodes:

$$p_{uv} = f_\delta(\mathbf{h}_u, \mathbf{h}_v) \tag{5.8}$$

where $f_\delta : \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ is a learnable, permutation-invariant function that maps pairs of node embeddings to same-label probabilities. The expected counts for same-label and opposite-label neighbors are:

$$\hat{d}_v^+ = \sum_{u \in \mathcal{N}(v)} f_\delta(\mathbf{h}_u, \mathbf{h}_v) \tag{5.9}$$

$$\hat{d}_v^- = \sum_{u \in \mathcal{N}(v)} (1 - f_\delta(\mathbf{h}_u, \mathbf{h}_v)) \tag{5.10}$$

The estimated signal preservation factor and depth benefit metric are:

$$\hat{\alpha}_v = \frac{1 + \hat{d}_v^+ - \hat{d}_v^-}{d_v + 1} \tag{5.11}$$

$$\hat{\varepsilon}_v^{t_{\max}} = \left(\hat{\alpha}_v^2 \cdot (d_v + 1)\right)^{t_{\max}} \tag{5.12}$$

This creates a differentiable mechanism that enables end-to-end optimization where depth allocation adapts based on learned similarity assessments and predicted depth benefits.

5.3.4 Training Objective

To ensure f_δ learns meaningful similarity assessments, we incorporate regularization using available label information on training nodes:

$$\mathcal{L}_{\text{reg}} = -\frac{1}{|E_{\text{train}}|} \sum_{(u,v)\in E_{\text{train}}} [y_{uv} \log\left(f_\delta(\mathbf{h}_u, \mathbf{h}_v)\right) + (1 - y_{uv}) \log\left(1 - f_\delta(\mathbf{h}_u, \mathbf{h}_v)\right)] \tag{5.13}$$

where $E_{\text{train}} = \{(u,v) \in E : u, v \in V_{\text{train}}\}$ is the set of edges between training nodes, and $y_{uv} = \mathbb{I}[y_u = y_v]$ is the true same-label indicator. The total training objective becomes:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \mathcal{L}_{\text{reg}} \tag{5.14}$$

where $\mathcal{L}_{\text{task}}$ is the downstream task loss. The High-level overview of AD-GNN is depicted in Figure 5.1.

5.3.5 Fast Variant

To accelerate computation, we derive AD-GNN$_{\text{fast}}$ by replacing the learnable similarity function with a static degree-based approximation. Leveraging high-degree assortativity (Arcagni et al., 2017), the principle that high-degree nodes tend to connect to other high-degree nodes and share similar labels in the network, we compute same-label probability as:

$$p_{uv} = \frac{d_u \times d_v}{\max_{(i,j)\in E}(d_i \times d_j)} \tag{5.15}$$

The estimated depth benefit metric becomes:

$$\hat{\varepsilon}_v^{t_{\max}} = \left(\left(\hat{\alpha}_{v,\text{degree}}\right)^2 \cdot (d_v + 1)\right)^{t_{\max}} \tag{5.16}$$

where $\hat{\alpha}_{v,\text{degree}}$ is computed using the degree-based same-label probability estimates. This variant eliminates the computational overhead of learning f_δ and requires no regularization term, limiting the training objective to the downstream task loss alone.

5.3.6 Complexity Analysis

We analyse the complexity of AD-GNN variants separately from the backbone GNN architecture. AD-GNN incurs $\mathcal{O}(|E| \times d)$ complexity for feature-based label probability computation, where d is the feature dimension, $\mathcal{O}(|E| + |V|)$ for depth benefit metric computation, $\mathcal{O}(|E| + |V|)$ per-layer complexity for threshold filtering mechanism, and $\mathcal{O}(|E_{\text{train}}|)$ for regularization term. This takes the total complexity of AD-GNN to $\mathcal{O}(|E| \times d + t_{\max} \times (|E| + |V|))$. AD-GNN$_{\text{fast}}$ achieves a reduced complexity of $\mathcal{O}(t_{\max} \times (|E| + |V|))$ by eliminating feature-based label probability computation

using degree-based similarity approximation, which incurs only $\mathcal{O}(|E|)$ complexity. Additionally, the fast variant does not require computing the regularization term, further reducing training overhead. Note that, since AD-GNN progressively filters edges at each layer, it also reduces the backbone GNN complexity by operating on smaller edge sets.

5.4 Experimental Design

5.4.1 Datasets

We evaluate our approach for the node classification task using 11 datasets that cover both homophilic and heterophilic settings. Homophilic datasets include Cora ML, Citeseer, Pubmed, and DBLP from the CitationFull benchmark (Bojchevski and Günnemann, 2018), as well as the Photo dataset from the Amazon benchmark (Shchur et al., 2018). Heterophilic datasets include Texas, Cornell, Wisconsin, Squirrel, Chameleon, and Film datasets from WebKB benchmark (Pei et al., 2020b). Additionally, we employ the ogbn-arxiv dataset from OGB benchmark (Hu et al., 2020) for our scalability analysis.

Dataset statistics, including homophily ratios, graph sizes, feature dimensions, and number of labels, are depicted in Table 5.1.

Dataset	Homophily Ratio	#Nodes	#Edges	#Classes	#Features
Cora ML	0.79	2,995	16,316	7	2,879
Citeseer	0.71	3,327	4,732	6	3,703
Pubmed	0.79	19,717	44,338	3	500
Photo	0.83	7,650	119,081	8	745
DBLP	0.83	17,716	105,734	4	1,639
Film	0.24	7,600	33,544	5	931
Squirrel	0.22	5,201	217,073	5	2,089
Chameleon	0.25	2,277	36,101	5	2,325
Cornell	0.11	183	295	5	1,703
Wisconsin	0.16	251	499	5	1,703
Texas	0.06	183	309	5	1,703
ogbn-arxiv	0.65	169,343	1,166,243	40	128

Table 5.1: Dataset statistics.

5.4.2 Baselines

We employ three classical GNNs: GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017), along with three modern GNNs: MixHop (Abu-El-Haija et al., 2019), GATv2 (Brody et al., 2022), and DirGNN (Rossi et al., 2024), as our backbones.

5.4.3 Evaluation Setting

We use a 60/20/20 random split strategy for the training/validation/test sets and report the mean and standard deviation of accuracy over 10 random initialization, similar to the setup in Suresh et al. (2021); Zheng et al. (2023b). For Squirrel and Chameleon datasets, we employ the data splits provided by Platonov et al. (2023), which contain filtered datasets with duplicate nodes removed. We report baseline results from previous papers using the same experimental setup. If unavailable, we generate baseline results based on the hyperparameters from the original papers. When a baseline model is modified using the AD-GNN, the resulting version is named with the prefix "AD-", such as AD-GCN. Also, the AD-GNN fast variant is denoted by an additional subscript $_{\text{fast}}$, such as AD-GCN$_{\text{fast}}$.

5.4.4 Model Hyper-Parameters

We search model hyper-parameters in following ranges: number of layers $\in \{2, 3, 4\}$, dropout $\in \{0.1, 0.5, 0.6\}$, learning rate $\in \{0.001, 0.005, 0.01\}$, hidden dimension $\in \{128, 256\}$, weight decay $\in \{1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-1}\}$, epochs $\in \{200, 500, 1000\}$, and $\lambda \in \{0.0, 0.1, 0.8, 0.9\}$. We employ the Adam algorithm (Kingma, 2015) for optimisation.

5.5 Results and Discussion

5.5.1 Node Classification

We present the node classification results on homophilic graphs in Table 5.2 and heterophilic graphs in Table 5.3. Our approach consistently outperforms the baseline methods in both homophilic and heterophilic graph benchmarks. Notably, our adaptive layer mechanism demonstrates greater performance improvements on heterophilic graphs compared to homophilic ones. This is mainly due to heterophilic datasets being more vulnerable to signal degradation than homophilic ones, as demonstrated in the theoretical analysis. Additionally, we observe that our method enhances the performance of GNNs that are designed with inductive biases for heterophilic graph settings, such as MixHop and DirGNN. This shows that our approach complements these models by capturing additional structural and label-dependent information that might otherwise be overlooked. Furthermore, the fast variant of AD-GNN also performs well, achieving results comparable to and sometimes surpassing those of the original AD-GNN, offering a scalable yet powerful solution.

5.5.2 Case Study

We conduct experiments to empirically validate the observations presented in Theorem 8. In these experiments, we generate a synthetic graph using a stochastic block model with controllable homophily by explicitly forming a specified proportion of

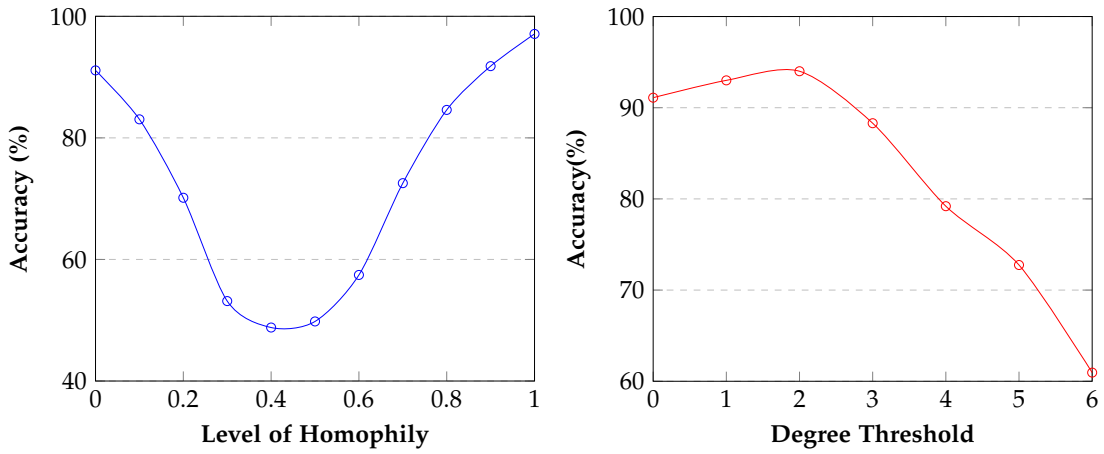
Methods	Cora-ML	Citeseer	Pubmed	Photo	DBLP
GCN	87.07 ±1.21	76.68 ±1.64	86.74 ±0.47	89.30 ±0.82	83.93 ±0.34
AD-GCN	87.32 ±1.25	**79.14** ±1.00	88.39 ±0.32	94.10 ±0.31	**84.14** ±0.44
AD-GCN$_{\text{fast}}$	**87.34** ±1.35	79.13 ±0.99	**88.41** ±0.37	**94.10** ±0.31	83.90 ±0.97
GAT	84.12 ±0.55	75.46 ±1.72	87.24 ±0.55	90.81 ±0.22	80.61 ±1.21
AD-GAT	**85.02** ±1.64	79.92 ±0.76	87.38 ±0.33	**94.03** ±0.34	**83.94** ±0.40
AD-GAT$_{\text{fast}}$	84.37 ±1.80	**79.95** ±0.78	**87.43** ±0.34	93.84 ±0.33	83.45 ±0.43
GraphSAGE	86.52 ±1.32	76.04 ±1.30	88.45 ±0.50	94.23 ±0.62	86.16 ±0.50
AD-GraphSAGE	**87.14** ±1.56	79.84 ±1.26	**88.99** ±0.64	**94.61** ±0.35	85.00 ±1.60
AD-GraphSAGE$_{\text{fast}}$	87.11 ±1.63	**79.88** ±1.27	88.88 ±0.60	94.54 ±0.41	**85.03** ±1.39
MixHop	87.29 ±1.19	70.75 ±2.95	80.75 ±2.29	94.83 ±0.41	84.27 ±0.31
AD-MixHop	87.66 ±1.24	**81.01** ±1.36	90.09 ±0.58	**95.09** ±0.57	84.26 ±0.81
AD-MixHop$_{\text{fast}}$	**87.73** ±1.31	80.95 ±1.33	**90.23** ±0.42	94.55 ±0.57	**84.46** ±0.79
GATv2	85.10 ±1.84	76.31 ±1.62	88.77 ±0.39	94.03 ±0.44	**84.85** ±0.61
AD-GATv2	**85.17** ±1.50	**80.10** ±0.99	**89.26** ±0.58	**94.33** ±0.70	84.26 ±0.52
AD-GATv2$_{\text{fast}}$	85.02 ±1.26	79.11 ±1.48	88.25 ±0.24	94.23 ±0.68	84.26 ±0.52
DirGNN	85.66 ±0.31	77.71 ±0.78	86.94 ±0.55	95.38 ±0.32	81.22 ±0.54
AD-DirGNN	**87.25** ±1.42	**78.36** ±1.82	**89.35** ±0.30	**95.55** ±0.51	**83.52** ±0.39
AD-DirGNN$_{\text{fast}}$	85.85 ±1.90	78.35 ±1.80	89.07 ±0.29	94.99 ±0.59	82.80 ±0.65

Table 5.2: Node classification accuracy ± standard deviation (%) on homophilic graphs. The best results are highlighted. Baseline results are sourced from Suresh et al. (2021); Platonov et al. (2023); Zheng et al. (2023b), and Chen et al. (2025).

Methods	Film	Squirrel	Chameleon	Cornell	Wisconsin	Texas
GCN	30.26 ±0.79	39.47 ±1.47	40.89 ±4.12	55.14 ±8.46	61.60 ±7.00	60.00 ±6.45
AD-GCN	**42.54** ±1.15	40.04 ±0.99	**43.66** ±0.73	**88.51** ±4.87	93.88 ±3.03	**92.30** ±4.52
AD-GCN$_{\text{fast}}$	41.39 ±1.46	**40.88** ±1.09	43.40 ±0.56	87.02 ±3.49	**94.38** ±2.86	91.64 ±5.16
GAT	26.28 ±1.73	35.62 ±2.06	39.21 ±3.08	53.64 ±11.1	60.00 ±11.0	61.21 ±8.17
AD-GAT	**41.25** ±0.77	36.73 ±0.83	**40.52** ±1.55	**86.17** ±4.69	91.50 ±2.42	90.49 ±5.72
AD-GAT$_{\text{fast}}$	41.03 ±0.96	**36.92** ±0.99	40.36 ±2.54	85.96 ±3.18	**92.62** ±2.82	**90.82** ±4.65
GraphSAGE	34.23 ±0.99	36.09 ±1.99	37.77 ±4.14	75.95 ±5.01	81.18 ±5.56	82.43 ±6.14
AD-GraphSAGE	**40.92** ±1.46	40.88 ±0.87	**40.10** ±1.45	89.57 ±4.30	**94.62** ±2.56	**92.95** ±2.84
AD-GraphSAGE$_{\text{fast}}$	40.90 ±1.49	**41.08** ±0.73	39.79 ±1.87	**90.64** ±4.28	94.25 ±2.38	92.79 ±2.95
MixHop	32.22 ±2.34	38.85 ±0.89	42.94 ±1.01	73.51 ±6.34	75.88 ±4.90	77.84 ±7.73
AD-MixHop	**43.37** ±1.17	39.76 ±0.90	**46.29** ±1.19	**90.21** ±3.59	**94.75** ±2.22	**94.43** ±2.56
AD-MixHop$_{\text{fast}}$	43.14 ±1.80	**40.27** ±1.05	45.05 ±1.83	90.00 ±3.16	92.00 ±3.41	90.00 ±5.65
GATv2	34.90 ±0.79	35.24 ±0.63	42.53 ±1.18	61.35 ±3.21	64.12 ±4.81	67.84 ±4.75
AD-GATv2	40.21 ±2.11	37.96 ±0.94	**42.99** ±1.25	86.38 ±4.48	91.25 ±1.94	90.16 ±4.69
AD-GATv2$_{\text{fast}}$	**42.02** ±1.88	**40.82** ±0.41	42.63 ±1.03	**87.66** ±5.19	**92.37** ±2.20	**92.46** ±4.47
DirGNN	35.76 ±1.68	38.67 ±1.09	**42.94** ±1.66	76.51 ±6.14	80.50 ±5.50	76.25 ±6.31
AD-DirGNN	**41.92** ±1.82	**40.58** ±0.94	42.06 ±0.77	**91.70** ±2.60	**95.13** ±1.89	**92.95** ±2.21
AD-DirGNN$_{\text{fast}}$	41.23 ±1.75	39.36 ±1.12	41.70 ±1.07	89.57 ±3.86	93.38 ±2.50	92.79 ±3.04

Table 5.3: Node classification accuracy ± standard deviation (%) on heterophilic graphs. The best results are highlighted. Baseline results are sourced from Suresh et al. (2021); Platonov et al. (2023); Zheng et al. (2023b), and Chen et al. (2025).

edges between same-class versus different-class nodes. We then measure the GCN performance under two cases.



(a) (b)

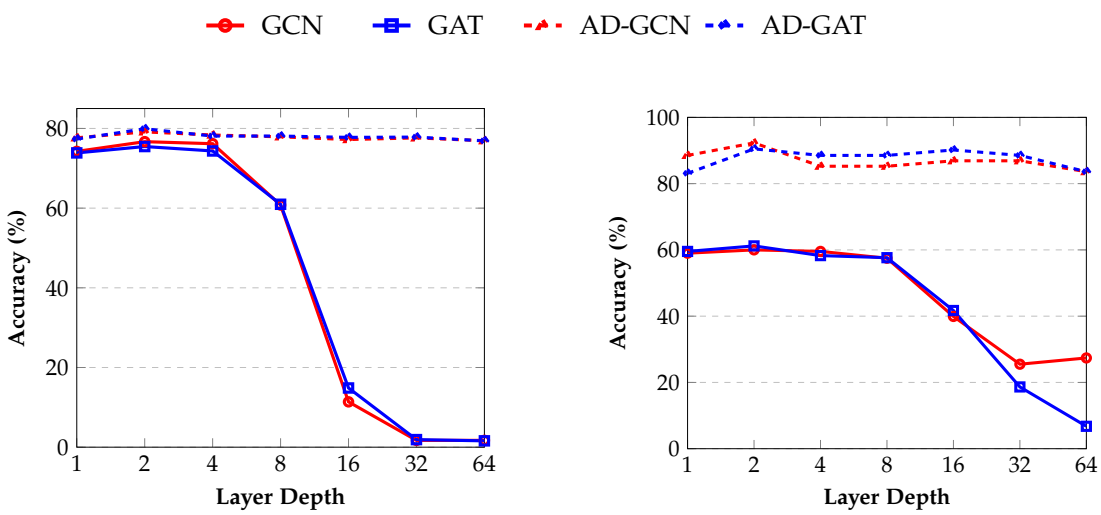
Figure 5.2: Case studies to empirically validate observations from Theorem 8.

In Figure 5.2(a), we evaluate performance across varying homophily degrees, from ideal heterophily to ideal homophily. Performance remains stable under both extremes, confirming Corollary 1 and Corollary 2. However, we observe performance decline in mixed homophily scenarios due to signal cancellation between neighbors with balanced same or opposite-label configurations, consistent with Corollary 3.

Figure 5.2(b) validates the low-degree scenario highlighted in Corollary 2. In this scenario, we examine strong heterophily while avoiding aggregation of nodes that fall below a certain degree threshold. The results demonstrate that excluding low-degree nodes (specifically, those with a degree of 1-2) from aggregation can improve performance. This improvement is attributed to the complete signal cancellation that occurs in low-degree nodes under strong heterophily conditions. By avoiding these nodes, we can achieve better performance. Conversely, preventing aggregation for high-degree nodes results in a decline in performance.

5.5.3 Oversmoothing Analysis

Figure 5.3 shows AD-GNN performance under varying layer depth. While traditional GNNs exhibit rapid degradation with increased depth, AD-GNN variants maintain consistent performance across deeper layers, effectively mitigating oversmoothing. This robustness can be attributed to our adaptive layer mechanism, which regulates information propagation to prevent excessive signal degradation.

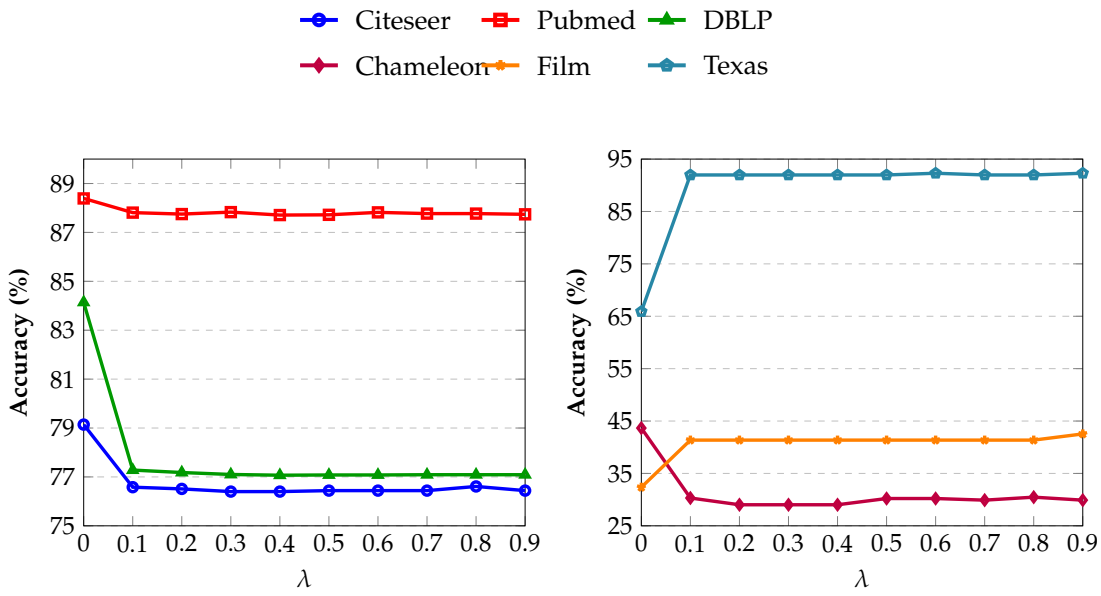


(a) Citeseer (b) Texas

Figure 5.3: Oversmoothing comparison.

5.5.4 Hyperparameter Sensitivity Analysis

We analyse the impact of hyperparameter λ for both homophilic and heterophilic datasets in Figure 5.4. Degree distributions for corresponding datasets are depicted in Figure 5.5.

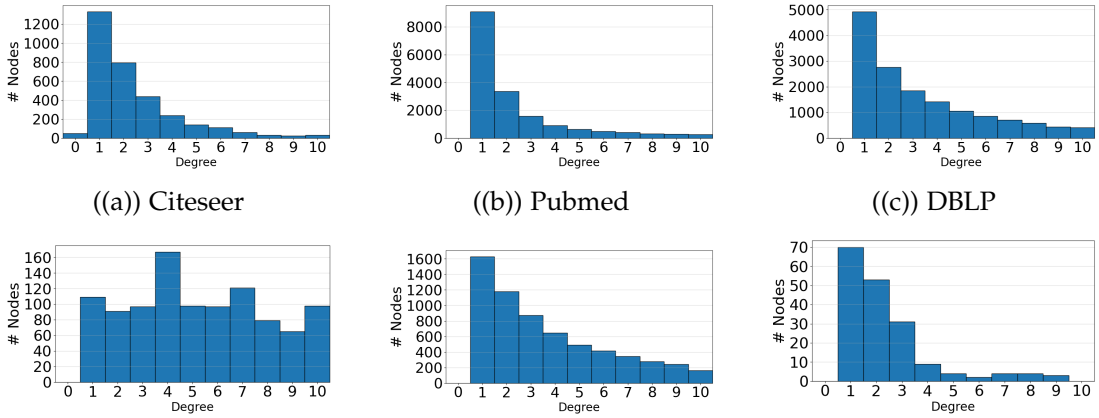


(a) Homophilic Graphs (b) Heterophilic Graphs

Figure 5.4: Sensitivity analysis of parameter λ on AD-GCN.

For all homophilic graphs (Citeseer, Pubmed, and DBLP), the best results are achieved with $\lambda = 0$. This is primarily because setting $\lambda = 0$ guarantees that every node is aggregated at least once. According to Corollary 1, aggregation under strong homophily is always beneficial, regardless of the degree.

Heterophilic graphs (Chameleon, Film, and Texas) exhibit some interesting deviations. Chameleon performs better when nodes have at least one aggregation layer (i.e., when $\lambda = 0$). This is mainly because most nodes in the graph have moderate to high degrees, which makes aggregation beneficial (as noted in Corollary 2). In contrast, the majority of nodes in Texas and Film have lower degrees (1-2). According to Corollary 2, under strong heterophily, low-degree nodes experience signal cancellation (i.e. $\alpha_v = 0$) during the aggregation process. Therefore, setting $\lambda > 0$ (meaning some nodes do not aggregate at all) would improve performance.



((a)) Citeseer ((b)) Pubmed ((c)) DBLP

((d)) Chameleon ((e)) Film ((f)) Texas

Figure 5.5: Degree distribution of datasets.

5.5.5 Scalability Analysis

We assess the scalability of AD-GNN using a large dataset (ogbn-arxiv) with varying depths. The hidden layer size is fixed at 128 for all methods. Each model is evaluated based on the number of learnable parameters, average runtime per epoch, and accuracy. The results are depicted in Table 6.9. Both AD-GCN variants exhibit a reasonable increase in computational complexity compared to the base model, while providing performance enhancements. Specifically, AD-GCN$_{\text{fast}}$ incurs minimal computational overhead, with a parameter count similar to that of GCN and a runtime increase of only around 5%.

	Depth = 4			Depth = 8		
	# Param (K)	Time (ms)	Acc. (%)	# Param (K)	Time (ms)	Acc. (%)
GCN	55.5	277.13	69.53	122.5	564.73	68.39
AD-GCN	88.5	405.85	70.32	155.6	655.52	70.63
AD-GCN$_{\text{fast}}$	55.5	286.40	70.18	122.5	580.15	70.42

Table 5.4: Performance comparison of AD-GCN variants.

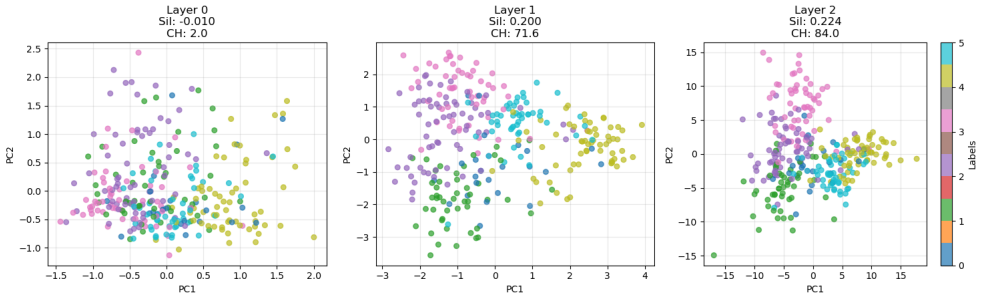
5.5.6 Visualization Analysis

We provide two visualisation experiments related to AD-GNN. First, we analyse the embedding quality evolution of AD-GNN variants compared to the baselines in Figure 5.6. We also present the Silhouette Score and Calinski-Harabasz Score (Wang and Xu, 2019) as metrics for cluster quality. Higher values of these metrics indicate better embedding clustering, resulting in improved classification. Based on the embedding visualisations, AD-GNN variants demonstrate superior performance compared to the baselines. AD-GNN achieves significantly better final layer separation, with a higher Silhouette score and a substantially higher Calinski-Harabasz score, compared to baselines. This indicates that AD-GNN's adaptive depth mechanism enables more effective feature learning and information propagation, resulting in better class boundaries in the final embedding space.

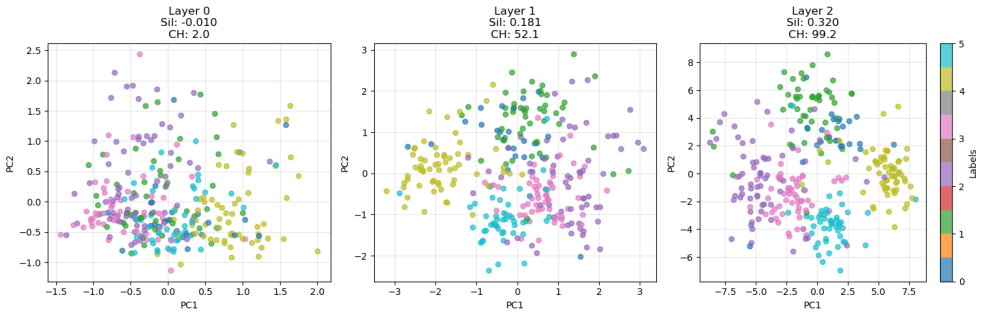
In the second experiment, we analyse the average depth benefit metric computed by AD-GCN across different node degrees in various datasets. The corresponding visualisation is shown in Figure 5.7. As observed, the depth benefit metric tends to increase with node degree across all datasets. This trend is primarily due to the decrease in noise variance as node degree increases. Also, heterophilic graphs exhibit lower depth benefit metrics, as nodes in these datasets tend to have mixed label neighbourhoods, compared to homophilic ones. Additionally, in heterophilic datasets, where most nodes have low degrees (such as Cornell and Texas), the depth benefit metric values are typically much lower. This is because avoiding aggregation for nodes with low degrees is beneficial under strong heterophily, as demonstrated in Corollary 2.

5.6 Summary

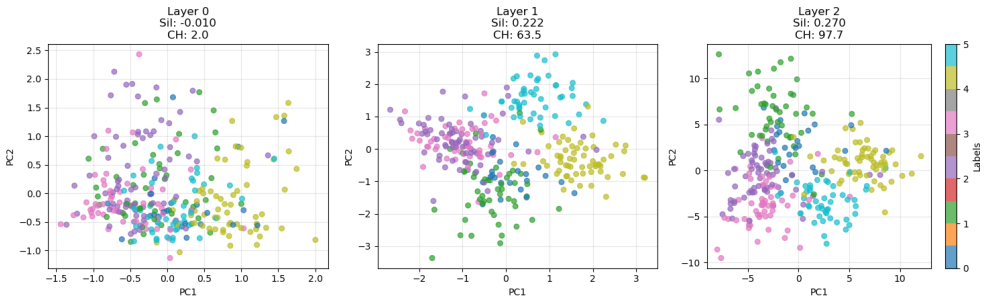
In this chapter, we provide novel theoretical insights into the impact of layer depth on the classification quality of nodes under varying homophily for node classification. Our findings indicate that strong heterophily is not necessarily negative, and the aggregation requirements of a node depend on the distribution of same-label neighbours, opposite-label neighbours, and total degree within its neighbourhood. By applying our theoretical insights, we develop a novel GNN plugin that adaptively determines the layer depth for each node, thereby improving their classification quality. Experiments on diverse graph structures and multiple GNN backbones demonstrate that our solution can consistently uplift performance in the node classification



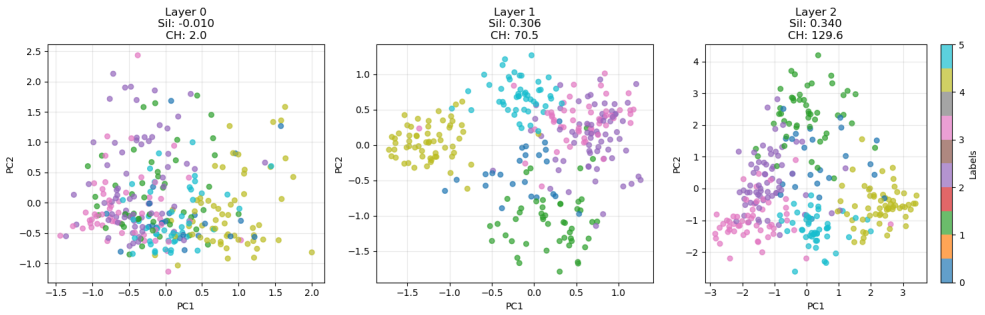
((a)) GCN



((b)) GAT



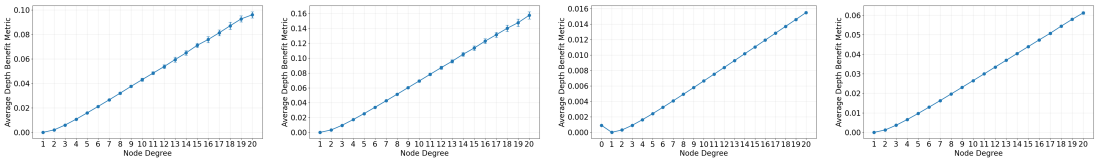
((c)) AD-GCN



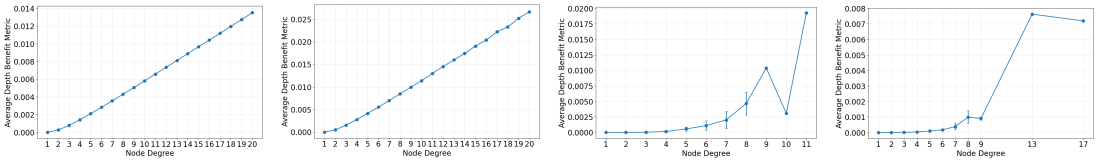
((d)) AD-GAT

Figure 5.6: Layer-wise Embedding Visualization for Citeseer Dataset. Sil and CH acronyms refer to the silhouette score and the Calinski-Harabasz score, respectively.

task.



((a)) Cora ML ((b)) Pubmed ((c)) Photo ((d)) DBLP



((e)) Squirrel ((f)) Chameleon ((g)) Cornell ((h)) Texas

Figure 5.7: Average depth benefit metric computed per degree in homophilic and heterophilic datasets.

Chapter 6

Graph Neural Diffusion via Generalized Opinion Dynamics

6.1 Overview

Recently, researchers have increasingly focused on formulating the message passing process in GNNs as a diffusion phenomenon to better understand and improve the propagation of information in graph data (Gasteiger et al., 2019; Zhao et al., 2021; Chamberlain et al., 2021). These models are inspired by physical diffusion processes (Han et al., 2023a), offering distinct advantages over traditional GNNs as they explicitly model the dynamics of information flow across networks. Although traditional GNNs rely mainly on aggregation operations in the local neighborhood (Kipf and Welling, 2017; Veličković et al., 2018), diffusion-based GNNs leverage principles from random walks, heat diffusion, and other physical processes, enabling them to capture local and global structural dependencies in graphs in a more effective manner (Lee and Jung, 2023; Gasteiger et al., 2019).

Despite their strengths, diffusion GNNs face several notable limitations. **Homogeneous Diffusion with Static Dynamics:** Traditional diffusion GNNs assume uniform information propagation across all nodes and fixed dynamics over time (Chamberlain et al., 2021; Gasteiger et al., 2019). This prevents them from capturing node-specific behaviors and temporal variations in information flow. Therefore, these models struggle to adapt to dynamic, node-dependent diffusion patterns, limiting their effectiveness in complex real-world tasks. **Computational Complexity and Interpretability Issues:** Deeper GNNs are often employed to enhance expressiveness, but this comes at the cost of increased model parameters, particularly due to unshared weights, which significantly raises computational overhead (Li et al., 2021a; Liu et al., 2020). Additionally, deeper architectures obscure the flow of information, making it more difficult to interpret how predictions are formed. **Limited Understanding of Convergence Properties:** While existing diffusion-based GNNs offer some insights into basic convergence dynamics, a rigorous theoretical characterization of their representational capacity remains lacking. In particular, the boundaries of the convergence configurations these models can capture are not well defined. Furthermore, the role of diffusion parameters in shaping convergence and represen-

tational behavior has not been systematically analyzed, hindering the adaptability of these models to diverse and complex tasks.

To address the aforementioned limitations, we propose incorporating principles from opinion dynamics into diffusion GNNs. Opinion dynamic models, such as the Friedkin-Johnsen (Friedkin and Johnsen, 1990), are robust mathematical frameworks that are designed to capture how opinions propagate within a group of agents (i.e., nodes) in a social network (Das et al., 2014; Hassani et al., 2022). These models trace dynamic behaviors of the underlying propagation, depicting how individual opinions evolve over time and eventually converge. We observe that these models exhibit unique characteristics that can significantly strengthen diffusion GNNs. **Heterogeneous Diffusion with Temporal Evolution:** Opinion dynamics usually model node-specific (i.e., agent-specific) behaviors through parameters such as stubbornness (i.e., attachment to initial opinion) and capture temporal evolution through dynamic influence modeling between neighbors, enabling more complex propagations. **Efficiency and Interpretability:** Opinion dynamics provide interpretable, sequential information flow that can introduce a strong inductive bias. Unlike traditional GNNs that rely on heavy learnable transformations at each step, opinion dynamics achieve effective propagation through principled update rules with minimal parameterization, remaining simple and coherent even as model depth increases. **Theoretically Established Convergence Properties:** Opinion dynamics builds upon well-established mathematical foundations with provable convergence configurations (DeGroot, 1974; Friedkin and Johnsen, 1990; Rainer and Krause, 2002), offering stronger theoretical insights into information propagation under diverse conditions. They also provide proper guidance on the impact of their propagation parameters, enabling adaptive parameter tuning based on task requirements.

However, integrating opinion dynamics into GNNs is a challenging task since it requires bridging several technical and conceptual gaps between computational social science and GNNs. This integration must effectively accommodate temporal modeling without incurring excessive computational costs and preserve convergence properties within the context of neural network training. Additionally, incorporating heterogeneous influence mechanisms requires adjustments to traditional message-passing schemes, which introduces new interpretability trade-offs that need careful consideration. Since different opinion dynamics models have distinct advantages, a unified framework is necessary to effectively integrate these mechanisms and fully leverage their potential to enhance GNN performance across various graph learning tasks. To address these challenges, we propose a novel Generalized Opinion Dynamics Neural Framework, *GODNF*, which is a more robust and theoretically grounded alternative to conventional diffusion GNNs.

The main contributions in this chapter are summarized as follows.

- **Generalized Opinion Dynamics Framework**: We introduce GODNF, a generalized framework grounded in multiple opinion dynamic models, which provides a principled foundation for designing diffusion-based GNNs and enables the efficient modeling of diverse information propagation behaviors.

- **Temporal Message Passing with Node-Specific Responses**: We advance diffusion GNNs by designing a message passing framework that captures evolving temporal dynamics with node-specific responses.
- **Deep Architecture with Efficiency**: We show that our framework can accommodate deep layers while maintaining interpretability and computational efficiency.
- **Theoretical Convergence Guarantees**: We provide theoretical insights by proving that our framework guarantees convergence and can exhibit diverse convergence configurations, offering greater flexibility in modeling various diffusion behaviors.
- **Empirical Performance**: Our extensive experiments on both static and temporal prediction tasks demonstrate that GODNF outperforms leading spatial and diffusion-based GNN models.

6.2 A Generalized Opinion Dynamics Framework for Graph Neural Diffusion

In this section, we introduce a generalized discrete-time framework that leverages opinion dynamics for diffusion GNNs. We demonstrate how node features in diffusion GNNs evolve similarly to opinions in social networks, facilitating the transfer of insights across domains to develop a more powerful learning method for capturing complex diffusion patterns in graph-structured data.

6.2.1 Preliminaries

Let $G = (V, E)$ be an undirected graph, where V denotes the set of nodes, E the set of edges, $|V| = n$, and $|E| = m$. The nodes are represented by the feature matrix $H \in \mathbb{R}^{n \times f}$, where f denotes the number of features per node. Further, $N(i)$ represents the set of nodes adjacent to node i.

6.2.2 Diffusion Mechanism in GODNF

Let $h_i \in H$ denote the feature vector of node i, and f_θ be a neural network parameterized by θ. The update rule of GODNF is defined as follows:

$$x_i(0) = f_\theta(h_i),$$

$$x_i(t+1) = \alpha \underbrace{x_i(t)}_{\text{(a)}} + (1-\alpha)\Big[\lambda_i \underbrace{x_i(0)}_{\text{(b)}} + (1-\lambda_i)\Big(\underbrace{\sum_{j\in\mathcal{N}(i)} w_{ij}(t)x_j(t)}_{\text{(c)}} - \mu \underbrace{L_g x_i(t)}_{\text{(d)}}\Big)\Big] \tag{6.1}$$

The update dynamics for a node's feature vector combine current feature retention, initial feature attachment, neighborhood influence, and structural regularization:

(a) **Currrent Feature Retention**: $x_i(t)$ represents the node's current feature vector. Controlled by a retention parameter α ($0 \leq \alpha < 1$), this term preserves the node's recent state.

(b) **Initial Feature Attachment**: $x_i(0)$ represents the node's initial feature after neural transformation, and the learnable parameter λ_i represents the node's stubbornness. A higher λ_i means node i is more attached to its initial feature, showing greater resistance to external influence.

(c) **Neighborhood Influence**: This term aggregates features from node i's neighbors. Each neighbor feature $x_j(t)$ is weighted by a time-dependent learnable parameter $w_{ij}(t)$ where $\sum_{j\in\mathcal{N}(i)} w_{ij}(t) = 1$. These adaptive weights allow the model to adjust the influence of each neighbor on node i over time, thereby capturing temporal evolutions.

(d) **Structural Regularization**: This term promotes feature smoothness across the nodes using a Laplacian operator L_g. The learnable coefficient $\mu \in \mathbb{R}_{\geq 0}$ controls the regularization strength, encouraging similar features among connected nodes.

Together, these terms allow GODNF to dynamically balance individual stubbornness with network-level adaptation, enabling nuanced diffusion dynamics over time.

6.2.2.1 Matrix Form

Let $X(t) \in \mathbb{R}^{n\times d}$ denote the feature matrix at time t, with $X(0)$ as the initial feature matrix after neural transformation. Stubbornness is modeled by a diagonal matrix $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$. Neighbor influence is captured by the time-dependent row-stochastic influence matrix $W(t) \in \mathbb{R}^{n\times n}$, where $W_{ij}(t)$ denotes the influence of node j on node i at time step t. Structural regularization is enforced via the normalized Laplacian operator $L_g \in \mathbb{R}^{n\times n}$. With these notations, the update rule can be written in matrix form as:

$$X(0) = f_\theta(H) \tag{6.2}$$

$$X(t+1) = \alpha X(t) + (1-\alpha)\left[\Lambda X(0) + (I-\Lambda)\left(W(t) - \mu L_g\right)X(t)\right]. \tag{6.3}$$

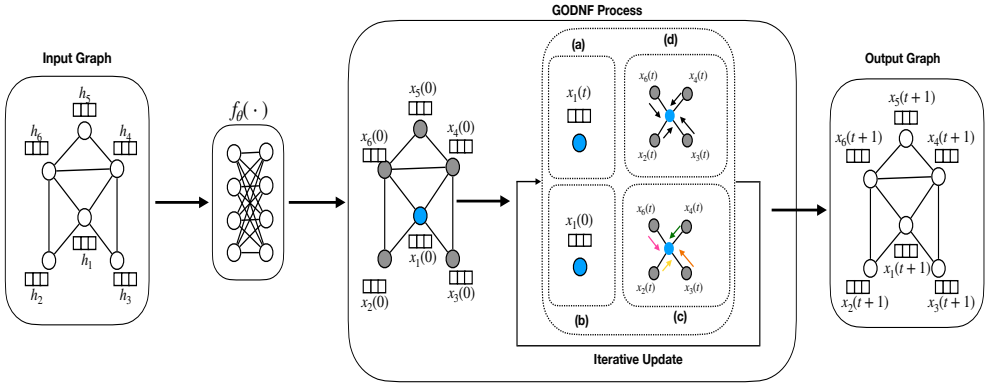


Figure 6.1: High-Level Architecture of GODNF: Raw features are first transformed to retrieve initial features. These initial features are iteratively updated based on four aspects: (a) Current feature retention, (b) Initial feature attachment, (c) Neighborhood influence, and (d) Structural regularization. The model is trained in an end-to-end manner.

6.2.2.2 A Simplified Variant: $\text{GODNF}_{\text{Static}}$

We also introduce a simplified variant of our model where the influence matrix $W(t)$ is constant over time ($W(t) = W$ for all t), meaning the influence between nodes remains fixed throughout propagation. The update rule then simplifies to:

$$X(t+1) = \alpha X(t) + (1-\alpha)\left[\Lambda X(0) + (I-\Lambda)\left(W - \mu L_g\right) X(t)\right]. \tag{6.4}$$

For brevity, we refer to our original model with the time-dependent influence matrix as $\textbf{GODNF}_{\textbf{Dynamic}}$.

6.2.3 Mapping of GODNF Components to Opinion Dynamic Models

Model	Current Feature Retention	Initial Feature Attachment	Dynamic Neighborhood Influence	Structural Regularization
FD	×	×	×	×
FJ	×	✓	×	×
HK	✓	×	✓	×
GODNF_{Static}	✓	✓	×	✓
$\text{GODNF}_{Dynamic}$	✓	✓	✓	✓

Table 6.1: Comparison of Opinion Dynamics Models and GODNF Components. Note that for FD and FJ models, we consider formulations without self-loops, hence no current feature retention. While these models incorporate neighbourhood influence, they utilise static (time-invariant) connection weights and topologies, thus marked as lacking *Dynamic Neighbourhood Influence*. Further, we consider the formulation where HK includes agent's current opinion in bounded confidence averaging.

In Table 6.1, we demonstrate the versatility of GODNF by showing how it general-

izes various opinion dynamics models, aligning its components with the mechanisms of these models.

6.2.4 Bounded Weight Evolution

We bound the time-dependent evolution of influence weights in the $\text{GODNF}_{\text{Dynamic}}$ model to prevent oscillation of $W(t)$, thereby ensuring stability in model training. To achieve this, we employ a decaying update rule:

$$W(t+1) = W(t) + \eta(t)\Delta W(t) \tag{6.5}$$

where $\eta(t)$ is a time-dependent learning rate that decreases over time with $\lim_{t\to\infty} \eta(t) = 0$. $\Delta W(t)$ is the weight adjustment computed from the gradient of the objective function w.r.t. $W(t)$. This ensures progressively smaller updates and eventual stabilization of $W(t)$ to a fixed matrix W^* as t increases.

Remark 3. $\lim_{t\to\infty} \eta(t) = 0$ *implies the convergence of* $W(t)$ *into* W^* *under following conditions: (1)* $||\Delta W(t)|| \leq k$ *for all* t *and some constant* $k > 0$*; (2)* $\sum_{t=0}^{\infty} \eta(t) = \infty$ *and* $\sum_{t=0}^{\infty} \eta(t)^2 < \infty$ *(Robbins and Monro, 1951). Condition (1) is naturally satisfied since* $\Delta W(t)$ *is computed on gradient weights under a well-behaved training loss function. To satisfy condition (2), we select a simple and efficient function* $\eta(t) = \frac{1}{1+t}$.

6.2.5 Convergence Conditions of GODNF

We present the following theorem regarding the convergence of GODNF. It demonstrates that, under given conditions, the node features will converge to a unique fixed point.

Theorem 10 (Convergence of GODNF Feature Updates)**.** *Let* $X(t) \in \mathbb{R}^{n\times d}$ *evolve according to the update rule in Eq.* (6.1)*, and define the combined influence matrix* $M(t) = (I - \Lambda)(W(t) - \mu L_g)$*. When the parameters* Λ *and* μ *are chosen such that the operator norm satisfies* $\|M(t)\|_{\text{op}} < 1$ *for all* t*, and* $W(t) \to W^*$*, the* $X(t)$ *is guaranteed to converge to a unique fixed point* X^* *as* $t \to \infty$*.*

Proof. We work in the Banach space $(\mathbb{R}^{n\times d}, \|\cdot\|)$. $W(t) \to W^*$, which implies $M(t) \to M^*$, where M^* is a fixed matrix. Since $\|M(t)\|_{\text{op}} < 1$ for all t, we have $\|M^*\|_{\text{op}} < 1$. Let X^* satisfy:

$$X^* = \alpha X^* + (1-\alpha)\Lambda X(0) + (1-\alpha)M^* X^*$$

Define $A(t) = \alpha I + (1-\alpha)M(t)$ and $A^* = \alpha I + (1-\alpha)M^*$.
For any t, we have:

$$\|A(t)\|_{\text{op}} \leq \alpha + (1-\alpha)\|M(t)\|_{\text{op}} < \alpha + (1-\alpha) = 1$$

Let $\beta = \sup_t \|A(t)\|_{\text{op}} < 1$. For the error $E(t) = X(t) - X^*$, we have:

$$E(t+1) = A(t)X(t) - A^*X^* + (1-\alpha)\Lambda X(0) - (1-\alpha)\Lambda X(0)$$

$$E(t+1) = A(t)E(t) + (A(t) - A^*)X^*.$$

Taking norms:

$$\|E(t+1)\| \leq \|A(t)\|_{\text{op}}\|E(t)\| + \|A(t) - A^*\|_{\text{op}}\|X^*\|$$

$$\|E(t+1)\| \leq \beta\|E(t)\| + (1-\alpha)\|M(t) - M^*\|_{\text{op}}\|X^*\|$$

Since $M(t) \to M^*$, for any $\tau > 0$, there exists T such that for all $t \geq T$, $\|M(t) - M^*\|_{\text{op}} < \frac{\tau}{(1-\alpha)\|X^*\|}$ (assuming $X^* \neq 0$).
For $t \geq T$:

$$\|E(t+1)\| \leq \beta\|E(t)\| + \tau$$

This is a contraction mapping with a small perturbation, as $t \to \infty$ and $\tau \to 0$. Thus, according to the Banach fixed point theorem (Mannan et al., 2021), $X(t)$ converges to X^*. □

Remark 4. *The condition $\|M(t)\|_{\text{op}} < 1$ is realistic and not overly restrictive as $W(t)$ (row-stochastic with eigenvalues in $[-1, 1]$) and L_g (normalized Laplacian with eigenvalues in $[0, 2]$) can be appropriately scaled by learnable parameters Λ and μ to satisfy this bound regardless of graph structure.*

6.2.6 Convergence-Preserving Regularization

According to Theorem 10, convergence of GODNF requires $\|M(t)\|_{\text{op}} < 1$ at every time step. Therefore, we design a regularization term enforcing this condition. However, computing $\|M(t)\|_{\text{op}}$ can be expensive in practice. To address this, we employ an efficient approximation based on a well-established matrix norm inequality. The operator norm of a matrix is bounded above by the geometric mean of its 1-norm and infinity-norm (Horn and Johnson, 2012),

$$\|M(t)\|_{\text{op}} \leq \sqrt{\|M(t)\|_1 \cdot \|M(t)\|_\infty}, \tag{6.6}$$

Unlike the op-norm, the infinity-norm and 1-norm can be computed in linear time.

6.2.6.1 Regularization Term

We design a regularization term to enforce the condition $\sqrt{\|M(t)\|_1 \cdot \|M(t)\|_\infty} < 1$, penalizing violations when this condition is not satisfied. The regularization term is:

$$\mathcal{L}_{reg} = \sum_{t=0}^{T} \max\left(0, \sqrt{\|M(t)\|_1 \cdot \|M(t)\|_\infty} - 1\right). \tag{6.7}$$

In $\text{GODNF}_{\text{static}}$ variant, since $W(t) = W$ is constant, which makes $M(t) = M$ time invariant, the regularization term can be simplified to:

$$\mathcal{L}_{reg} = \max\left(0, \sqrt{\|M\|_1 \cdot \|M\|_\infty} - 1\right). \tag{6.8}$$

This eliminates the need to sum over time steps, reducing computational complexity and making $\text{GODNF}_{\text{static}}$ more efficient than its dynamic counterpart.

6.2.6.2 Training Loss

We define training loss of GODNF as a combination of downstream task loss and the regularization term as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \mathcal{L}_{reg}. \tag{6.9}$$

6.3 Theoretical Analysis

6.3.1 Complexity Analysis

The time complexity of GODNF comprises the initialization and iterative update phases. Initial feature tranformation costs $\mathcal{O}(n \cdot p_f)$, where p_f is the upper bound on the cost of computing $f_\theta(h_i)$. The complexity of each iteration incurred by GODNF components is: initial feature attachment $\mathcal{O}(n \cdot d)$, neighborhood influence and structural regularization both $\mathcal{O}(m \cdot d)$ through sparse matrix multiplication, and norm regularization term $\mathcal{O}(m)$, which is efficient due to the sparsity of $M(t)$. Here, d is the feature dimension. The total time complexity over T iterations is $\mathcal{O}(n \cdot p_f + T \cdot (m \cdot d + n \cdot d + m)) = \mathcal{O}(n \cdot p_f + T \cdot m \cdot d)$, using the graph connectivity assumption. In practical scenarios where p_f, d, and T are usually small, this complexity is bounded by $\mathcal{O}(m)$, which is asymptotically optimal.

The space complexity of GODNF includes storing feature matrices $X(0)$ and $X(t)$ with $\mathcal{O}(n \cdot d)$, the sparse matrices $W(t)$ and L_g with $\mathcal{O}(m)$, and the diagonal matrix Λ with $\mathcal{O}(n)$. Overall, the space complexity is $\mathcal{O}(n \cdot d + m)$.

Overall, GODNF achieves comparable complexity to traditional naive GNNs such as GCN (Kipf and Welling, 2017). The absence of intermediate feature transformations in GODNF can lead to computational efficiency gains, especially for deeper layers.

6.3.2 Representation of Diverse Convergence Configurations

In this section, we demonstrate that GODNF can capture multiple convergence configurations, emphasizing its adaptability to diverse learning scenarios. We start by defining these convergence configurations and show that GODNF can theoretically converge to these diverse configurations by adjusting its components. For the following theorems, we consider the following conditions of GODNF to hold: (1) $W^t \to W^*$, (2) $\|M(t)\|_{\text{op}} < 1$ for all t where $M(t) = (I - \Lambda)(W(t) - \mu L_g)$.

6.3.2.1 Single Consensus

Definition 13. *A diffusion model is said to reach a* single consensus *if there exists a vector* $z \in \mathbb{R}^d$ *such that, for all nodes* $i \in V$*,*

$$\lim_{t\to\infty} x_i(t) = z,$$

We now establish conditions under which GODNF converges to a single consensus.

Theorem 11 (Single Consensus under Fully Diffusive Dynamics)**.** *GODNF reaches a single consensus if* $\Lambda = 0$*,* $\mu = 0$*,* $\alpha = 0$*, and* W^* *is a row-stochastic matrix.*

Single consensus is valuable for modeling linear diffusion processes on networks, such as heat diffusion and basic opinion dynamics, as it captures the convergence of these processes.

Proof. With these parameter configurations, the update rule of GODNF becomes $X(t+1) = W^*X(t)$. Given that W^* is a row-stochastic matrix, this would be equivalent to the FD model. As proven in DeGroot et al. (DeGroot, 1974), FD model converges to single consensus. Therefore, GODNF will exhibit the same behavior. □

6.3.2.2 Multi Consensus

Definition 14. *A diffusion model is said to reach a* multi consensus *if there exist distinct vectors* $v_1, v_2, \ldots, v_k \in \mathbb{R}^d$ *with* $k < n$ *such that, for all nodes* $i \in V$*,*

$$\lim_{t\to\infty} x_i(t) = v_j, \quad j \in \{1, 2, \ldots, k\}.$$

We now present the conditions under which GODNF converges to a multi consensus.

Theorem 12 (Multi-Consensus under Block-Structured Dynamics)**.** *GODNF reaches multi consensus if the matrix* $M^* = (1-\Lambda)(W^* - \mu L_g)$ *has a block diagonal structure corresponding to multiple disconnected components, with initial conditions and parameters differ for at least two components.*

Proof. The fixed point of GODNF can be written as,

$$X^* = (I - M^*)^{-1}\Lambda X(0)$$

Since $||M(t)||_{\text{op}} < 1$ for all t, $||M^*||_{\text{op}} < 1$ and $(I - M^*)$ invertible. Therefore, the existence of the above fixed point is guaranteed. Let $B^* = (I - M^*)^{-1}$, and $X^* = B^*\Lambda X(0)$. Given M^* has block structure, we can write B^* as follows:

$$B^* = \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_k \end{bmatrix}$$

This leads to set of independent components $X_1^* = B_1^*\Lambda_1 X_1(0)$, $X_2^* = B_2^*\Lambda_2 X_2(0)$, $\dots$ $X_k^* = B_k^*\Lambda_k X_k(0)$. Each component reaches its own consensus independently, leading to multiple consensuses when initial conditions and parameters differ for at least two components. □

The multi consensus is beneficial for multi-label classification as it can avoid the oversmoothing problem in traditional GNNs. GODNF can preserve multiple clusters rather than converging to a single value, effectively aligning node features with class boundaries and handling both homophilic and heterophilic scenarios.

6.3.2.3 Individualized Consensus

Definition 15. *A diffusion model is said to reach an* individualized consensus *if there exist distinct vectors $v_1, v_2, \dots, v_n \in \mathbb{R}^d$ such that, for all nodes $i \in V$,*

$$\lim_{t\to\infty} x_i(t) = v_i, \quad i \in \{1, 2, \dots, n\}, \text{ with } v_i \neq v_j \text{ for } i \neq j.$$

We next look into the conditions that lead GODNF to reach individualized consensus, where each node converges to a unique value.

Theorem 13 (Individualized Consensus under Strong Feature Retention)**.** *GODNF reaches individualized consensus if the initial features $x_i(0)$ are distinct for all nodes $i \in V$, and $\lambda_i \in (0, 1]$ are sufficiently large (i.e., close to 1).*

Individualized consensus is beneficial for node-level prediction tasks like regression. Unlike standard GNNs that oversmooth features, this consensus maintains distinct node features while still capturing network structure, offering adaptability to diverse graph settings where preserving node identity is essential.

Proof. The fixed point of GODNF is,

$$X^* = (I - M^*)^{-1}\Lambda X(0)$$

where $M^* = (1 - \Lambda)(W^* - \mu L_g)$. Since $||M(t)||_{\text{op}} < 1$ for all t, $||M^*||_{\text{op}} < 1$ and $(I - M^*)$ invertible. This ensures the existence of the above fixed point. When λ_i approaches 1 for all nodes, $M \approx 0$, giving:

$$X^* \approx (I - 0)^{-1}\Lambda X(0) = \Lambda X(0)$$

Since Λ has positive diagonal entries close to 1, each node's convergence value closely preserves its initial feature. With $X(0)$ containing distinct initial features, this naturally leads to individualized consensus. □

Remark 5. *In addition to the three fundamental convergence configurations above, GODNF can adapt to more complex convergence patterns due to its unification of diverse opinion dynamics mechanisms. This flexibility enables GODNF to align with sophisticated convergence configurations that capture the nuances present in the ground truth. In practice, the*

FD-inspired setting is appropriate when a single consensus across the graph is expected. The FJ-inspired setting suits tasks where nodes should retain individual characteristics, such as heterophilic node classification. The HK-inspired setting is better suited to tasks with dynamic neighbourhood influence such as influence estimation. The model learns these behaviours from data, but domain knowledge about the expected convergence behaviour can guide initialisation.

6.4 Experimental Design

We evaluate GODNF on two types of tasks: node classification and influence estimation. The goal of the node classification task is to predict the label of individual nodes in a graph. The influence estimation is a node regression task that focuses on determining a node's susceptibility to activation given a specific diffusion model and an initial set of activated nodes (Xia et al., 2021). Although node classification operates as a static downstream task, influence estimation requires capturing the temporal dynamics of the diffusion process to learn the underlying propagation characteristics accurately.

6.4.1 Datasets

For node classification, we use ten benchmark datasets that include both homophily and heterophily label patterns. For homophily datasets, we use the Cora-ML, CiteSeer, PubMed, DBLP, and Cora Full datasets from the CitationFull benchmark (Bojchevski and Günnemann, 2018). For heterophily datasets, we include Texas, Cornell, Wisconsin, and Film datasets from WebKB (Pei et al., 2020b), and the Amazon-rating dataset from the Heterophilous Graph benchmark (Platonov et al., 2023). Further, we employ the OGB benchmark's ogbn-arxiv dataset (Hu et al., 2020) for our scalability analysis. For influence estimation, we use four real-world datasets: Jazz, Network Science, Power Grid (Rossi and Ahmed, 2015), and Cora-ML (Bojchevski and Günnemann, 2018).

Dataset statistics for node classification and node regression tasks are depicted in table 6.2 and table 6.3, respectively.

Type	**Dataset**	**Homophily Level**	**# Nodes**	**# Edges**	**# Classes**
Heterophily	Texas	0.06	183	309	5
Heterophily	Cornell	0.12	183	295	5
Heterophily	Wisconsin	0.18	251	499	5
Heterophily	Film	0.22	7,600	33,544	5
Heterophily	Amazon-rating	0.38	24,492	186,100	5
Homophily	Cora Full	0.57	19,793	126,842	70
Homophily	Citeseer	0.74	3,312	4,732	6
Homophily	Cora-ML	0.79	2,995	16,316	7
Homophily	PubMed	0.80	19,717	44,338	3
Homophily	DBLP	0.83	17,716	105,734	4

Table 6.2: Dataset statistics for node classification

Dataset	# Nodes	# Edges
Jazz	198	2,742
Cora-ML	2,810	7,981
Network Science	1,565	13,532
Power Grid	4,941	6,594

Table 6.3: Dataset statistics for influence estimation task

6.4.2 Baselines

For node classification, we compare GODNF against both spatial and diffusion-based GNNs. Our spatial GNN baselines consist of three classical architectures: GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017), as well as three advanced models: AirGNN (Liu et al., 2021a), DirGNN (Rossi et al., 2024), and BEC-GCN (Hevapathige et al., 2025b). Diffusion GNNs include both discrete and continuous variants, spanning classical and state-of-the-art models such as APPNP (Gasteiger et al., 2018), GDC (Gasteiger et al., 2019), ODNET (Zhou et al., 2024), GRAND (Chamberlain et al., 2021), GRAND++ (Thorpe et al., 2022b), ACMP (Wang et al., 2022a), HiD-Net (Li et al., 2024b), and GNRF (Chen et al., 2025). AirGNN and APPNP have residual connections in their architectures that are conceptually similar to the FJ model, while ODNet is designed based on the HK model. For influence estimation, we use a comprehensive set of aforementioned general-purpose baselines that span spatial and diffusion GNNs and GNNs specifically designed for the influence estimation task. These specialized models include SGNN (Kumar et al., 2022), DeepIM (Ling et al., 2023), DeepIS (Xia et al., 2021), and GLIE (Panagopoulos et al., 2023). Additionally, we tailor UniGO (Li et al., 2025) as a baseline for influence estimation.

6.4.3 Evaluation settings

For node classification, we use a 60:20:20 random split strategy for the training, validation, and test sets and report the mean and standard deviation of accuracy over 10 random initializations, following the experimental setup of Chen et al. (Chen et al., 2025). We evaluate influence estimation performance under three diffusion models: Linear Threshold (LT) (Kempe et al., 2003), Independent Cascade (IC) (Goldenberg et al., 2001a), and Susceptible-Infected-Susceptible (SIS) (Kimura et al., 2009). LT and IC are progressive diffusion models (i.e., nodes activate once and remain activated permanently), whereas SIS is non-progressive (i.e., nodes can undergo multiple activation cycles) (Li et al., 2023a). For data splits, we follow the experimental setup of Ling et al. (2023), with an initial activation node set consisting of 10% of the total nodes in the graph. For evaluation, we employ 10-fold cross-validation and report the mean and standard deviation of Mean Absolute Error (MAE). For baseline results, we report the results from previous papers that used the same experimental setup. When such results are unavailable, we generate baseline results by adopting the hyperparameter configurations specified in the original papers.

6.4.4 Model Hyperparameters

In our experiments, we employ a Grid search to find optimal hyperparameters of GODNF in the following ranges: number of layers $\in \{2, 3, 4\}$, learning rate $\in \{5e-3, 1e-2\}$, weight decay $\in \{5e-4, 1e-1\}$, dropout $\in \{0.4, 0.5\}$, and the hidden dimension size $\in \{32, 64, 256\}$. GODNF model hyperparameter, α is chosen from {0.1, 0.3, 0.6, 0.9}. Note that Λ, and μ parameters in our model are learned from the ground truth. In node classification, we train models for up to 1000 epochs, while for influence estimation, we use 200 epochs. We utilize the Adam algorithm (Kingma, 2015) as the optimizer.

6.4.5 Implementation Details

All experiments were conducted on a Linux server with an Intel Xeon W-2175 2.50 GHz processor, comprising 28 cores, an NVIDIA RTX A6000 GPU, and 512 GB of RAM. We use the following Python libraries for our implementation: PyTorch version 2.3.1, torchvision 0.18.1, torchaudio 2.3.1, torch-geometric 2.7.0, torch-cluster 1.6.3, torch-scatter 2.0.9, and torch_sparse 0.6.18.

6.5 Results and Discussion

6.5.1 Node Classification

	Texas	Cornell	Wisconsin	Film	Amazon-rating
GCN	60.00±6.45	55.14±8.46	61.60±7.00	32.16±1.34	37.99±0.61
GAT	61.21±8.17	53.64±11.10	60.00±11.00	32.63±1.61	42.52±1.22
GraphSAGE	82.46±5.19	75.32±5.96	80.00±4.81	36.00±1.20	41.32±0.63
AirGNN	65.57±11.19	53.19±10.43	56.25±5.03	36.34±2.73	42.29±0.88
DirGNN	76.25±6.31	76.51±6.14	80.50±5.50	35.76±1.68	46.66±0.61
BEC-GCN	68.85±11.02	65.96±9.03	66.25±4.55	34.34±2.08	42.17±0.58
APPNP	62.30±6.65	58.75±5.97	61.70±6.94	36.44±1.78	43.48±0.68
GDC	72.13±5.48	53.19±12.19	62.50±5.13	30.65±1.20	40.06±2.50
ODNet	85.25±8.36	86.49±4.06	88.75±2.15	37.57±0.94	43.33±0.91
GRAND	81.70±8.42	81.76±13.90	84.00±7.50	27.93±1.25	37.53±0.36
GRAND++	79.34±7.22	81.34±7.12	81.50±6.00	36.32±0.50	38.01±0.50
ACMP	87.65±3.54	85.66±5.10	86.50±5.00	34.44±1.36	37.32±0.64
HiD-Net	77.11±6.91	83.33±7.10	81.30±6.60	28.86±1.00	41.19±1.03
GNRF	87.39±4.13	87.28±3.12	88.00±2.00	34.22±1.40	46.89±1.08
$\text{GODNF}_{\text{Static}}$	92.95±2.32	87.23±4.66	91.62±3.49	40.21±1.85	49.17±1.06
$\text{GODNF}_{\text{Dynamic}}$	93.93±2.65	87.87±4.57	92.50±2.56	39.96±2.21	49.40±0.44

Table 6.4: Node classification accuracy ± standard deviation (%) on heterophilic datasets. The best and second-best results are highlighted in blue and orange, respectively. The baseline results are sourced from Chen et al. (2025).

	Cora Full	Citeseer	Cora-ML	PubMed	DBLP
GCN	68.06±0.98	77.72±1.18	87.07±1.21	86.74±0.47	83.93±0.84
GAT	67.55±1.23	76.79±1.00	84.12±0.55	87.24±0.55	80.61±1.21
GraphSAGE	69.77±0.46	77.42±0.83	86.52±1.32	84.50±0.39	86.16±0.50
AirGNN	65.27±0.72	78.20±1.17	88.09±1.25	83.51±0.87	84.69±0.30
DirGNN	67.80±0.53	77.71±0.78	85.66±0.31	86.94±0.55	81.22±0.54
BEC-GCN	71.34±0.44	78.14±1.31	88.65±1.32	87.00±0.28	85.20±0.48
APPNP	70.63±0.51	77.89±1.07	87.41±1.54	83.37±0.42	85.36±0.40
GDC	62.23±0.59	78.17±1.11	84.41±1.59	80.61±0.47	84.87±0.40
ODNet	70.23±0.43	74.76±1.15	76.26±1.03	86.74±0.37	83.45±0.47
GRAND	67.66±1.01	76.55±1.69	88.49±0.81	86.79±0.57	84.60±0.99
GRAND++	67.53±0.74	78.51±1.58	88.44±0.53	87.21±0.33	85.21±0.24
ACMP	71.76±0.03	75.73±1.38	76.11±2.12	88.01±1.44	82.31±0.44
HiD-Net	68.11±0.64	77.85±0.65	89.00±0.51	88.60±0.45	84.92±0.31
GNRF	72.12±0.50	75.79±0.94	89.18±0.19	90.37±0.69	85.73±0.76
$\text{GODNF}_{\text{Static}}$	72.60±0.58	78.83±1.22	88.85±1.52	90.66±0.47	88.24±2.01
$\text{GODNF}_{\text{Dynamic}}$	72.51±0.53	78.85±0.99	89.26±1.13	90.78±0.68	88.62±1.92

Table 6.5: Node classification accuracy ± standard deviation (%) on homophilic datasets. The best and second-best results are highlighted in blue and orange, respectively. The baseline results are sourced from Chen et al. (2025).

We present node classification results on heterophilic datasets in Table 6.4 and homophilic datasets in Table 6.5. GODNF consistently outperforms baseline models in both heterophily and homophily datasets. In particular, GODNF demonstrates significant superiority over spatial and discrete diffusion GNNs, especially within heterophily datasets. Additionally, GODNF comfortably surpasses continuous diffusion GNNs by exceeding their expressive power. We credit GODNF's success to its capacity to model complex diffusion dynamics in graphs, which is enhanced by its representation power derived from multiple opinion dynamic components. The performance of models such as ODNET, APPNP, and AirGNN further supports this point. Although these models incorporate elements related to opinion dynamics, they do not achieve the same level of performance as GODNF, primarily because their components lack the representation capabilities found in GODNF. Further, we observe that the dynamic variant of GODNF performs better than the static variant because it captures temporal dependencies between diffusion time steps.

6.5.2 Influence Estimation

We report influence estimation results across three diffusion models (IC, LT, and SIS) in Tables 6.6, 6.7, and 6.8, respectively. GODNF outperforms or provides comparable performance to both general-purpose spatial and diffusion GNNs, as well as GNNs specifically designed for influence estimation. While DeepIS and GLIE excel in the IC model due to their tailored update rules, they exhibit suboptimal performance within other diffusion models. In contrast, GODNF shows robust generalizability, consistently maintaining better performance across all diffusion models.

Additionally, $\text{GODNF}_{\text{dynamic}}$ consistently outperforms $\text{GODNF}_{\text{static}}$ because the temporal dynamics captured by the dynamic variant are beneficial for tasks like influence estimation, which rely on temporal characteristics.

	Jazz	Cora-ML	Network Science	Power Grid
GCN	0.233±0.010	0.277±0.007	0.270±0.019	0.313±0.024
GAT	0.342±0.005	0.352±0.004	0.274±0.002	0.331±0.002
GraphSAGE	0.201±0.028	0.255±0.010	0.241±0.010	0.313±0.024
DirGNN	0.205±0.016	0.255±0.006	0.236±0.011	0.287±0.010
SGNN	0.183±0.004	0.210±0.003	0.213±0.003	0.257±0.002
UniGO	0.192±0.013	0.255±0.001	0.201±0.001	0.231±0.002
DeepIS	0.151±0.003	0.203±0.001	0.223±0.001	0.206±0.001
DeepIM	0.178±0.002	0.210±0.002	0.216±0.003	0.258±0.003
GLIE	0.136±0.003	0.199±0.041	0.163±0.031	0.183±0.009
APPNP	0.200±0.006	0.265±0.022	0.248±0.006	0.290±0.006
ODNet	0.180±0.003	0.232±0.001	0.255±0.002	0.274±0.001
HIDNet	0.216±0.003	0.258±0.001	0.242±0.002	0.282±0.001
GNRF	0.195±0.006	0.255±0.005	0.294±0.005	0.280±0.003
$\text{GODNF}_{\text{Static}}$	0.177±0.005	0.150±0.001	0.205±0.002	0.223±0.002
$\text{GODNF}_{\text{Dynamic}}$	0.175±0.003	0.148±0.001	0.197±0.002	0.221±0.001

Table 6.6: Influence estimation (IC model) mean absolute error ± standard deviation. Lower value indicates better performance. The best and second-best results are highlighted in blue and orange, respectively.

	Jazz	Cora-ML	Network Science	Power Grid
GCN	0.199±0.006	0.255±0.008	0.190±0.012	0.335±0.023
GAT	0.156±0.100	0.192±0.010	0.114±0.008	0.280±0.015
GraphSAGE	0.120±0.004	0.203±0.019	0.112±0.005	0.341±0.018
DirGNN	0.124±0.002	0.193±0.012	0.084±0.017	0.257±0.062
SGNN	0.164±0.014	0.134±0.004	0.049±0.004	0.192±0.001
UniGO	0.159±0.020	0.155±0.019	0.110±0.049	0.235±0.005
DeepIS	0.219±0.002	0.301±0.005	0.306±0.001	0.374±0.001
DeepIM	0.134±0.014	0.271±0.010	0.118±0.003	0.331±0.002
GLIE	0.055±0.028	0.286±0.016	0.160±0.063	0.384±0.020
APPNP	0.124±0.003	0.220±0.031	0.084±0.006	0.189±0.011
ODNet	0.053±0.003	0.210±0.004	0.104±0.015	0.296±0.002
HIDNet	0.180±0.003	0.273±0.006	0.166±0.005	0.295±0.007
GNRF	0.150±0.020	0.312±0.029	0.293±0.001	0.459±0.019
$\text{GODNF}_{\text{Static}}$	0.051±0.002	0.104±0.005	0.036±0.002	0.116±0.003
$\text{GODNF}_{\text{Dynamic}}$	0.050±0.001	0.094±0.005	0.032±0.003	0.108±0.004

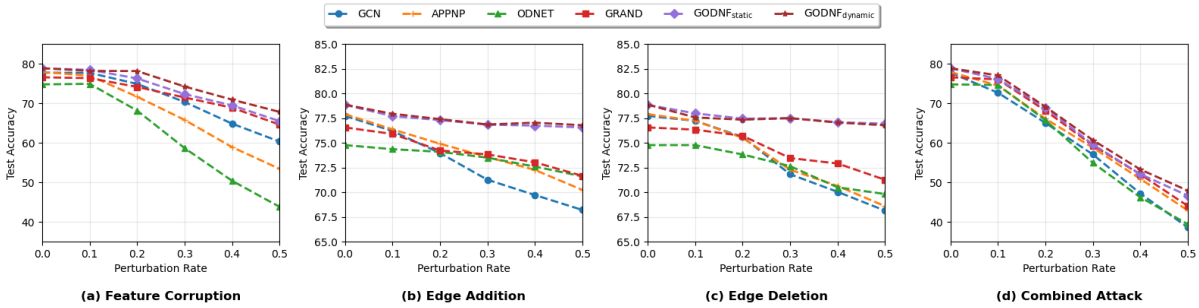
Table 6.7: Influence estimation (LT model) mean absolute error ± standard deviation. Lower value indicates better performance. The best and second-best results are highlighted in blue and orange, respectively.

6.5.3 Robustness to adversarial attacks

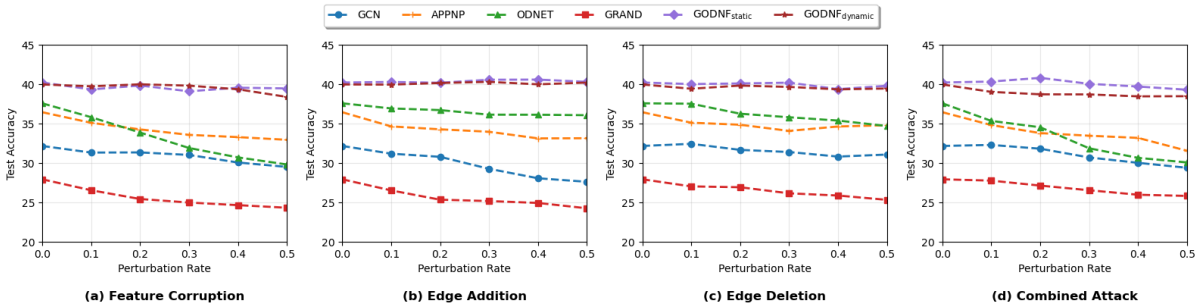
We evaluate the robustness of GODNF under a diverse set of adversarial attack scenarios, benchmarking its resilience against four distinct attack strategies and com-

	Jazz	Cora-ML	Network Science	Power Grid
GCN	0.344±0.023	0.365±0.065	0.180±0.007	0.207±0.001
GAT	0.288±0.017	0.208±0.008	0.123±0.013	0.133±0.001
GraphSAGE	0.301±0.018	0.222±0.051	0.102±0.005	0.133±0.001
DirGNN	0.379±0.032	0.229±0.025	0.142±0.019	0.132±0.001
SGNN	0.330±0.007	0.211±0.006	0.127±0.00	0.175±0.004
UniGO	0.335±0.002	0.212±0.001	0.108±0.023	0.206±0.023
DeepIS	0.434±0.003	0.304±0.001	0.256±0.001	0.251±0.001
DeepIM	0.383±0.010	0.270±0.006	0.135±0.004	0.205±0.005
GLIE	0.454±0.062	0.205±0.029	0.103±0.023	0.132±0.026
APPNP	0.357±0.059	0.321±0.042	0.100±0.012	0.132±0.001
ODNet	0.322±0.006	0.196±0.002	0.106±0.005	0.145±0.001
HIDNet	0.404±0.005	0.360±0.003	0.168±0.001	0.224±0.001
GNRF	0.444±0.024	0.280±0.027	0.186±0.006	0.196±0.018
$\text{GODNF}_{\text{Static}}$	0.318±0.004	0.197±0.001	0.098±0.002	0.120±0.002
$\text{GODNF}_{\text{Dynamic}}$	0.309±0.006	0.192±0.002	0.096±0.001	0.119±0.002

Table 6.8: Influence estimation (SIS model) mean absolute error ± standard deviation. Lower value indicates better performance. The best and second-best results are highlighted in blue and orange, respectively.



((a)) Citeseer dataset



((b)) Film dataset

Figure 6.2: Node classification performance under different adversarial attacks.

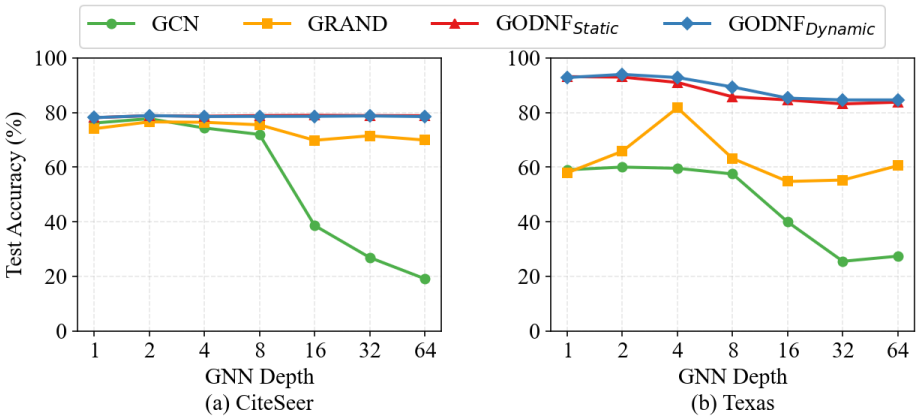


Figure 6.3: Deep layer comparison for node classification

paring it with established baselines. In the feature corruption setting, Gaussian noise is injected into node features by sampling a noise matrix and adding it to the original features, following Li et al. (2024b). The edge addition attack perturbs the topology by inserting random edges (excluding self-loops and duplicates) at a specified perturbation rate, while edge deletion removes randomly selected edges. The combined attack applies edge deletion, edge addition, and feature corruption sequentially at the same perturbation rate. The results are depicted in Figure 6.2.

GODNF exhibits consistent and superior robustness across all adversarial attack types. Under feature corruption, it shows only gradual performance degradation versus steep drops in baselines. For structural attacks such as edge addition or deletion, GODNF variants maintain near-constant performance while baselines degrade significantly. Combined attacks further accentuate GODNF's resilience. This robustness stems from GODNF's multi-component defence mechanism, creating redundant information pathways. The initial feature attachment acts as persistent memory, structural regularization provides graph-based smoothing, and bounded weight evolution ensures stability. When attacks compromise one component, the others compensate, thereby preventing the single-pathway failures common in standard GNNs.

6.5.4 Oversmoothing Analysis

We evaluate the performance of GODNF in a deep layer setting to understand its resilience to oversmoothing. The results are depicted in Figure 6.3. Unlike traditional GNNs like GCN and GRAND, which show substantial performance degradation when the layer depth is increased, both GODNF variants show robustness against oversmoothing by maintaining consistent performance across varying layer depths.

6.5.5 Impact of Different Components

We conduct an ablation study to evaluate the importance of each component of GODNF on its performance. To demonstrate this, we derive variants of GODNF

by omitting each component and assess their performance. The results are depicted in Figure 6.4.

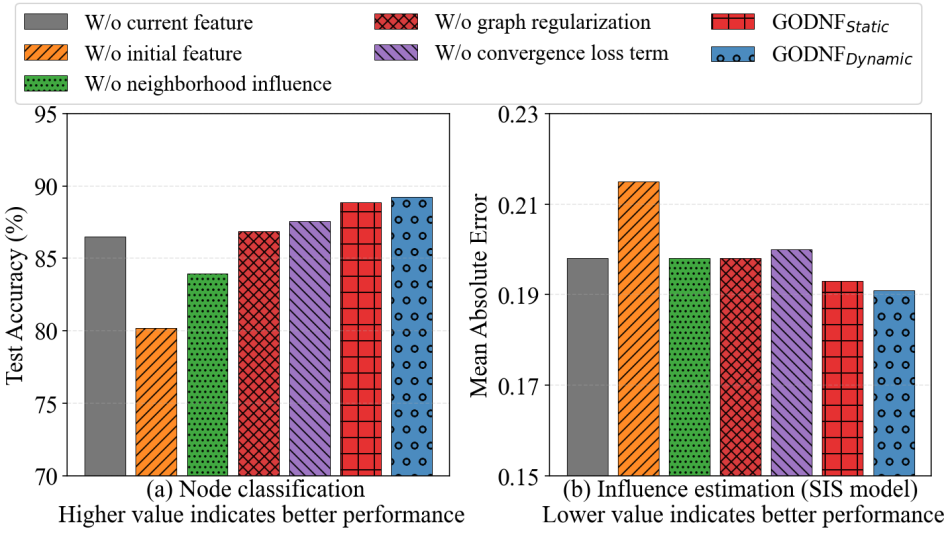


Figure 6.4: Performance comparison of GODNF component variants on Cora ML dataset

As anticipated, removing each component decreases performance, demonstrating their individual importance. Further, we observe that the initial feature attachment has more impact on performance than other components. We believe this is due to complex node-specific behaviors captured by that component.

6.5.6 Parameter Analysis

We evaluate the sensitivity of GODNF to its hyperparameter α in Eq. (6.1), by varying its value from 0 to 0.9. In Figure 6.5, we observe that the performance of homophilic datasets (Cora ML and Coral Full) in node classification tends to decrease slowly as α increases, while the opposite behavior can be seen in heterophilic datasets (Film and Texas). This is because a decrease of α promotes neighborhood information aggregation, which helps homophily datasets where neighbors tend to share the same label, but adversely impacts heterophilic datasets, where neighbors tend to have different labels.

Then, we present the distribution of learned node stubbernness λ for datasets under node classification in Figure 6.6. The results reveal few interesting observations. First, most nodes exhibit moderate stubbornness (0.4-0.7), highlighting the importance of balancing initial features with neighborhood information. Second, the distribution variability shows adaptive learning of heterogeneous aggregation requirements across nodes. Third, heterophily datasets tend to yield higher stubbornness values compared to homophily datasets, suggesting that when neighbors have dissimilar labels, nodes benefit from retaining their initial representations to mitigate information loss from inappropriate aggregation.

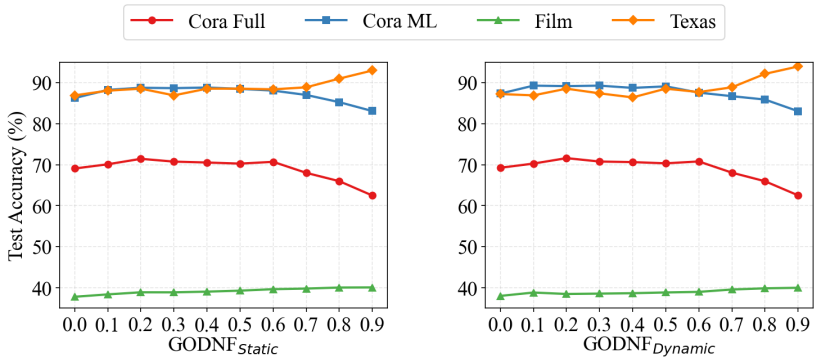
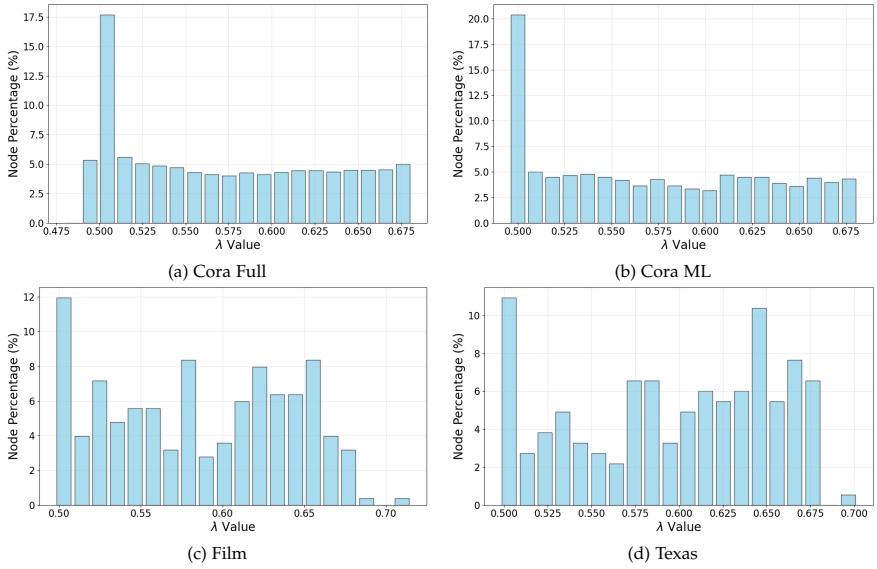


Figure 6.5: Sensitivity analysis of α

We further observe that the learned value for μ is in the range of 0.7-0.9 for most datasets. We believe this range offers essential structural regularisation for convergence stability.



(a) Cora Full (b) Cora ML

(c) Film (d) Texas

Figure 6.6: Distribution of learned node stubbornness λ_i

6.5.7 Scalability Analysis

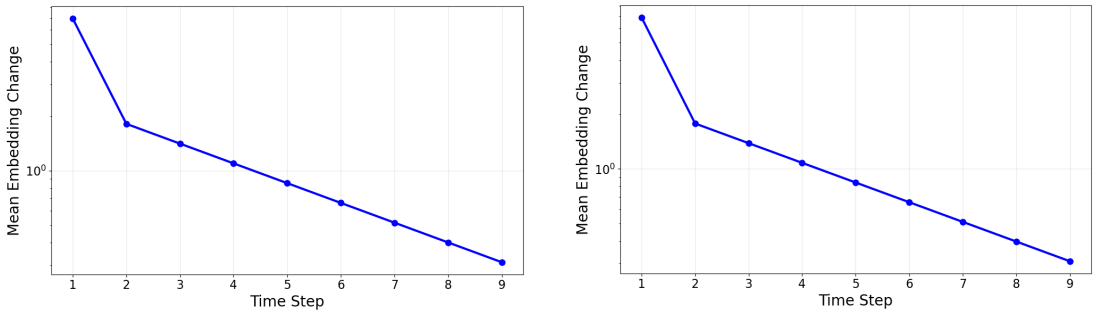
We compare the scalability of our approach with existing GNNs in Table 6.9. For this, we perform node classification task and employ two datasets: Cora Full and ogbn-arxiv. ogbn-arxiv is a large-scale dataset with 169,343 nodes and 1,166,243 edges. For a fair comparison, we fix the hidden layer size at 128 for all GNNs, and record average runtime per epoch, number of learnable parameters, and accuracy metrics.

As observed, GODNF is significantly more effective than traditional GNNs like GAT and diffusion GNNs like GRAND and ACMP. Furthermore, GODNF has a runtime and parameter complexity comparable to that of highly scalable traditional

Layers	Model	Cora Full			ogbn-arxiv		
		# Param (k)	Time (ms)	Acc. (%)	# Param (k)	Time (ms)	Acc. (%)
4	GCN	1157	20.6	68.44	55	154.2	69.53
	GAT	4631	65.3	59.87	206	OOM	OOM
	GRAND	716	111.1	69.27	81	2077.0	69.30
	ACMP	1158	1107.6	70.94	55	OOM	OOM
	$\text{GODNF}_{\text{Static}}$	1144	31.7	72.42	191	926.4	69.76
	$\text{GODNF}_{\text{Dynamic}}$	1144	35.5	72.58	191	936.9	70.43
8	GCN	1223	34.6	65.87	123	327.2	68.39
	GAT	4897	121.3	31.10	474	OOM	OOM
	GRAND	716	150.7	69.81	81	3196.5	67.24
	ACMP	1158	2231.5	71.76	55	OOM	OOM
	$\text{GODNF}_{\text{Static}}$	1144	51.1	72.42	191	2156.1	70.23
	$\text{GODNF}_{\text{Dynamic}}$	1144	63.4	72.49	191	2378.4	70.91

Table 6.9: Scalability comparison for node classification. Best results are highlighted in blue. OOM refers to out-of-memory.

GNNs like GCN. This efficiency primarily comes from eliminating the learnable transformation layers found in conventional GNNs, resulting in less parameter count and memory usage, even at increasing depths.



((a)) $\text{GODNF}_{\text{Static}}$ ((b)) $\text{GODNF}_{\text{Dynamic}}$

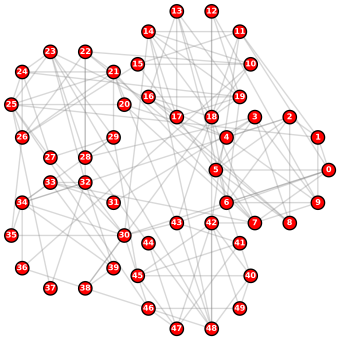
Figure 6.7: Step-wise embedding changes of GODNF over time steps for the Cora Full for node classification.

6.5.8 Case Study

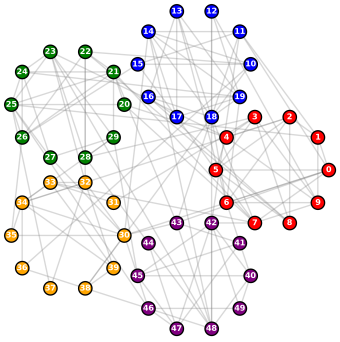
We conduct experiments to empirically validate the convergence properties of GODNF, performing node classification tasks on the Cora Full dataset. Figure 6.7 shows the convergence behaviour of GODNF variants through step-wise average embedding changes over evolution time. The exponential decay pattern demonstrates that both GODNF variants achieve stable convergence, with average embedding changes decreasing rapidly from initial large adjustments to minimal fluctuations by step 7-8, confirming the theoretical convergence guarantees presented in Theorem 10.

Next, we empirically validate GODNF's ability to depict multiple convergence configurations. Specifically, we create a stochastic block model (SBM) graph (Abbe, 2018) (50 nodes and five communities) and generate node labels to be aligned with

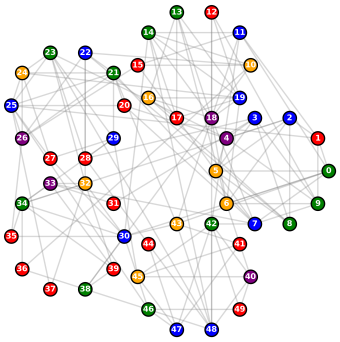
different convergence configurations, as described in Section 6.3.2, and train our model to learn these label patterns. In a multi-consensus scenario, we examine homophily and heterophily label patterns separately. The visualization of color-coded node labels learned by our model for each convergence configuration is depicted in Figure 6.8.



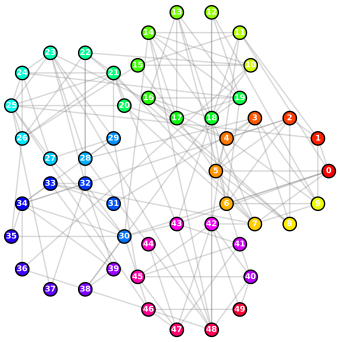
(a) Single Consensus



(b) Multi Consensus - Homophily Pattern



(c) Multi Consensus - Heterophily Pattern



(d) Individualized Consensus

Figure 6.8: Visualisation of colour-coded node labels learned by GODNF for different convergence configurations.

In the single consensus (Figure 5a), GODNF learns to assign identical labels across all nodes. In the multi-consensus scenario with homophily labels (Figure 5b), our model correctly identifies the pattern with neighbors having similar labels. In contrast, the multi-consensus configuration with heterophily labels (Figure 5c) yields more intricate patterns, with neighboring nodes exhibiting markedly different labels. Finally, the individualized consensus (Figure 5d) shows the highest color diversity, capturing the personalized nature of node consensus.

6.6 Summary

In this chapter, we proposed a novel neural diffusion framework based on opinion dynamics. Our method generalizes multiple opinion dynamic models, providing

an efficient and expressive solution that advances existing GNNs. We provided a comprehensive theoretical analysis of our framework's convergence properties. The experiments on node classification and influence estimation tasks demonstrated the effectiveness and efficiency of our solution.

Chapter 7

DeepSN: A Sheaf Neural Framework for Influence Maximization

7.1 Overview

Influence maximization (IM) is a challenging network science problem that involves identifying a set of nodes which, when activated, maximize the spread of influence across the network. IM has significant real-world applications including viral marketing (Li et al., 2009; Kempe et al., 2003), disease control (Marquetoux et al., 2016), social media content management (Hosseini-Pozveh et al., 2017), and crisis communication (Fan et al., 2021). Despite decades of research, IM still remains challenging primarily due its exponentially large search space and the complex nature of the influence diffusion processes.

A plethora of traditional methods have been proposed to obtain optimal or near-optimal solutions for IM (Kempe et al., 2003; Leskovec et al., 2007; Wang et al., 2010; Tang et al., 2015; Li et al., 2019). These methods vary in strategy and effectiveness: some offer theoretical performance guarantees, while others use heuristics to enhance scalability. A common trait among these traditional approaches is their reliance on the explicit specification of the influence diffusion model as input. Recently, researchers have turned to learning-based methods focusing on their ability to automatically identify the diffusion model from the ground truth and accommodate multiple diffusion models, thereby achieving broader applicability (Li et al., 2023c).

Learning-based approaches for IM encounter two key challenges. *(1) Effectively modeling the underlying diffusion model from ground truth:* There is a growing interest in leveraging Graph Neural Networks (GNNs) for modeling influence propagation, due to their ability to capture structural insights in networks (Kumar et al., 2022; Ling et al., 2023; Panagopoulos et al., 2023). However, existing work relies on traditional GNN techniques like attention (Lee et al., 2019) and convolution (Zhang et al., 2019). While these traditional GNNs excel in tasks with a static nature such as node classification and graph regression, their inductive bias and inherent assumptions limit their ability to model the dynamic behaviors of influence diffusion.

They also suffer from issues like over-smoothing (Chen et al., 2020a) which hampers their capacity to capture global information and long-range dependencies crucial for influence diffusion (Xia et al., 2021). *(2) Identifying the optimal subset of nodes with maximal Influence:* Designing an optimization objective to select the optimal seed set is arduous due to the vast search space, especially in large networks. Existing learning-based approaches often use deep reinforcement learning (Li et al., 2022b; Chen et al., 2023b) and ranking-based methods (Kumar et al., 2022; Panagopoulos et al., 2020a) to approximate the optimal seed set. However, these methods often incur significant computational costs and suffer from a lack of interpretability.

Our work addresses the limitations of existing approaches by leveraging sheaf theory (Tennison, 1975; Bredon, 2012), an algebraic-topological framework that captures the topological and geometric properties of complex networks, essential for understanding dynamic processes. Current sheaf diffusion GNNs (Hansen and Gebhart, 2020; Bodnar et al., 2022; Barbero et al., 2022) are limited by their homogeneous diffusion behaviors and inability to handle the evolving dynamics crucial for influence propagation. To tackle this, we propose *DeepSN*, a novel sheaf GNN framework designed to address the influence maximization problem. Our architecture effectively estimates diverse influence propagation processes by learning adaptive structural relationships between nodes, enabling effective identification of influence patterns in the network. Building on this, we design a seed set inference mechanism that uses subgraphs to efficiently reduce the search space in influence maximization. Our contributions are summarized as follows.

- **Influence Diffusion:** We redefine influence propagation as a sheaf diffusion-reaction process, effectively capturing the complex dynamics present in real-world information diffusion models.
- **GNN Architecture:** We propose a novel GNN architecture rooted in sheaf theory, specifically designed to capture the unique nuances of influence propagation.
- **Seed Selection:** We employ a subgraph-based strategy that leverages structural insights from our sheaf GNN to minimize overlapping influence among seed nodes, reducing the search space for influence maximization.
- **Experiments:** We conduct rigorous evaluations of our framework on real-world and synthetic datasets, showcasing its superior performance and generalizability compared to state-of-the-art methods.

This chapter addresses influence propagation through a sheaf-theoretic framework, different from the opinion dynamics approach of Chapter 6. The key distinction is in the modelling perspective: Chapter 6 draws on opinion dynamics to model how node states evolve under neighbourhood influence, while this chapter models influence propagation as a topological diffusion-reaction process using sheaf theory. Additionally, this chapter addresses the influence maximisation problem , extending beyond the estimation task.

7.2 DeepSN Framework

In this chapter, we introduce a deep learning framework called *DeepSN* to address the influence maximization problem. Our framework comprises two phases: *learning to estimate influence* and *optimizing seed selection*. In the influence estimation phase, our goal is to measure the total influence exerted by a set of initially activated nodes in a learnable way. This estimated influence is then used in the seed selection optimization phase to identify the optimal seed set that maximizes overall influence. Figure 7.1 provides an overview of our framework.

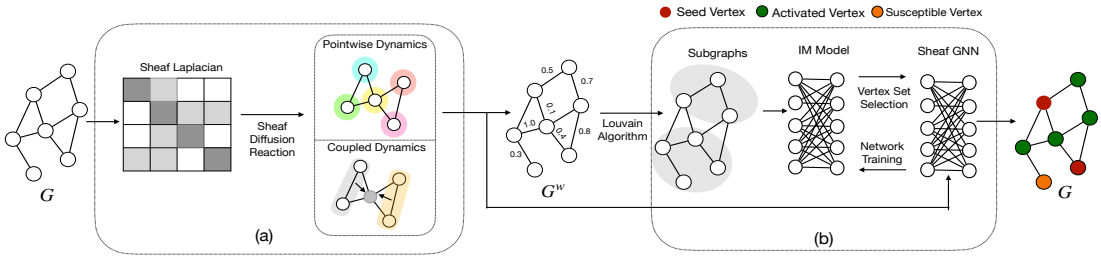


Figure 7.1: The DeepSN framework consists of two phases: (a) learning to estimate influence with sheaf GNN; b) optimizing seed selection using the subgraphs, IM model, and trained sheaf GNN.

7.2.1 Learning to Estimate Influence

We introduce a novel sheaf GNN model based on sheaf theory, specifically designed to adapt to the complex diffusion patterns inherent in influence propagation. A sheaf on a graph is a structure that assigns a vector space to each node and each edge, together with linear maps that relate node spaces to their incident edge spaces. Intuitively, node spaces store information local to each node while edge spaces capture the shared information exchanged along connections. The sheaf Laplacian measures the inconsistency between what a node projects onto an edge and what its neighbour projects onto the same edge. This richer structure allows the model to represent heterogeneous information flow across different edges, going beyond the scalar edge weights of standard GNNs and making sheaf-based diffusion particularly well-suited to capturing complex propagation patterns.

7.2.1.1 Topological Sheaf Diffusion

We begin by introducing the concept of *cellular sheaf*, the foundational building block of our GNN.

Definition 16 (Cellular Sheaf)**.** *A* (cellular) sheaf (G, F) *on a graph* $G = (V, E)$ *consists of a vector space* F_v *for each node* $v \in V$ *, a vector space* F_e *for each edge* $e \in E$*, a linear map* $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}_v \to \mathcal{F}_e$ *for each incident node-edge pair* $v \trianglelefteq e$*.*

$\mathcal{F}_v$ and $\mathcal{F}_e$ are the *node sheaf* and *edge sheaf*, respectively, while $\mathcal{F}_{v \trianglelefteq e}$ is the *transformation map*. Let $\bigoplus$ denote the direct sum of vector spaces. The space of *0-cochains* is the direct sum of vector spaces over the nodes: $C^0(G, F) = \bigoplus_{v \in V} F_v$, and the space of *1-cochains* is the direct sum of vector spaces over the edges: $C^1(G, F) = \bigoplus_{e \in E} F_e$.

We formulate influence propagation in a network as an opinion propagation process, where each node corresponds to a node sheaf, representing a private opinion. Edge sheaves represent public opinions related to influence propagation, while transformation maps translate information from node sheaves to edge sheaves, extracting public opinions from private opinions. Let $x_v \in F_v$ be a d-dimensional feature vector for each $v \in V$. The *coboundary map* $\delta : C^0(G, F) \to C^1(G, F)$ is a linear transformation,

defined for an edge (v, u) as

$$\delta(x)_e = \mathcal{F}_{v \trianglelefteq e}.x_v - \mathcal{F}_{u \trianglelefteq e}.x_u \tag{7.1}$$

where $\mathcal{F}_{v \trianglelefteq e} \cdot x_v$ and $\mathcal{F}_{u \trianglelefteq e} \cdot x_u$ represent the public opinions associated with e, originating from nodes v and u, respectively.

The sheaf Laplacian of a node measures the disparity between its public opinion and that of its neighbors. To capture the fine-grained nuances of trust, a crucial factor in modeling influence propagation, we incorporate learnable sheaf coefficients to model each node's confidence in the opinions received from its neighbors.

Definition 17 (Non-linear Sheaf Laplacian)**.** *Let $\psi_{vu} \in [0, 1]$. The sheaf Laplacian $L_{\mathcal{F}} : C^0(G; F) \to C^0(G; F)$ on a sheaf (G, F) is defined node-wise as*

$$L_{\mathcal{F}}(x_v) = \sum_{v,u \trianglelefteq e} \mathcal{F}^T_{v \trianglelefteq e}.\psi_{vu}.\left(\mathcal{F}_{v \trianglelefteq e}.x_v - \mathcal{F}_{u \trianglelefteq e}.x_u\right). \tag{7.2}$$

Here, ψ_{vu} is the sheaf coefficient which is learnable. Note that the sheaf Laplacian employed by Bodnar et al. (2022) is a specific instance of our Laplacian, where $\psi_{u,v} = 1$ for all $(u, v) \in E$.

To ensure that the Laplacian matrix $\hat{L}_{\mathcal{F}}$ remains positive definite, a necessary condition for the convergence of the diffusion process, we apply the following modification:

$$\hat{L}_{\mathcal{F}} = L_{\mathcal{F}} + \epsilon \cdot I \tag{7.3}$$

where ϵ is a scalar, I is the identity matrix, and $\cdot$ denotes element-wise multiplication. The following lemma establishes the necessary and sufficient conditions for $\hat{L}_{\mathcal{F}}$.

Lemma 2. *Let λ_{min} denote the smallest eigenvalue of $L_{\mathcal{F}}$. $\hat{L}_{\mathcal{F}}$ is positive definite if and only if $\epsilon > -\lambda_{min}$.*

Proof. Assume $\hat{L}_{\mathcal{F}} = L_{\mathcal{F}} + \epsilon I$ is positive definite. This implies all eigenvalues $\mu_i = \lambda_i + \epsilon > 0$, where λ_i are eigenvalues of $L_{\mathcal{F}}$. The smallest eigenvalue, $\lambda_{\min}$, sets the strictest condition: $\lambda_{\min} + \epsilon > 0$, or $\epsilon > -\lambda_{\min}$. Conversely, if $\epsilon > -\lambda_{\min}$, then $\mu_i = \lambda_i + \epsilon > 0$ for all eigenvalues λ_i of $L_{\mathcal{F}}$. Therefore, $\hat{L}_{\mathcal{F}}$ is positive definite. The proof is complete. □

The sheaf diffusion operator (i.e., normalized sheaf Laplacian) models opinion diffusion by capturing discrepancies between node opinions and enabling smooth information propagation across the network, defined as

$$\Delta_{\mathcal{F}} = D^{-\frac{1}{2}} L_{\mathcal{F}} D^{-\frac{1}{2}}, \tag{7.4}$$

where D is the block-diagonal of $L_{\mathcal{F}}$.

Let $x \in C^0(G, F)$ denote an nd-dimensional vector constructed by column-stacking the individual vectors x_v. This vector x is then transformed into a node feature matrix $X \in \mathbb{R}^{(nd)\times f}$, where each column corresponds to a vector in $C^0(G; F)$ and f is the number of feature channels. The sheaf standard diffusion process is defined by the following Partial Differential Equation (PDE),

$$X(0) = X, \quad \frac{\partial X(t)}{\partial t} = -\Delta_{\mathcal{F}} X(t). \tag{7.5}$$

Given a diffusion coefficient $\alpha \in [0, 1]$, the discrete-time update rule for diffusion is defined as

$$X(t+1) = X(t) - \alpha \cdot \frac{\partial X(t)}{\partial t}; \alpha > 0. \tag{7.6}$$

When node features do not change between time steps, (i.e., $X(t+1) = X(t)$), a fixed point $X(t)$ is reached, which is also referred to as a *steady state*.

7.2.1.2 Sheaf Reaction Diffusion

Traditional diffusion GNNs including sheaf diffusion in Eq. 7.5 (Bodnar et al., 2022; Hansen and Gebhart, 2020) use a uniform diffusion process that primarily focuses on influence spreading but overlooks how individual node characteristics and neighboring state transitions shape the process, which is essential for modeling influence diffusion. To address this issue, we model influence dynamics as a diffusion-reaction process (Turing, 1990; Choi et al., 2023). In this process, *diffusion* refers to the spread of information across the network, governed by its connectivity structure, and *reaction* involves altering this information based on interactions and inherent characteristics of nodes within the network.

Reaction Operators To capture complex dynamics related to influence diffusion, we introduce the *sheaf reaction diffusion* which extends the sheaf standard diffusion with two reaction operators for *pointwise dynamics* and *coupled dynamics*. That is,

$$\begin{aligned} \frac{\partial X_v(t)}{\partial t} = &- \underbrace{\alpha \Delta_{\mathcal{F}} X_v(t)}_{\text{Diffusion}} + \underbrace{\beta A\left(X_v(t)\right)}_{\text{Pointwise Dynamics}} \\ &+ \underbrace{\gamma R_v\left(X(t), A, S^t\right)}_{\text{Coupled Dynamics}} \end{aligned} \tag{7.7}$$

Here, $X_v(t)$ represents the feature vector of node v at time t. α, β, and γ control

the contribution of each term, and $S^t \in [0,1]^n$ is a vector representing the activation probabilities of nodes at time step t.

(a) Pointwise Dynamics Pointwise dynamics refers to inherent characteristics that govern its activation or susceptibility. For instance, in models of opinion dynamics or epidemic propagation, a node may alter its activation state due to internal factors such as personal reassessment or spontaneous recovery, independent of its interactions with other nodes. Thus, we design a reaction operator to capture such node evolution as

$$A\left(X_v(t)\right) = \Phi_v^1 \odot \frac{X_v(t)}{\kappa_v^1 + |X_v(t)|} \tag{7.8}$$

where $\Phi_v^1, \kappa_v^1 \in \mathbb{R}^{d\times f}$ are coefficient vectors with $\kappa_v^1 > 0, \kappa_v^1 \neq -|X_v(t)|$, and $\odot$ is the Hadamard product.

(b) Coupled Dynamics Coupled dynamics describe how neighboring interactions of nodes influence the activation or susceptibility of a node, creating a dynamic interplay between individual behaviors and network-wide interactions. We define a reaction operator for node $v \in V$ as

$$X_v^\dagger(t) = \sum_{u\in N(v)} \left(S_u^t \cdot X_u(t)\right) - \sum_{u\in N(v)} \left((1 - S_u^t) \cdot X_u(t)\right); \tag{7.9}$$

$$R_v\left(X(t), A, S^t\right) = \Phi_v^2 \odot \frac{X_v^\dagger(t)}{\kappa_v^2 + |X_v^\dagger(t)|} \tag{7.10}$$

where Φ_v^2 and κ_v^2 are coefficient vectors with $\kappa_v^2 > 0, \kappa_v^2 \neq -|X_v^\dagger(t)|$ for any $t > 0$, and S_u^t denotes the activation probability of node u at time t. This reaction operator accounts for the combined impact of both activated and susceptible neighbors by producing either positive or negative effects: a high activation level in the neighborhood yields a positive impact, whereas a high susceptibility leads to a negative impact. Thus, it can enhance the model's ability to capture complex dynamics in a network.

Remark 6. *Recent research has largely focused on progressive models (Chen et al., 2022a,b), where nodes remain indefinitely active once activated. However, this is often unrealistic, as many real-world propagation models involve nodes transitioning between active and inactive states (Lou et al., 2014). While diffusion terms typically follow fixed-point dynamics, our sheaf diffusion model uses reaction operators to introduce non-monotonic, adaptive, and oscillatory behaviors, capturing dynamic node state transitions.*

Stability of Reaction Diffusion Stability in a diffusion process refers to its ability to maintain a controlled and predictable behavior over time (Wang et al., 2022a). A stable process maintains node features within a bounded region over time. This bounded behavior allows the diffusion process to converge to or remain near a fixed point. If node features become unbounded, it signals instability, indicating that the

diffusion process can deviate significantly from a fixed point or expected behavior (Bellman, 1947; Sastry, 2013). Such instability can lead to unpredictable outcomes and make it difficult to control or interpret diffusion dynamics effectively. In the following, we explain the bounded behavior of the reaction operators.

Lemma 3. *The reaction operators in Eq. 7.7 are bounded for all $t > 0$ and $v \in V$ in terms of norms $||\cdot||$. We have $\|A(X_v(t))\| < \|\Phi_v^1\|$ and $\|R_v(X(t), A, S^t)\| < \|\Phi_v^2\|$.*

Proof. Let $X_v(t) = (X_v^1(t), \ldots, X_v^i(t), \ldots, X_v^f(t))$. Then, $A_v(X(t))$ can be expressed as:

$$A_v(X(t)) = \begin{pmatrix} \frac{\Phi_{v,1}^1 X_v^1(t)}{\kappa_{v,1}^1 + |X_v^1(t)|}, \\ \vdots \\ \frac{\Phi_{v,i}^1 X_v^i(t)}{\kappa_{v,i}^1 + |X_v^i(t)|}, \\ \vdots \\ \frac{\Phi_{v,n}^1 X_v^n(t)}{\kappa_{v,n}^1 + |X_v^n(t)|} \end{pmatrix}$$

As $X_v^i(t) \to 0$:

$$\lim_{X_v^i(t)\to 0} \frac{\Phi_{v,i}^1 X_v^i(t)}{\kappa_{v,i}^1 + |X_v^i(t)|} = \frac{\Phi_{v,i}^1 \cdot 0}{\kappa_{v,i}^1 + 0} = 0$$

As $X_v^i(t) \to \infty$:

$$\lim_{X_v^i(t)\to \infty} \frac{\Phi_{v,i}^1 X_v^i(t)}{\kappa_{v,i}^1 + |X_v^i(t)|} = \frac{\Phi_{v,i}^1}{1 + \frac{\kappa_{v,i}^1}{X_v^i(t)}} = \Phi_{v,i}^1$$

As $X_v^i(t) \to -\infty$:

$$\lim_{X_v^i(t)\to -\infty} \frac{\Phi_{v,i}^1 X_v^i(t)}{\kappa_{v,i}^1 + |X_v^i(t)|} = \frac{\Phi_{v,i}^1}{-1 + \frac{\kappa_{v,i}^1}{X_v^i(t)}} = -\Phi_{v,i}^1$$

Further, $\frac{\Phi_{v,i}^1 X_v^i(t)}{\kappa_{v,i}^1 + |X_v^i(t)|} < \Phi_{v,i}^1$ always holds since $\kappa_{v,i}^1 > 0$ and $\kappa_{v,i}^1 \neq -|X_v^i(t)|$. Therefore, the values of $A_v(X(t))$ are bounded by the corresponding values of Φ_v^1. Applying the same reasoning, we can prove that the values of $R_v\left(X(t), A, S^t\right)$ are bounded by the corresponding values of Φ_v^2. Consequently, the norms of $A_v(X(t))$ and $R_v\left(X(t), A, S^t\right)$ are also bounded by the norms of Φ_v^1 and Φ_v^2, respectively. This completes the proof.

□

The boundedness of the reaction operators ensures that the fixed-point of Eq. 7.7 is also bounded, as demonstrated in the lemma below.

Lemma 4. *Let $X^\star$ denote the fixed point of Eq. 7.7. Given that $\Delta_{\mathcal{F}}$ is well-defined (i.e., positive definite) and operates on a finite-dimensional space, $X^\star$ is bounded.*

Proof. Let us consider the diffusion PDE:

$$\frac{\partial X(t)}{\partial t} = -\alpha\Delta_{\mathcal{F}}X(t) + \beta A\left(X(t)\right) + \gamma R\left(X(t), A, S^t\right)$$

Assuming the system reaches a fixed point, we have $\frac{\partial X(t)}{\partial t} = 0$. At this point, the equation simplifies to:

$$\alpha\Delta_{\mathcal{F}}X^\dagger = \beta A\left(X^\dagger\right) + \gamma R\left(X^\dagger, A, S^t\right)$$

According to Lemma 3, the terms $A(X(t))$ and $R(X(t), A, S^t)$ are bounded between $-\Phi_1$ and Φ_1, and $-\Phi_2$ and Φ_2, respectively, where Φ_1 and Φ_2 are matrices constructed by stacking the vectors Φ_v^1 and Φ_v^2 as rows.

First, we derive an upper bound for the L2 norm of $X^\dagger$.

$$\|\alpha\Delta_{\mathcal{F}}X^\dagger\|_2 \leq |\beta| \cdot \|\Phi_1\|_2 + |\gamma| \cdot \|\Phi_2\|_2$$

Rearranging by factoring out $|\alpha|$ gives:

$$|\alpha|\|\Delta_{\mathcal{F}}X^\dagger\|_2 \leq |\beta| \cdot \|\Phi_1\|_2 + |\gamma| \cdot \|\Phi_2\|_2$$

Dividing through by $|\alpha|$ (assuming $\alpha \neq 0$) leads to:

$$\|\Delta_{\mathcal{F}}X^\dagger\|_2 \leq \frac{|\beta| \cdot \|\Phi_1\|_2 + |\gamma| \cdot \|\Phi_2\|_2}{|\alpha|}.$$

Given that $\Delta_{\mathcal{F}}$ is positive definite, it is symmetric and invertible, with all eigenvalues strictly positive. The largest eigenvalue is denoted as $\lambda_{\max}$, implying that the $||\Delta_{\mathcal{F}}||_2$ equal to $\lambda_{\max}$. Combining these facts, the norm of $X^\dagger$ can be bounded by:

$$\|X^\dagger\|_2 \leq \frac{1}{|\alpha|\lambda_{\max}}(|\beta| \cdot \|\Phi_1\|_2 + |\gamma| \cdot \|\Phi_2\|_2).$$

□

7.2.2 Sheaf GNN Training

Building on the sheaf Laplacian and reaction operators, our GNN applies the following diffusion propagation rule to update node features:

$$\begin{aligned} X_v(t+1) = X_v(t) &- \alpha\left(\Delta_{\mathcal{F}}\left(I_n \otimes W_1^t\right) X_v(t) W_2^t\right) \\ &+ \beta\left(\Phi_v^1 \odot \frac{X_v(t)}{\kappa_v^1 + X_v(t)}\right) \\ &+ \gamma\left(\Phi_v^2 \odot \frac{X_v^\dagger(t)}{\kappa_v^2 + |X_v^\dagger(t)|}\right) \end{aligned} \tag{7.11}$$

Notably, $W_1^t \in \mathbb{R}^{d\times d}$, $W_2^t \in \mathbb{R}^{f\times f}$, and $\Phi_v^1, \Phi_v^2, \kappa_v^1, \kappa_v^2 \in \mathbb{R}^{d\times f}$ are learnable weight

matrices, $\otimes$ denotes the Kronecker product, and $I_n \in \mathbb{R}^{n \times n}$ represents an identity matrix. Moreover, $X(0)$ is obtained by transforming the node features of the network through a multi-layer perceptron (MLP), followed by reshaping, resulting in a matrix of dimensions $(nd) \times f$.

After obtaining the updated node embeddings in each iteration, we utilize a non-linear neural function $f_\eta(\cdot)$, parameterized by η, to determine the activation state of each node $v \in V$ at the time step $t+1$ as

$$S_v^{t+1} = f_\eta\Big(X_v(t+1)\Big); S_v^{t+1} \in [0,1]. \tag{7.12}$$

We use the Mean Square Error (MSE) loss to measure the difference between the predicted activation probabilities $\hat{S}$ and the ground truth probabilities $Y \in [0,1]^n$ as:

$$\mathcal{L}_{train} = ||\hat{S} - Y||_2^2 \tag{7.13}$$

7.2.3 Optimizing Seed Selection

In this section, we present a learning-based approach to optimally select a seed set that maximizes influence spread. Selecting an optimal seed set for influence maximization faces a combinatorial explosion, with node combinations growing exponentially as the graph size increases.

We address this challenge leveraging two observations. First, instead of relying on the adjacency matrix for graph connectivity, which fails to capture network dynamics in diffusion models, we learn the connectivity through sheaf coefficients, enhancing model flexibility. This leads to a weighted graph G^w that is constructed from sheaf coefficients and the adjacency matrix of the input graph. Then, we employ a partitioning mechanism to identify subgraphs, which reduces the search space in the process of determining the optimal seed set. We apply the Louvain algorithm (Blondel et al., 2008; Dugué and Perez, 2015; Traag et al., 2019) to divide the graph G^w into r subgraphs $\{G_i\}_{i=1}^r$, minimizing the overlap of influence between nodes in different subgraphs. Then, allocating seed nodes across subgraphs can significantly reduce the search space by limiting the number of node combinations within each smaller subgraph, rather than across the entire graph.

We train a neural network $\mathcal{T}_\phi$, parameterized by ϕ, to select seed nodes within subgraphs in a learnable manner. More specifically, $\mathcal{T}_\phi$ learns to select $S_i \subseteq V_i$ seed nodes from each subgraph $G_i = (V_i, E_i)$ such that $S = \bigcup_{i=1}^k S_i$ can maximize the overall influence spread

$$S^* = \mathcal{T}_\phi\Big(\{G_i\}_{i=1}^r, G^w\Big) \text{ subject to } \bigwedge_{i \in [1,r]} |S_i| \leq \frac{k}{n}|V_i|. \tag{7.14}$$

The condition in the above equation ensures that the number of seeds is chosen proportionally to the size of each subgraph. $\mathcal{T}_\phi$ is trained using a loss function based

on the difference between the maximal influence and the predicted influence

$$\mathcal{L}_{train} = n - \sigma\Big(\bigcup_{i=1}^{r} S_i, G^w; \theta\Big). \tag{7.15}$$

Note that we use the sheaf GNN with trained parameters θ to approximate the influence diffusion function $\sigma(\cdot)$.

7.3 Theoretical Analysis

7.3.1 Complexity Analysis

We first analyze the layer-wise complexity of our GNN component. The diffusion operation within our GNN has a complexity of $O\left(n\left(f^2d^2 + d^3\right) + m\left(fd + d^3\right)\right)$, where m is the number of edges. The complexity of this operation is similar to that reported in Bodnar et al. (2022). Each reaction operator introduces an additional complexity of $O(ndf)$. Consequently, the total complexity of our GNN is $O\left(n\left(f^2d^2 + d^3\right) + m\left(fd + d^3\right)\right)$. Given that we employ $d = \{1, 2\}$ in our experiments, our GNN incurs only a constant overhead compared to traditional GNNs, such as GCN (Kipf and Welling, 2016).

The Louvain algorithm, used as a preprocessing step, has a time complexity of $O(lm)$, where l is the number of iterations, and a space complexity of $O(n + m)$ (Lancichinetti and Fortunato, 2009). In our experiments, we employ DeepSN$_{\text{SP}}$ with a sparsified graph structure, resulting in a time complexity of $O(l \times n \log n)$ and a space complexity of $O(n)$. We implement $\mathcal{T}_\phi$ using a multi-layer perceptron (MLP) with a time complexity of $O(ndh)$, where h represents the number of hidden neurons in the MLP.

7.3.2 Node Feature Separability

We investigate the separation power of our sheaf GNN for node features, which plays a crucial role in mitigating oversmoothing (Bodnar et al., 2022). The following proposition shows the existence of the fixed point in the sheaf diffusion process under Eq. 7.5 and the corresponding properties of transformation maps.

Proposition 2. *For any graph $G = (V, E)$ with sheaf Laplacian $L_{\mathcal{F}}$ and initial node features $X(0)$, there exists a unique fixed-point $X(t)$ in the sheaf diffusion process such that for any $(v, u) \in E$, $\mathcal{F}_{v \trianglelefteq e} x_v = \mathcal{F}_{u \trianglelefteq e} x_u$ holds.*

Proof. Let us consider the diffusion PDE given in Eq. (7.5):

$$X(0) = X, \ \frac{\partial X(t)}{\partial t} = -\Delta_{\mathcal{F}} X(t)$$

In accordance with Theorem 2.2 in Hansen and Ghrist (2021), X converges to $H_0(G; \mathcal{F})$ in the fixed point, where $H_0(G; \mathcal{F}) = \{x \in C_0(G; \mathcal{F}) \mid \mathcal{F}_{u \trianglelefteq e} x_u = \mathcal{F}_{v \trianglelefteq e} x_v\}$.

This implies that, for any adjacent nodes v and u, the following holds:

$$\mathcal{F}_{v \trianglelefteq e} x_v = \mathcal{F}_{u \trianglelefteq e} x_u.$$

□

Due to the condition $\mathcal{F}_{v \trianglelefteq e} x_v = \mathcal{F}_{u \trianglelefteq e} x_u$ at the fixed point, the sheaf diffusion process has limited capacity to separate distinct node features between a node v and its neighbors u, as shown in Proposition 3.

Proposition 3. *There exists a graph $G = (V, E)$ with the fixed point $X(t)$ in the sheaf diffusion process which satisfies $x_u = x_v$ for at least one $(u, v) \in E$.*

Proof. We employ Proposition 9 from Bodnar et al. (2022) to prove this proposition. Consider a set of connected bipartite graphs $G = (A, B, E)$, with partitions A and B forming two distinct classes, where $|A| = |B|$. According to their proposition, restriction maps defined by symmetric sheaves of dimension 1, specifically

$$\mathcal{H}_{\text{sym}} := \{(F, G) : F_{v \trianglelefteq e} = F_{u \trianglelefteq e}, \det(F_{v \trianglelefteq e}) \neq 0\},$$

are incapable of separating the classes of any graph in G for any initial conditions $X(0)$. This is due to the fact that, for adjacent nodes u and v, we have $x_v = x_u$, which prevents differentiation between the classes.

□

Unlike traditional sheaf diffusion, the sheaf reaction diffusion (Eq. 7.7) does not require the condition $\mathcal{F}_{v \trianglelefteq e} x_v = \mathcal{F}_{u \trianglelefteq e} x_u$ to be satisfied upon convergence. The following proposition demonstrates this.

Proposition 4. *For any graph $G = (V, E)$ with sheaf Laplacian $L_{\mathcal{F}}$ and initial node features $X(0)$, there exists a unique fixed-point $X(t)$ in the sheaf reaction diffusion process; however, $\mathcal{F}_{v \trianglelefteq e} x_v = \mathcal{F}_{u \trianglelefteq e} x_u$ does not necessarily hold for each $(v, u) \in E$.*

Proof. We start with the reaction diffusion equation:

$$X(t+1) = X(t) - \frac{\partial X(t)}{\partial t}$$

where,

$$\frac{\partial X(t)}{\partial t} = -\alpha \Delta_{\mathcal{F}} X(t) + \beta A\left(X(t)\right) + \gamma R\left(X(t), A, S^t\right)$$

When the system reaches a fixed point, we have $\frac{\partial X_(t)}{\partial t} = 0$. At this point, the equation simplifies to:

$$\alpha \Delta_{\mathcal{F}} X^\dagger = \beta A\left(X^\dagger\right) + \gamma R\left(X^\dagger, A, S^t\right)$$

We can write this as:

$$\alpha\Delta_{\mathcal{F}}X^{\dagger} = \frac{a \cdot X^{\dagger}}{b + X^{\dagger}} + \frac{c \cdot X^{\dagger}}{d + X^{\dagger}}$$

where a, b, c, d are matrices with same dimension as $\Delta_{\mathcal{F}}$. First, we analyze the existence of a non-trivial fixed point. To do this, we can simplify the equation by dividing both sides by $X^{\dagger}$, assuming $X^{\dagger} \neq 0$. This gives us the equation $\alpha\Delta_{\mathcal{F}} = f(X^{\dagger})$, where

$$f(X^{\dagger}) = \frac{a}{b + X^{\dagger}} + \frac{c}{d + X^{\dagger}}.$$

The function $f(X^{\dagger})$ is continuous and strictly decreasing. This is because as $X^{\dagger}$ increases, the terms $\frac{a}{b+X^{\dagger}}$ and $\frac{c}{d+X^{\dagger}}$ both decrease, making the entire function $f(X^{\dagger})$ decrease. Specifically, when $X^{\dagger} = 0$, the function $f(X^{\dagger})$ takes its maximum value, $\frac{a}{b} + \frac{c}{d}$. As $X^{\dagger}$ increases towards infinity, the function $f(X^{\dagger})$ decreases toward 0. For a non-trivial fixed point $X^{\dagger} > 0$ to exist, the value $\alpha\Delta_{\mathcal{F}}$ must fall within the range of values that $f(X^{\dagger})$ can take. This means $\alpha\Delta_{\mathcal{F}}$ must satisfy $0 < \alpha\Delta_{\mathcal{F}} < \frac{a}{b} + \frac{c}{d}$. The uniqueness of $X^{\dagger}$ is guaranteed because $f(X^{\dagger})$ is strictly decreasing and continuous. Therefore, under the condition that $0 < \alpha\Delta_{\mathcal{F}} < \frac{a}{b} + \frac{c}{d}$, there is a unique, non-trivial fixed point $X^{\dagger} > 0$ that solves the equation.

We again consider the equation:

$$\alpha\Delta_{\mathcal{F}}X^{\dagger} = \beta A\left(X^{\dagger}\right) + \gamma R\left(X^{\dagger}, A, S^{t}\right)$$

$$\alpha\Delta_{\mathcal{F}}X^{\dagger} = \frac{a \cdot X^{\dagger}}{b + X^{\dagger}} + \frac{c \cdot X^{\dagger}}{d + X^{\dagger}}$$

Since $X^{\dagger}$ can be non-trivial (i.e., not equal to 0), we can see that R.H.S. can be non-zero, leading to $\alpha\Delta_{\mathcal{F}}X^{\dagger} \neq 0$. This implies that in the fixed-point, the solution does not necessarily converge to $H_0(G;\mathcal{F}) = \{x \in C_0(G;\mathcal{F}) \mid \mathcal{F}_{u \trianglelefteq e}x_u = \mathcal{F}_{v \trianglelefteq e}x_v\}$ (i.e., it can converge to $\mathcal{F}_{u \trianglelefteq e}x_u \neq \mathcal{F}_{v \trianglelefteq e}x_v$ for adjacent nodes u and v). □

Remark 7. *Relaxation of condition $\mathcal{F}_{v \trianglelefteq e}x_v = \mathcal{F}_{u \trianglelefteq e}x_u$ in proposition 4 significantly improves the separability of distinct node features. This allows sheaf GNN to learn node features distinguishable from their neighbors through more powerful transformation maps, offering an advantage over existing sheaf diffusion networks (Bodnar et al., 2022).*

7.4 Experimental Design

A set of experiments was conducted to evaluate the performance of DeepSN, considering two variants: *DeepSN* and *DeepSN$_{SP}$*, a computationally efficient variant that uses sheaf coefficients to sparsify the graph structure. We focus on three diffusion models: IC, LT, and SIS (Li et al., 2018). IC and LT are progressive, while SIS is non-progressive.

7.4.1 Datasets

We evaluate DeepSN against other methods using a diverse set of datasets, including five real-world datasets (Jazz (Rossi and Ahmed, 2015), Network Science (Rossi and Ahmed, 2015), Cora-ML (McCallum et al., 2000), Power Grid (Rossi and Ahmed, 2015), and Digg (Lerman and Galstyan, 2008)) and one synthetic dataset (Random (Ling et al., 2023)), which range from small graphs to those with over 250,000 nodes. Table 7.1 summarizes statistics of these datasets.

Table 7.1: Summary of Dataset Statistics

Dataset	**# of nodes**	**# of Edges**
Jazz	198	2,742
Network Science	1,565	13,532
Cora-ML	2,810	7,981
Power Grid	4,941	6,594
Random	50,000	250,000
Digg	279,613	1,170,689

7.4.2 Experimental Setups

For influence maximization, we adopt the experimental setup outlined by Ling et al. (2023) for our influence maximization tasks, selecting 1%, 5%, 10%, and 20% of the nodes in each dataset as seed nodes, simulating each diffusion model until the diffusion process halts, or converges to a steady-state and recording the average influence spread over 100 repetitions. Baseline results are taken from Ling et al. (2023).

To evaluate the influence estimation performance in DeepSN, we adhere to the training-testing-validation process outlined by Vabalas et al. (2019), splitting the dataset into 60% for training, 20% for testing, and 20% for validation. For the baseline methods, we use the hyper-parameters reported in their original papers, and report the results.

7.4.3 Baselines

For influence maximization task, we compare DeepSN with traditional baselines: IMM (Tang et al., 2015), OPIM (Tang et al., 2018), SubSIM (Guo et al., 2020) and learning-based methods: IMINFECTOR (Panagopoulos et al., 2020b), ToupleGDD (Chen et al., 2023b), PIANO (Li et al., 2022b), DeepIM (Ling et al., 2023).

We compare the influence estimation performance of DeepSN with several widely used GNNs: GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017), DSN (Bodnar et al., 2022).

7.4.4 Hyper-parameters

In our experiments, we search the hyper-parameters of DeepSN within the following ranges: the number of GNN layers $\in \{2, 5, 10\}$, the dimension $d \in \{1, 2\}$, dropout rate $\in \{0.1, 0.2, 0.5, 0.9\}$, learning rate $\in \{0.001, 0.002, 0.004\}$, batch size $\in \{2, 8, 16, 32\}$, the number of hidden units in the MLP $\in \{32, 64, 128\}$, and the resolution parameter of the Louvain algorithm $\in \{0.1, 1, 2\}$. We employ the Adam algorithm as the optimizer (Kingma, 2015). Further, DeepSN$_{SP}$ employs a threshold of 0.5 to convert continuous sheaf coefficients into binary values.

7.4.5 Computational Resources

All experiments were performed on a Linux server equipped with an Intel Xeon W-2175 2.50GHz processor with 28 cores, an NVIDIA RTX A6000 GPU, and 512GB of main memory.

7.5 Results and Discussion

7.5.1 Influence Maximization

We evaluate DeepSN's performance in selecting an optimal seed set for the IM task across various budget constraints $\{1\%, 5\%, 10\%, 20\%\}$ (i.e. seed set size as a percentage of total number of nodes). The results are presented in tables 7.2, 7.3, and 7.4 for IC, LT, and SIS models, respectively. We observe that both DeepSN and DeepSN$_{SP}$ outperform or deliver comparable performance across all diffusion models. Notably, DeepSN achieves a substantial improvement over all baseline methods for the SIS diffusion model. This enhanced performance is attributed to DeepSN's capability to effectively capture complex diffusion dynamics inherent to non-progressive diffusion models.

7.5.2 Influence Estimation

In Figure 7.2, we compare the influence estimation performance of DeepSN with several widely used GNNs. The results show that DeepSN significantly outperforms traditional GNNs, such as GCN, GAT, and GraphSAGE, across all diffusion models and datasets, highlighting its superiority in influence estimation. These traditional GNNs often struggle to capture long-range dependencies due to oversmoothing, leading to lower performance. Furthermore, while existing sheaf neural networks like DSN address some issues, they still struggle to capture the intricate dynamics needed for influence estimation, leading to suboptimal performance.

7.5.3 Impact of Influence Maximization Component

We evaluate the effectiveness of our IM model $\mathcal{T}_\phi$ by replacing it with different IM variants. These variants include: **DeepSN-CELF**, which incorporates the CELF algo-

	Cora-ML				Network Science				Power Grid				Jazz				Random				Digg			
Methods	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
IMM	8.1	26.2	37.3	50.2	5.2	16.8	27.0	45.7	5.6	17.4	31.5	51.1	2.6	20.1	31.4	42.8	9.2	26.2	36.3	51.6	7.4	18.6	32.8	49.6
OPIM	13.4	26.9	37.4	50.9	6.4	19.4	28.9	48.6	5.7	17.7	29.7	50.1	2.4	20.1	34.4	46.8	9.6	25.3	36.6	51.7	7.6	18.5	32.9	48.9
SubSIM	10.1	25.7	36.8	51.1	4.8	15.4	27.9	44.8	4.6	19.2	31.7	50.2	3.6	18.8	37.6	44.7	9.5	26.7	36.5	51.3	7.5	18.9	33.3	49.4
IMINfECTOR	9.6	26.8	37.7	50.6	5.4	17.9	27.8	47.6	5.4	18.2	31.6	50.9	3.6	19.7	37.5	45.9	9.1	26.2	36.1	51.3	7.9	18.6	33.5	49.8
PIANO	9.8	25.2	37.4	51.1	5.3	18.1	27.1	47.2	5.3	18.1	31.7	50.2	2.2	19.2	36.6	43.2	9.2	25.6	36.5	51.6	7.6	18.3	33.6	49.5
ToupleGDD	10.6	27.5	38.5	51.5	6.3	17.8	28.3	50.5	5.4	19.3	31.6	51.3	3.3	20.4	37.2	45.7	9.5	26.8	37.1	51.4	-	-	-	-
DeepIM	14.1	28.1	39.6	52.4	7.8	20.9	31.5	51.2	6.3	21.0	32.5	52.4	4.9	23.3	41.5	49.9	**11.6**	27.4	38.7	52.1	8.4	19.3	34.2	51.3
DeepSN	11.5	25.6	40.9	52.8	6.2	**22.0**	**32.0**	**52.4**	6.4	**24.0**	36.9	**61.0**	8.5	26.9	41.6	53.8	10.6	**27.8**	**38.8**	**52.8**	8.9	**19.5**	35.2	**52.8**
DeepSN$_{SP}$	**14.2**	**30.1**	**42.3**	**58.4**	**7.9**	18.6	30.0	51.2	**7.1**	22.4	**37.4**	57.4	**8.8**	**29.9**	**41.9**	**55.6**	9.8	27.2	37.6	51.6	**9.1**	19.2	**35.4**	51.9

Table 7.2: Performance comparison under Independent Cascade (IC) diffusion model. - indicates out-of-memory error. The best results are highlighted in **bold**. Baseline results are sourced from Ling et al. (2023).

	Cora-ML				Network Science				Power Grid				Jazz				Random				Digg			
Methods	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
IMM	1.7	34.8	52.2	66.4	2.5	11.8	18.1	33.6	4.6	19.9	31.7	56.9	1.4	5.7	13.4	24.5	1.1	5.2	13.1	66.9	2.4	10.8	37.4	55.6
OPIM	2.3	36.9	51.2	71.5	1.6	12.0	18.1	34.1	4.4	21.6	29.4	55.5	1.4	6.9	12.6	20.9	1.3	5.4	12.6	62.1	2.1	11.3	38.2	57.1
SubSIM	1.7	33.6	54.7	70.1	1.8	10.4	19.2	34.1	4.5	21.1	31.2	57.4	1.4	5.9	11.4	21.2	1.4	5.5	13.1	69.6	2.4	11.3	37.9	56.9
IMINfECTOR	2.1	33.9	51.3	70.6	2.1	11.8	18.7	34.5	4.2	21.3	31.6	56.2	1.4	6.2	13.5	22.8	1.3	5.2	12.9	67.4	2.2	11.1	38.9	58.7
PIANO	2.1	33.5	53.3	69.8	2.1	11.3	19.1	33.9	4.3	21.3	31.4	57.1	1.1	6.2	12.2	22.4	1.2	5.2	12.8	67.4	-	-	-	-
ToupleGDD	2.3	36.2	54.5	70.9	2.8	12.4	19.8	34.6	4.8	21.9	32.6	58.1	1.4	6.5	12.9	23.6	1.3	5.5	13.4	70.2	-	-	-	-
DeepIM	**13.4**	**69.2**	**83.5**	94.1	**4.1**	**16.6**	26.7	41.5	6.3	24.4	46.8	71.7	1.9	6.5	**16.4**	99.1	1.5	**6.5**	**15.5**	99.9	**3.5**	15.9	41.3	**76.2**
DeepSN	7.4	40.7	68.2	**95.3**	2.9	14.4	25.3	**52.0**	**6.3**	**24.6**	**47.2**	**73.2**	**2.0**	**6.7**	14.9	96.9	**1.6**	5.8	13.5	**99.9**	3.2	**16.1**	**41.7**	72.1
DeepSN$_{SP}$	7.9	46.4	72.8	93.8	3.6	15.5	**27.8**	51.8	5.2	23.8	40.3	68.1	1.5	5.5	14.8	**99.5**	1.3	6.2	13.3	99.9	3.3	15.9	41.2	71.6

Table 7.3: Performance comparison under Linear Threshold (LT) diffusion model. - indicates out-of-memory error. The best results are highlighted in **bold**. Baseline results are sourced from Ling et al. (2023).

	Cora-ML				Network Science				Power Grid				Jazz				Random				Digg			
Methods	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%	1%	5%	10%	20%
IMM	2.0	9.5	15.4	27.6	1.3	5.6	12.2	22.1	1.1	5.6	11.0	22.9	7.6	37.8	55.6	67.1	2.7	12.6	20.9	37.7	2.5	9.4	16.3	32.6
OPIM	2.3	9.3	16.2	27.2	1.4	5.9	13.0	22.1	1.2	5.9	11.1	22.4	8.2	35.1	56.8	68.3	2.8	12.5	20.2	36.1	2.3	9.3	16.5	32.3
SubSIM	2.3	9.2	16.9	28.8	1.5	5.6	12.2	23.3	1.2	5.6	11.4	21.9	2.9	30.1	53.8	67.0	3.2	14.4	24.5	39.1	2.5	9.5	16.1	32.3
IMINfECTOR	2.1	9.4	16.1	27.9	1.7	5.8	12.4	22.3	1.3	5.5	10.2	23.1	8.8	35.4	54.8	66.2	2.5	12.4	20.5	36.6	2.3	9.1	16.4	32.4
DeepIM	7.1	16.1	21.9	30.8	2.7	8.7	15.1	25.1	1.9	7.6	13.3	23.8	27.1	57.1	68.1	74.1	3.2	14.4	24.5	39.1	5.6	11.4	18.8	**36.3**
DeepSN	12.8	24.7	34.2	46.5	2.0	9.6	16.1	28.1	**2.4**	**9.8**	**15.7**	25.3	**34.3**	**64.9**	**75.6**	**85.8**	**5.2**	**24.2**	**37.6**	**54.1**	**16.1**	20.2	**24.7**	33.2
DeepSN$_{SP}$	**16.8**	**29.9**	**37.9**	**46.9**	**2.7**	**9.9**	**17.4**	**29.0**	2.1	8.2	14.9	**26.3**	35.2	58.3	73.4	82.4	4.1	18.7	32.8	52.6	16.1	**20.3**	23.6	33.1

Table 7.4: Performance comparison under Susceptible-Infected-Susceptible (SIS) diffusion model. The best results are highlighted in **bold**. Baseline results are sourced from Ling et al. (2023).

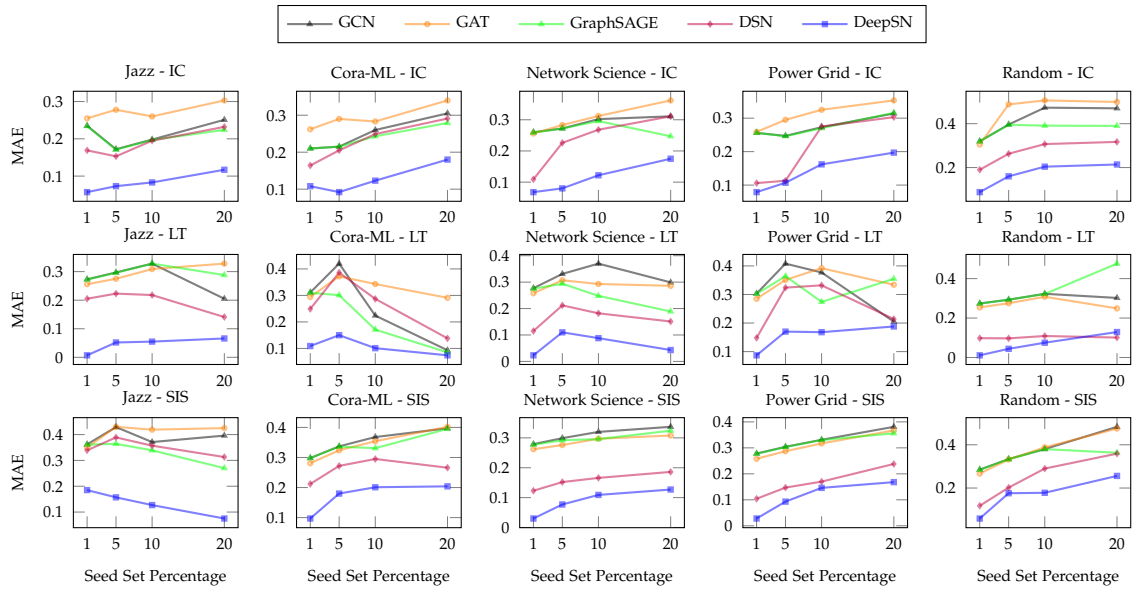


Figure 7.2: Performance of DeepSN for influence estimation in terms of MAE (Mean Absolute Error), compared to baseline methods, under IC, LT, and SIS diffusion models.

	IC		LT		SIS	
Methods	10%	20%	10%	20%	10%	20%
DeepSN-CELF	37.2	52.8	**76.0**	88.6	30.6	38.7
DeepSN-WC	39.2	50.4	46.3	88.6	25.3	35.9
DeepSN-WSA	40.0	50.6	48.0	89.6	26.0	36.0
DeepSN	40.9	52.8	68.2	**95.3**	34.2	46.5
DeepSN$_{SP}$	**42.3**	**58.4**	72.8	93.8	**37.9**	**46.9**

Table 7.5: Comparison of DeepSN with various influence maximization approaches for Cora-ML dataset.

rithm (Leskovec et al., 2007) as the IM component; **DeepSN-WC**, where seed nodes are optimized across the entire network as a single subgraph; and **DeepSN-WSA**, which uses the adjacency matrix for dividing the graph into subgraphs, instead of sheaf coefficients. The results in Table 7.5 show that our IM model outperforms other algorithms. Particularly, DeepSN and DeepSN$_{SP}$ consistently exceed the performance of DeepSN-WC and DeepSN-WSA. This is largely due to subgraph-based seed optimization and sheaf coefficients.

7.5.4 Impact of Layer Depth and Feature Dimension

We explore the effect of layer depth and feature dimension on the performance of DeepSN. Figure 7.3 illustrates how these factors influence the performance of DeepSN in selecting optimal seed nodes for Cora ML dataset, evaluated under the

IC diffusion model with a 10% seed set.

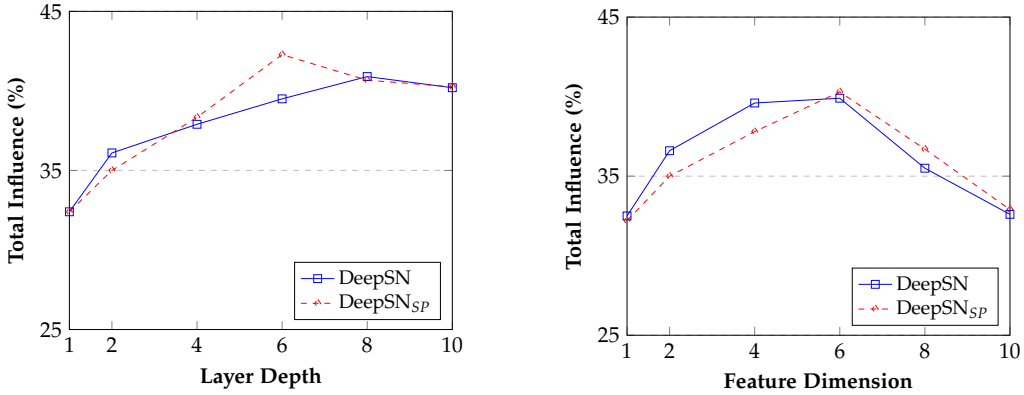


Figure 7.3: Impact of layer depth and feature dimension on DeepSN's performance

As expected, DeepSN's performance improves with greater layer depth, effectively capturing long-range interactions and demonstrating robustness against oversmoothing. However, performance only improves with increasing feature dimension up to a certain point. Beyond this, the added complexity, such as more learnable parameters, begins to outweigh the benefits of enhanced representational power, leading to a decline in performance.

7.5.5 Impact of Reaction Operators

Methods	1%	5%	10%	20%
Without Reaction Components	9.7	20.5	32.8	49.8
Only Pointwise Dynamics	10.9	23.2	34.0	51.0
Only Coupled Dynamics	10.7	22.4	37.9	50.7
With Reaction Components	**11.5**	**25.6**	**40.9**	**52.8**

Table 7.6: Impact of reaction terms on DeepSN performance.

Table 7.6 shows the impact of different reaction term configurations on DeepSN's performance. The table presents the average total influence(%) for the Cora-ML dataset across various seed set percentages. While each component offers improvements, the combined approach outperforms all other configurations.

7.6 Summary

In this chapter, we proposed a novel learning framework for the IM problem. Our approach integrates a GNN that harnesses sheaf theory to learn the underlying influence diffusion model in a data-driven manner, while effectively addressing the topological and dynamic complexities of propagation phenomena. Additionally, we proposed a subgraph-based maximization objective to identify the optimal seed set,

thereby reducing the combinatorial search space inherent to the IM problem. The empirical results demonstrated the effectiveness of the proposed framework.

The influence maximization framework discussed in this chapter has positive applications, such as targeted vaccination, resource allocation, and information dissemination. However, the same methods could also be misused for harmful purposes, such as spreading misinformation, manipulating public opinion through targeted social media campaigns, or amplifying extremist content by identifying key nodes in online networks. To ensure responsible use of these methods, it is essential to maintain transparency in selecting seed nodes and to adhere to relevant legal frameworks.

Chapter 8

Conclusions and Future Work

8.1 Summary of Contributions

In this thesis, we have explored fundamental challenges in graph neural networks, focusing on three key paradigms: (1) enhancing expressivity through structural learning, (2) adaptive information propagation mechanisms, and (3) modeling dynamic processes in graphs. We developed novel architectures and theoretical frameworks that advance the state-of-the-art in graph representation learning while addressing critical limitations of existing approaches. In summary, we have presented the following main contributions in Chapters 3–7:

- **Chapter 3: Structural Expressivity through Graph Partitioning.** We introduced permutation-invariant graph partitioning as a mechanism for capturing complex structural interactions in graphs. Our approach moves beyond traditional message-passing schemes by explicitly modeling how different structural components interact within a graph. We developed Graph Partitioning Neural Networks (GPNNs), establishing theoretical connections between graph partitioning and isomorphism tests. Our analysis demonstrates that GPNNs surpass the expressivity of the 1-Weisfeiler-Lehman test while efficiently approaching the capabilities of 3-WL. Through comprehensive complexity analysis, we showed that our approach balances computational efficiency with representational power. Empirical evaluation across graph classification, regression, and node classification tasks validates the effectiveness of incorporating structural interactions into neural architectures.

- **Chapter 4: Geometry-Informed Adaptive Depth Mechanisms.** We proposed a novel framework that integrates learnable Bakry-Émery curvature into graph neural networks to enable adaptive depth allocation. Unlike traditional curvature measures that focus solely on topology, Bakry-Émery curvature captures both structural properties and diffusion dynamics. We established theoretical connections between node curvature and feature distinctiveness, demonstrating that nodes with higher curvature require fewer message-passing iterations for effective representation learning. Our depth-adaptive mechanism dynamically adjusts aggregation depth for individual nodes based on their learned

curvature values. This approach serves as a model-agnostic enhancement that integrates seamlessly with existing GNN architectures including GCN, GAT, and GraphSAGE. Experimental results across node classification, regression, graph classification, and regression tasks demonstrate consistent performance improvements while maintaining computational efficiency comparable to standard GNNs.

- **Chapter 5: Theoretical Foundations for Adaptive Aggregation.** We developed a theoretical framework connecting neighborhood characteristics to optimal aggregation strategies in node classification. Our analysis reveals that strong heterophily is not inherently detrimental; rather, a node's aggregation requirements depend on the interplay between same-label neighbors, opposite-label neighbors, and degree within its local neighborhood. We derived signal preservation factors that quantify how neighborhood composition affects class-discriminative information during aggregation, extending this analysis to multi-layer settings. Building on these insights, we developed Adaptive Depth Graph Neural Networks (AD-GNN) that dynamically determine layer depth for individual nodes. Our framework provides a unified solution for both homophilic and heterophilic graphs, eliminating the need for separate architectures or preprocessing steps. Extensive experiments demonstrate that AD-GNN consistently enhances multiple GNN backbones across diverse graph structures.

- **Chapter 6: Opinion Dynamics for Graph Diffusion.** We introduced a generalized framework that unifies multiple opinion dynamics models to enhance diffusion-based graph neural networks. Traditional diffusion GNNs assume homogeneous propagation with static dynamics, limiting their ability to model complex real-world processes. Our Generalized Opinion Dynamics Neural Framework (GODNF) incorporates node-specific stubbornness parameters, dynamic neighborhood influence, and structural regularization to capture heterogeneous diffusion patterns. We provided rigorous theoretical analysis of convergence properties, proving that GODNF can exhibit diverse convergence configurations including single consensus, multi-consensus, and individualized consensus. This representational flexibility enables the model to adapt to various graph learning scenarios. The framework maintains computational efficiency by limiting the number of learnable parameters across layers while supporting deep architectures. Experimental validation on node classification and influence estimation tasks demonstrates substantial improvements over both spatial and diffusion-based GNN baselines.

- **Chapter 7: Sheaf Theory for Influence Maximization.** We developed Deep Sheaf Networks (DeepSN), a novel framework for influence maximization that leverages sheaf theory to model complex propagation dynamics. Traditional approaches either require explicit diffusion model specification or rely on standard GNN architectures that struggle with temporal dynamics. We formulated

influence propagation as a sheaf diffusion-reaction process, introducing pointwise and coupled dynamics operators to capture both intrinsic node transformations and neighborhood interactions. Our framework learns adaptive structural relationships through sheaf coefficients, enabling effective identification of diverse influence patterns including both progressive and non-progressive diffusion models. To address the combinatorial challenge of seed selection, we developed a subgraph-based optimization strategy that reduces the search space while accounting for overlapping influence. Comprehensive evaluation across multiple diffusion models demonstrates that DeepSN significantly outperforms traditional and learning-based baselines.

Collectively, these contributions establish a coherent framework that integrates structural, geometric, and dynamic perspectives to advance graph neural network design. Rather than treating expressivity, adaptive propagation, and temporal modeling as independent challenges, this thesis demonstrates their fundamental interconnections: structural expressivity provides the foundation for capturing complex graph patterns (Chapter 3), geometric principles guide how information should propagate through these structures (Chapters 4-5), and dynamic frameworks enable modeling of real-world processes that unfold over graphs (Chapters 6-7). This integrated approach not only enhances theoretical understanding of GNN capabilities but also provides practical mechanisms for building more adaptive and interpretable models that respond to the intrinsic properties of graph data.

8.2 Computational Cost and Practical Trade-offs

The methods proposed in this thesis are designed to balance representational power of GNNs with computational feasibility. We briefly summarise the cost-accuracy trade-offs across all contributions below.

- **Chapter 3: GPNN** All GPNN variants fall strictly between the $\mathcal{O}(|E|)$ complexity of 1-WL and the $\mathcal{O}(|V|^3)$ complexity of 3-WL. The key factor governing cost is the size of the interaction edge sets, which remain small on sparse real-world graphs, keeping the lighter variants close to linear in practice. The partitioning schemes used as preprocessing are themselves efficient (i.e., linear in the number of edges), ensuring graph partitioning adds negligible overhead to training.
- **Chapter 4: BEC-GNN** The dominant additional cost of BEC-GNN comes from computing the curvature approximation, which scales with the maximum node degree rather than the graph size, and is performed via lightweight MLPs that do not expand the GNN parameter space. Empirically, the parameter count and runtime grow only modestly as more curvature approximation functions are added, while accuracy improves consistently, confirming a favourable cost-accuracy trade-off.
- **Chapter 5: AD-GNN** AD-GNN adds per-node depth computation on top of the backbone GNN, with cost scaling linearly in the number of edges and feature

dimension. The fast variant replaces feature-based depth estimation with a degree-based approximation, reducing overhead to near-baseline levels while retaining most of the accuracy benefit.

- **Chapter 6: GODNF** GODNF's complexity scales linearly with the number of edges and is bounded by $\mathcal{O}(m)$ under practical assumptions on feature dimension and iteration count, matching the asymptotic cost of standard GNNs. The absence of intermediate feature transformations across iterative updates yields additional efficiency gains at greater depth compared to conventional architectures.
- **Chapter 7: DeepSN** The sheaf diffusion component introduces only a constant-factor overhead relative to a standard GNN, since the sheaf stalk dimension is kept small in practice. Community detection used as preprocessing runs in near-linear time. Crucially, the subgraph-based seed selection mechanism replaces an otherwise exponential combinatorial search with a learned graph partitioning, making influence maximisation tractable at scale without sacrificing solution quality.

Overall, the computational overhead introduced by each proposed method is moderate and well-justified by the corresponding empirical gains. Where computational budget is constrained, lighter variants are available across contributions and are specifically designed for scalable deployment.

8.3 Cross-Method Comparison

While each chapter addresses a distinct research objective, several contributions operate on overlapping tasks and datasets, enabling direct empirical comparison. In this section, we compare these results to highlight the relative strengths and complementary nature of the proposed methods.

8.3.1 Graph Classification: GPNN vs. BEC-GNN

Both GPNN and BEC-GNN are evaluated on graph classification benchmarks. GPNN improves expressivity through structural interaction modelling, while BEC-GNN improves aggregation depth through curvature-guided adaptation. Table 8.1 compares the two approaches on shared datasets. GPNN tends to show stronger gains due to its higher expressive power, while BEC-GNN provides consistent improvements as a plug-in to any backbone. The two methods are complementary and could, in principle, be combined.

8.3.2 Node Classification: BEC-GNN vs. AD-GNN vs. GODNF

BEC-GNN, AD-GNN and GODNF all evaluate on node classification. BEC-GNN adapts aggregation depth via curvature, AD-GNN adapts it via homophily, and

Table 8.1: Comparison of GPNN and BEC-GNN on graph classification benchmarks. We adopt the setup outlined by Feng et al. (2022). Best results are highlighted in **bold**.

Method	MUTAG	PTC-MR	PROTEINS	IMDB-B	BZR	COX2	ogbg-moltox21
GIN	92.80 ± 5.9	65.60 ± 6.5	78.80 ± 4.1	78.10 ± 3.5	91.05 ± 3.4	88.87 ± 2.3	74.91 ± 0.5
GIN + BEC	96.10 ± 3.6	72.90 ± 5.7	79.10 ± 3.7	**80.80 ± 3.3**	92.40 ± 3.6	89.30 ± 3.1	75.70 ± 0.7
$\lambda_{\text{core-degree}}$-GPNN*	**97.89 ± 2.6**	78.17 ± 6.2	81.40 ± 3.5	80.10 ± 3.1	**94.05 ± 2.6**	89.09 ± 3.3	**76.13 ± 0.68**
$\lambda_{\text{core-degree}}$-GPNN$^{\diamond}$	97.37 ± 4.9	**79.61 ± 7.3**	**85.53 ± 6.2**	78.10 ± 3.1	91.84 ± 1.6	**89.72 ± 2.3**	75.89 ± 0.30

GODNF models node classification as a diffusion process. Table 8.2 compares the three approaches on datasets, using GCN as a common backbone where applicable. All methods outperform GCN by significant margins. BEC-GNN performs well on homophilic datasets while AD-GNN excels on heterophilic ones. GODNF achieves strong performance across both settings due to its flexible diffusion mechanism.

Table 8.2: Cross-chapter comparison on node classification datasets. We adopt the setup outlined by Chen et al. (2025). Best results per dataset are in **bold**.

Method	Texas	Cornell	Wisconsin	Film	Citeseer	Cora-ML
GCN	60.00 ± 6.45	55.14 ± 8.46	61.60 ± 7.00	30.26 ± 0.79	76.68 ± 1.64	87.07 ± 1.21
GCN + BEC	68.85 ± 11.02	65.96 ± 9.03	66.25 ± 4.55	34.34 ± 2.08	78.14 ± 1.31	88.65 ± 1.32
AD-GCN	92.30 ± 4.52	**88.51 ± 4.87**	**93.88 ± 3.03**	**42.54 ± 1.15**	**79.14 ± 1.00**	87.32 ± 1.25
$\text{GODNF}_{\text{Dynamic}}$	**93.93 ± 2.65**	87.87 ± 4.57	92.50 ± 2.56	39.96 ± 2.21	78.85 ± 0.99	**89.26** ± 1.13

8.3.3 Influence Estimation: BEC-GNN vs. GODNF vs. DeepSN

BEC-GNN, GODNF, and DeepSN evaluate on influence estimation. Table 8.3 compares the three approaches at 10% seed set size. BEC-GNN improves over the GCN backbone by better capturing structural properties relevant to propagation. GODNF models the diffusion process explicitly through an opinion dynamics framework, while DeepSN formulates influence propagation as a sheaf reaction-diffusion process. BEC-GNN consistently improves the GCN baseline. GODNF leads under LT diffusion while DeepSN achieves stronger results under IC.

Table 8.3: Comparison on influence estimation MAE at 10% seed set under IC and LT diffusion models. We adopt the experimental setup oulined by Ling et al. (2023). Best results per dataset are in **bold**.

	Jazz		Cora-ML		Network Science		Power Grid	
Method	IC	LT	IC	LT	IC	LT	IC	LT
GCN	0.233 ± 0.010	0.199 ± 0.006	0.277 ± 0.007	0.255 ± 0.008	0.270 ± 0.019	0.190 ± 0.012	0.313 ± 0.024	0.335 ± 0.023
GCN + BEC	0.202 ± 0.007	0.124 ± 0.002	0.258 ± 0.006	0.194 ± 0.010	0.233 ± 0.010	0.123 ± 0.026	0.329 ± 0.019	0.298 ± 0.039
$\text{GODNF}_{\text{Dynamic}}$	0.175 ± 0.003	**0.050 ± 0.001**	0.148 ± 0.001	**0.094 ± 0.005**	0.197 ± 0.002	**0.032 ± 0.003**	0.221 ± 0.001	**0.108 ± 0.004**
DeepSN	**0.083 ± 0.004**	0.055 ± 0.003	**0.123 ± 0.005**	0.101 ± 0.004	**0.122 ± 0.006**	0.088 ± 0.005	**0.162 ± 0.007**	0.168 ± 0.008

Taken together, these comparisons confirm that the five contributions presented in this thesis address the same downstream tasks from distinct and complementary angles, each offering advantages in different regimes.

8.4 Limitations

While these contributions substantially advance current GNN methodologies, several limitations remain.

- **Chapter 3:** The expressivity gains of GPNN depends on the partitioning scheme producing meaningful structural separation. On highly regular graphs where all nodes share identical local structure, the interaction embeddings may carry little additional information beyond standard message passing. The graph partitioning schemes are determined through preprocessing rather than learned end-to-end. While we explored several permutation-invariant partitioning strategies based on structural properties such as k-core and degree, these schemes remain fixed during training.
- **Chapter 4:** The theoretical guarantees assume bounded node degree and well-behaved diffusion dynamics. On graphs with highly skewed degree distributions, curvature estimates for high-degree hub nodes may be less reliable, and the framework has not been studied under dynamic or noisy graph structures. The curvature-based depth allocation mechanism employs a fixed threshold hyperparameter to determine stopping depth for nodes.
- **Chapter 5:** Our theoretical analysis relies on several simplifying assumptions including binary classification and the contextual stochastic block model. The depth benefit metric may be less reliable where feature correlation across layers violates the independence assumption underlying the analysis.
- **Chapter 6:** The convergence guarantees of GODNF require the combined influence matrix to satisfy a spectral norm condition throughout training, which may not always hold on inference. The fixed topology assumption also limits applicability to settings where network structure evolves over time. Further, the framework currently uses a global feature retention parameter α that applies uniformly across all nodes.
- **Chapter 7:** The sheaf neural framework currently supports diffusion models with only two states (active and susceptible). Further, the subgraph-based seed selection relies on community structure identified by the Louvain algorithm, and may underperform on graphs without well-defined community organisation.

8.5 Future Work

The limitations identified above, alongside broader open questions in graph representation learning, motivate several directions for future research.

- **Chapter 3:** From a theoretical perspective, it would be valuable to investigate how optimal partitioning schemes might be learned directly from data. Future

work could investigate attention-based mechanisms and differentiable partitioning strategies that could adaptively identify structural components while preserving permutation invariance. Such approaches could potentially discover task-specific partitioning patterns that better align with downstream objectives.

- **Chapter 4:** While we demonstrated robustness across various threshold values, automatically learning this threshold from the graph structure and data distribution would enhance the method's applicability. Future work could investigate end-to-end learning strategies where the threshold adapts based on task requirements. Additionally, extending this approach to more challenging scenarios including transfer learning across different graph domains and out-of-distribution generalization where the test graphs exhibit significantly different structural properties than training graphs would demonstrate broader applicability.

- **Chapter 5:** While our empirical results demonstrate that the insights transfer to multi-class settings and real-world graphs, developing more comprehensive theoretical frameworks with relaxed assumptions would strengthen the foundation of our work. Furthermore, our neural architecture incorporates a learnable parameter λ that could potentially be learned in a data-driven manner rather than determined through hyperparameter search. Investigating neural architectures that jointly optimize this parameter alongside other model components presents an interesting direction for future research.

- **Chapter 6:** While this design ensures parameter efficiency and stable convergence, allowing node-specific retention parameters could capture more nuanced diffusion patterns. Investigating learning mechanisms that balance the expressiveness of node-level parameters with the stability guarantees provided by our current formulation would be valuable. Additionally, extending evaluation to challenging settings such as transfer learning and distribution shift would demonstrate the robustness of this approach across diverse graph learning scenarios.

- **Chapter 7:** Many real-world propagation phenomena involve multiple states and complex transition dynamics. For instance, epidemic models often include exposed, infected, and recovered states with varying transition probabilities (Tolles and Luong, 2020). Extending this framework to handle multi-state diffusion models would broaden its applicability to domains such as epidemiology and complex social contagion processes.

Beyond individual extensions, the five contributions in this thesis are thematically connected and present natural opportunities for integration. The curvature-based depth allocation of Chapter 4 and the homophily-driven depth adaptation of Chapter 5 both arrive at per-node propagation decisions through different lenses, and combining these signals could yield a more robust depth allocation mechanism that accounts for both geometric and label-structural properties simultaneously. The

graph partitioning framework of Chapter 3 could inform the community detection step in Chapter 7, replacing the Louvain preprocessing with a learned, task-aware partitioning that better reflects influence propagation boundaries. Similarly, the adaptive depth mechanisms of Chapters 4 and 5 could be incorporated into the opinion dynamics framework of Chapter 6, allowing the number of diffusion iterations to vary per node rather than being applied uniformly. Exploring these combinations presents a promising direction toward a unified, adaptive graph learning framework that jointly optimises expressivity, propagation depth, and dynamic process modelling.

Bibliography

Abbahaddou, Y.; Malliaros, F. D.; Lutzeyer, J. F.; and Vazirgiannis, M., 2025. Admp-gnn: Adaptive depth message passing gnn. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. (Cited on page 28.)

Abbe, E., 2018. Community detection and stochastic block models: recent developments. *Journal of Machine Learning Research*, 18, 177 (2018), 1–86. (Cited on page 112.)

Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Ver Steeg, G.; and Galstyan, A., 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*. (Cited on pages 7, 30, 63, and 84.)

Acemoglu, D. and Ozdaglar, A., 2011. Opinion dynamics and learning in social networks. *Dynamic Games and Applications*, 1, 1 (2011), 3–49. (Cited on page 26.)

Alvarez-Hamelin, J.; Dall'Asta, L.; Barrat, A.; and Vespignani, A., 2005. Large scale networks fingerprinting and visualization using the k-core decomposition. *Advances in neural information processing systems*, (2005). (Cited on page 41.)

Ambrosio, L.; Gigli, N.; and Savaré, G., 2015. Bakry–émery curvature-dimension condition and riemannian ricci curvature bounds. (2015). (Cited on pages 8, 53, and 54.)

Arcagni, A.; Grassi, R.; Stefani, S.; and Torriero, A., 2017. Higher order assortativity in complex networks. *European Journal of Operational Research*, 262, 2 (2017), 708–719. (Cited on page 83.)

Bailleul, I. and Bernicot, F., 2016. Heat semigroup and singular pdes. *Journal of Functional Analysis*, 270, 9 (2016), 3344–3452. (Cited on page 55.)

Bakry, D. and Émery, M., 2006. Diffusions hypercontractives. In *Séminaire de Probabilités XIX 1983/84: Proceedings*, 177–206. Springer. (Cited on page 59.)

Balaban, A. T., 1985. Applications of graph theory in chemistry. *Journal of chemical information and computer sciences*, 25, 3 (1985), 334–343. (Cited on page 1.)

Balcilar, M.; Renton, G.; Héroux, P.; Gaüzère, B.; Adam, S.; and Honeine, P., 2021. Analyzing the expressive power of graph neural networks in a spectral perspective. In *International conference on learning representations*. (Cited on page 4.)

Banerjee, S.; Jenamani, M.; and Pratihar, D. K., 2020. A survey on influence maximization in a social network. *Knowledge and Information Systems*, 62 (2020). (Cited on page 28.)

Barbero, F.; Bodnar, C.; de Ocáriz Borde, H. S.; Bronstein, M.; Veličković, P.; and Liò, P., 2022. Sheaf neural networks with connection laplacians. In *Topological, Algebraic and Geometric Learning Workshops*. (Cited on page 116.)

Barceló, P.; Geerts, F.; Reutter, J.; and Ryschkov, M., 2021. Graph neural networks with local graph parameters. *Advances in Neural Information Processing Systems*, 34 (2021), 25280–25293. (Cited on pages 7, 20, and 30.)

Beaini, D.; Passaro, S.; Ltourneau, V.; Hamilton, W.; Corso, G.; and Lio, P., 2021. Directional graph networks. In *International Conference on Machine Learning*. (Cited on page 43.)

Belkin, M. and Niyogi, P., 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14 (2001). (Cited on page 3.)

Bellman, R., 1947. On the boundedness of solutions of nonlinear differential and difference equations. *Transactions of the American Mathematical Society*, 62, 3 (1947). (Cited on page 121.)

Berahmand, K.; Daneshfar, F.; Salehi, E. S.; Li, Y.; and Xu, Y., 2024. Autoencoders and their applications in machine learning: a survey. *Artificial intelligence review*, 57, 2 (2024), 28. (Cited on page 3.)

Bevilacqua, B.; Frasca, F.; Lim, D.; Srinivasan, B.; Cai, C.; Balamurugan, G.; Bronstein, M. M.; and Maron, H., 2022. Equivariant subgraph aggregation networks. *International Conference on Learning Representations*, (2022). (Cited on pages 7, 21, 30, and 42.)

Bi, W.; Du, L.; Fu, Q.; Wang, Y.; Han, S.; and Zhang, D., 2024. Make heterophilic graphs better fit gnn: A graph rewiring approach. *IEEE Transactions on Knowledge and Data Engineering*, (2024). (Cited on page 23.)

Biondi, E.; Boldrini, C.; Passarella, A.; and Conti, M., 2023. Dynamics of opinion polarization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53, 9 (2023), 5381–5392. (Cited on page 26.)

Bisgin, H.; Agarwal, N.; and Xu, X., 2012. A study of homophily on social media. *World Wide Web*, 15, 2 (2012), 213–232. (Cited on page 22.)

Blondel, V. D.; Guillaume, J.-L.; Lambiotte, R.; and Lefebvre, E., 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008, 10 (2008). (Cited on page 123.)

Bo, D.; Wang, X.; Liu, Y.; Fang, Y.; Li, Y.; and Shi, C., 2023. A survey on spectral graph neural networks. *arXiv preprint arXiv:2302.05631*, (2023). (Cited on page 4.)

Bo, D.; Wang, X.; Shi, C.; and Shen, H., 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 3950–3957. (Cited on page 22.)

Bochner, S., 1946. Vector fields and ricci curvature. (1946). (Cited on pages 8, 24, and 52.)

Bodnar, C.; Di Giovanni, F.; Chamberlain, B.; Lio, P.; and Bronstein, M., 2022. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *Advances in Neural Information Processing Systems*, 35 (2022), 18527–18541. (Cited on pages 27, 116, 118, 119, 124, 125, 126, and 127.)

Bodnar, C.; Frasca, F.; Otter, N.; Wang, Y.; Lio, P.; Montufar, G. F.; and Bronstein, M., 2021. Weisfeiler and lehman go cellular: Cw networks. *Advances in neural information processing systems*, 34 (2021), 2625–2640. (Cited on pages xiii, 20, 30, 42, 43, 44, and 45.)

Bojchevski, A. and Günnemann, S., 2018. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*. (Cited on pages 84 and 103.)

Bose, K.; Banerjee, S.; and Das, S., 2025. Can graph neural networks tackle heterophily? yes, with a label-guided graph rewiring approach! *IEEE Transactions on Neural Networks and Learning Systems*, (2025). (Cited on page 23.)

Bouritsas, G.; Frasca, F.; Zafeiriou, S. P.; and Bronstein, M., 2022. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2022). (Cited on pages 7, 20, and 30.)

Bredon, G. E., 2012. *Sheaf theory*, vol. 170. Springer Science & Business Media. (Cited on pages 12 and 116.)

Brody, S.; Alon, U.; and Yahav, E., 2022. How attentive are graph attention networks? In *International Conference on Learning Representations*. (Cited on pages xiii, 4, 28, 64, and 84.)

Cai, J.-Y.; Furer, M.; and Immerman, N., 1992. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12, 4 (1992), 389–410. (Cited on pages 21 and 32.)

Cai, T.; Luo, S.; Xu, K.; He, D.; Liu, T.-y.; and Wang, L., 2021. Graphnorm: A principled approach to accelerating graph neural network training. In *International Conference on Machine Learning*, 1204–1215. PMLR. (Cited on page 4.)

Chamberlain, B.; Rowbottom, J.; Gorinova, M. I.; Bronstein, M.; Webb, S.; and Rossi, E., 2021. Grand: Graph neural diffusion. In *International conference on machine learning*, 1407–1418. PMLR. (Cited on pages 5, 10, 18, 93, and 104.)

Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; and Sun, X., 2020a. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI conference on artificial intelligence*, vol. 34. (Cited on page 116.)

Chen, J.; Deng, B.; Chen, C.; Zheng, Z.; et al., 2025. Graph neural ricci flow: Evolving feature from a curvature perspective. In *The Thirteenth International Conference on Learning Representations*. (Cited on pages xiv, xv, 25, 86, 104, 105, 106, and 137.)

Chen, K.; Liu, S.; Zhu, T.; Qiao, J.; Su, Y.; Tian, Y.; Zheng, T.; Zhang, H.; Feng, Z.; Ye, J.; et al., 2023a. Improving expressivity of gnns with subgraph-specific factor embedded normalization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 237–249. (Cited on page 4.)

Chen, S.; Zhang, C.; Gu, F.; and Wang, H., 2024. Rsgnn: residual structure graph neural network. *International Journal of Machine Learning and Cybernetics*, 15, 9 (2024), 4079–4092. (Cited on page 4.)

Chen, T.; Yan, S.; Guo, J.; and Wu, W., 2023b. Touplegdd: A fine-designed solution of influence maximization by deep reinforcement learning. *IEEE Transactions on Computational Social Systems*, 11, 2 (2023). (Cited on pages 28, 116, and 127.)

Chen, W.; Castillo, C.; and Lakshmanan, L. V., 2022a. *Information and influence propagation in social networks*. Springer Nature. (Cited on page 120.)

Chen, W.; Peng, B.; Schoenebeck, G.; and Tao, B., 2022b. Adaptive greedy versus non-adaptive greedy for influence maximization. *Journal of Artificial Intelligence Research*, (2022). (Cited on page 120.)

Chen, X.; Zhang, X.; Xie, Y.; and Li, W., 2017. Opinion dynamics of social-similarity-based hegselmann–krause model. *Complexity*, 2017, 1 (2017), 1820257. (Cited on page 27.)

Chen, Z.; Chen, L.; Villar, S.; and Bruna, J., 2020b. Can graph neural networks count substructures? *Advances in neural information processing systems*, (2020). (Cited on page 43.)

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O., 2021. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*. (Cited on page 17.)

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bougares, F.; Schwenk, H.; and Bengio, Y., 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. (Cited on page 4.)

Choi, J. et al., 2023. Gread: Graph neural reaction-diffusion networks. In *International Conference on Machine Learning*, 5722–5747. PMLR. (Cited on pages 5 and 119.)

Ciotti, V.; Bonaventura, M.; Nicosia, V.; Panzarasa, P.; and Latora, V., 2016. Homophily and missing links in citation networks. *EPJ Data Science*, 5, 1 (2016), 7. (Cited on page 22.)

Cotta, L.; Morris, C.; and Ribeiro, B., 2021. Reconstruction for powerful graph representations. *Advances in Neural Information Processing Systems*, (2021). (Cited on pages 7, 21, and 30.)

Craven, M.; DiPasquo, D.; Freitag, D.; McCallum, A.; Mitchell, T.; Nigam, K.; and Slattery, S., 2000. Learning to construct knowledge bases from the world wide web. *Artificial intelligence*, 118, 1-2 (2000), 69–113. (Cited on page 42.)

Cushing, D.; Liu, S.; and Peyerimhoff, N., 2020. Bakry–émery curvature functions on graphs. *Canadian Journal of Mathematics*, 72, 1 (2020), 89–143. (Cited on page 53.)

Dan, T.; Ding, J.; Wei, Z.; Kovalsky, S.; Kim, M.; Kim, W. H.; and Wu, G., 2023. Rethink and re-design graph neural networks in spaces of continuous graph diffusion functionals. In *Advances in Neural Information Processing Systems*, vol. 36, 59375–59387. (Cited on page 5.)

Das, A.; Gollapudi, S.; and Munagala, K., 2014. Modeling opinion dynamics in social networks. In *Proceedings of the 7th ACM international conference on Web search and data mining*, 403–412. (Cited on pages 11, 26, and 94.)

DeGroot, M. H., 1974. Reaching a consensus. *Journal of the American Statistical association*, 69, 345 (1974), 118–121. (Cited on pages 10, 26, 94, and 101.)

Deshpande, Y.; Sen, S.; Montanari, A.; and Mossel, E., 2018. Contextual stochastic block models. *Advances in Neural Information Processing Systems*, 31 (2018). (Cited on page 74.)

Devriendt, K. and Lambiotte, R., 2022. Discrete curvature on graphs from the effective resistance. *Journal of Physics: Complexity*, 3, 2 (2022), 025008. (Cited on page 68.)

Dugué, N. and Perez, A., 2015. *Directed Louvain: maximizing modularity in directed networks*. Ph.D. thesis, Université d'Orléans. (Cited on page 123.)

Dwivedi, V. P.; Joshi, C. K.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X., 2023. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24, 43 (2023), 1–48. (Cited on pages 42, 43, 62, and 63.)

d'Onofrio, A., 2008. A note on the global behaviour of the network-based sis epidemic model. *Nonlinear Analysis: Real World Applications*, 9, 4 (2008). (Cited on page 19.)

Eliasof, M.; Haber, E.; and Treister, E., 2021. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. *Advances in neural information processing systems*, 34 (2021), 3836–3849. (Cited on page 18.)

Errica, F.; Christiansen, H.; Zaverkin, V.; Maruyama, T.; Niepert, M.; and Alesiani, F., 2023. Adaptive message passing: A general framework to mitigate oversmoothing, oversquashing, and underreaching. *arXiv preprint arXiv:2312.16560*, (2023). (Cited on page 28.)

Errica, F.; Podda, M.; Bacciu, D.; Micheli, A.; et al., 2020. A fair comparison of graph neural networks for graph classification. In *Proceedings of the Eighth International Conference on Learning Representations (ICLR 2020)*. (Cited on page 63.)

Evans, L. C., 2022. *Partial differential equations*, vol. 19. American Mathematical Society. (Cited on page 56.)

Fan, C.; Jiang, Y.; and Mostafavi, . A., 2021. The role of local influential users in spread of situational crisis information. *Journal of Computer-Mediated Communication*, 26, 2 (2021). (Cited on page 115.)

Feng, J.; Chen, Y.; Li, F.; Sarkar, A.; and Zhang, M., 2022. How powerful are k-hop message passing graph neural networks. *Advances in Neural Information Processing Systems*, (2022). (Cited on pages xiii, xiv, xv, 7, 21, 30, 42, 43, 44, 49, 63, 66, and 137.)

Fesser, L.; de Haro Iványez, S. S.; Devriendt, K.; Weber, M.; and Lambiotte, R., 2024. Augmentations of forman's ricci curvature and their applications in community detection. *Journal of Physics: Complexity*, 5, 3 (2024), 035010. (Cited on pages 24 and 25.)

Fesser, L. and Weber, M., 2024. Mitigating over-smoothing and over-squashing using augmentations of forman-ricci curvature. In *Learning on Graphs Conference*, 19–1. PMLR. (Cited on pages 8, 24, 25, 52, and 68.)

Finkelshtein, B.; Huang, X.; Bronstein, M. M.; and Ceylan, I. I., 2024. Cooperative graph neural networks. In *Forty-first International Conference on Machine Learning*. (Cited on page 28.)

Fisher, R. A., 1936. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7, 2 (1936), 179–188. (Cited on page 73.)

Forman, 2003. Bochner's method for cell complexes and combinatorial ricci curvature. *Discrete & Computational Geometry*, 29 (2003), 323–374. (Cited on page 25.)

Forman, R. et al., 1999. Combinatorial differential topology and geometry. *New perspectives in geometric combinatorics*, 38 (1999), 177–206. (Cited on pages 8, 24, and 52.)

Frasca, F.; Bevilacqua, B.; Bronstein, M.; and Maron, H., 2022. Understanding and extending subgraph gnns by rethinking their symmetries. *Advances in Neural Information Processing Systems*, 35 (2022), 31376–31390. (Cited on page 21.)

Friedkin, N. E. and Johnsen, E. C., 1990. Social influence and opinions. *Journal of mathematical sociology*, 15, 3-4 (1990), 193–206. (Cited on pages 10, 26, and 94.)

Gallicchio, C. and Micheli, A., 2010. Graph echo state networks. In *The 2010 international joint conference on neural networks (IJCNN)*, 1–8. IEEE. (Cited on page 16.)

Gasteiger, J.; Bojchevski, A.; and Günnemann, S., 2018. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*. (Cited on pages 17, 27, and 104.)

Gasteiger, J.; Weissenberger, S.; and Günnemann, S., 2019. Diffusion improves graph learning. *Advances in Neural Information Processing Systems*, 32 (2019). (Cited on pages 5, 10, 17, 93, and 104.)

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E., 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, 1263–1272. PMLR. (Cited on pages 6 and 30.)

Giraldo, J. H.; Skianis, K.; Bouwmans, T.; and Malliaros, F. D., 2023. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, 566–576. (Cited on page 68.)

Goldenberg, J.; Libai, B.; and Muller, E., 2001a. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing letters*, 12 (2001), 211–223. (Cited on pages 63 and 104.)

Goldenberg, J.; Libai, B.; and Muller, E., 2001b. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing Science Review*, 9, 3 (2001). (Cited on page 19.)

Gori, M.; Monfardini, G.; and Scarselli, F., 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.*, vol. 2, 729–734. IEEE. (Cited on pages 3 and 16.)

Granovetter, M., 1978. Threshold models of collective behavior. *American journal of sociology*, 83, 6 (1978). (Cited on pages 19 and 63.)

Grover, A. and Leskovec, J., 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 855–864. (Cited on page 3.)

Guang, M.; Yan, C.; Xu, Y.; Wang, J.; and Jiang, C., 2024. Graph convolutional networks with adaptive neighborhood awareness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2024). (Cited on page 28.)

Guo, J.; Du, L.; Bi, W.; Fu, Q.; Ma, X.; Chen, X.; Han, S.; Zhang, D.; and Zhang, Y., 2023. Homophily-oriented heterogeneous graph rewiring. In *Proceedings of the ACM web conference 2023*, 511–522. (Cited on page 23.)

Guo, J.; Huang, K.; Zhang, R.; and Yi, X., 2024. Es-gnn: Generalizing graph neural networks beyond homophily with edge splitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2024). (Cited on page 22.)

Guo, Q.; Wang, S.; Wei, Z.; and Chen, M., 2020. Influence maximization revisited: Efficient reverse reachable set generation with bound tightened. In *ACM SIGMOD international conference on management of data*. (Cited on page 127.)

Hamilton, W.; Ying, Z.; and Leskovec, J., 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30 (2017). (Cited on pages 4, 16, 43, 58, 63, 82, 84, 104, and 127.)

Hamilton, W. L., 2020. *Graph representation learning*. Morgan & Claypool Publishers. (Cited on page 1.)

Han, A.; Shi, D.; Lin, L.; and Gao, J., 2023a. From continuous dynamics to graph neural networks: Neural diffusion and beyond. *Transactions on Machine Learning Research*, (2023). (Cited on page 93.)

Han, A.; Shi, D.; Lin, L.; and Gao, J., 2024. From continuous dynamics to graph neural networks: Neural diffusion and beyond. *Transactions on Machine Learning Research*, (2024). (Cited on page 17.)

Han, H.; Liu, X.; Mao, H.; Torkamani, M.; Shi, F.; Lee, V.; and Tang, J., 2023b. Alternately optimized graph neural networks. In *International Conference on Machine Learning*, 12411–12429. PMLR. (Cited on pages xiii and 64.)

Hansen, J. and Gebhart, T., 2020. Sheaf neural networks. *arXiv preprint arXiv:2012.06333*, (2020). (Cited on pages 116 and 119.)

Hansen, J. and Ghrist, R., 2021. Opinion dynamics on discourse sheaves. *SIAM Journal on Applied Mathematics*, 81, 5 (2021). (Cited on page 124.)

Harrison, R. L., 2010. Introduction to monte carlo simulation. In *AIP conference proceedings*, vol. 1204, 17. (Cited on page 28.)

Haruta, S.; Konishi, T.; and Kurokawa, M., 2023. A novel graph aggregation method based on feature distribution around each ego-node for heterophily. In *Asian Conference on Machine Learning*, 452–466. PMLR. (Cited on page 23.)

Hassani, H.; Razavi-Far, R.; Saif, M.; Chiclana, F.; Krejcar, O.; and Herrera-Viedma, E., 2022. Classical dynamic consensus and opinion dynamics models: A survey of recent trends and methodologies. *Information Fusion*, 88 (2022), 22–40. (Cited on pages 11 and 94.)

He, D.; Liang, C.; Liu, H.; Wen, M.; Jiao, P.; and Feng, Z., 2022. Block modeling-guided graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, 4022–4029. (Cited on page 22.)

Hevapathige, A. and Wang, Q., 2026. Permutation-invariant graph partitioning: How graph neural networks capture structural interactions? *Neural Networks*, (2026), 108869. (Cited on page iv.)

Hevapathige, A.; Wang, Q.; and Zehmakan, A. N., 2025a. Deepsn: a sheaf neural framework for influence maximization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, 17177–17185. (Cited on page iv.)

Hevapathige, A.; Wijesinghe, A.; and Zehmakan, A. N., 2026. Beyond fixed depth: Adaptive graph neural networks for node classification under varying homophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, 21717–21725. (Cited on page iv.)

Hevapathige, A.; Zehmakan, A. N.; and Wang, Q., 2025b. Depth-adaptive graph neural networks via learnable bakry-'emery curvature. *31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, (2025). (Cited on pages iv and 104.)

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, 9, 8 (1997), 1735–1780. (Cited on page 16.)

Horn, M.; De Brouwer, E.; Moor, M.; Moreau, Y.; Rieck, B.; and Borgwardt, K., 2022. Topological graph neural networks. *International Conference on Learning Representations*, (2022). (Cited on pages 20 and 30.)

Horn, R. A. and Johnson, C. R., 2012. *Matrix analysis*. Cambridge university press. (Cited on page 99.)

Hosseini-Pozveh, M.; Zamanifar, K.; and Naghsh-Nilchi, A. R., 2017. A community-based approach to identify the most influential nodes in social networks. *Journal of Information Science*, 43, 2 (2017). (Cited on page 115.)

Hssayni, E. h.; Boufssasse, A.; Joudar, N.-E.; and Ettaouil, M., 2025. Novel dropout approach for mitigating over-smoothing in graph neural networks. *Applied Intelligence*, 55, 5 (2025), 354. (Cited on page 23.)

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J., 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, (2020). (Cited on pages xiii, xiv, 42, 43, 44, 46, 62, 63, 66, 84, and 103.)

Huang, N. T. and Villar, S., 2021. A short tutorial on the weisfeiler-lehman test and its variants. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8533–8537. IEEE. (Cited on page 37.)

Huang, Y.; Peng, X.; Ma, J.; and Zhang, M., 2023. Boosting the cycle counting power of graph neural networks with i^2-gnns. *International Conference on Learning Representations*, (2023). (Cited on pages 21 and 30.)

Huisken, G., 1985. Ricci deformation of the metric on a riemannian manifold. *Journal of Differential Geometry*, 21, 1 (1985), 47–62. (Cited on page 24.)

Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; and Coleman, R. G., 2012. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52, 7 (2012), 1757–1768. (Cited on page 45.)

Jin, E.; Bronstein, M. M.; Ceylan, I. I.; and Lanzinger, M., 2024. Homomorphism counts for graph neural networks: All about that basis. In *Forty-first International Conference on Machine Learning*. (Cited on pages 21 and 30.)

Jin, Y. and Zhu, X., 2025. Oversmoothing alleviation in graph neural networks: a survey and unified view. *Knowledge and Information Systems*, (2025), 1–27. (Cited on page 23.)

Jordan, M. I.; Ghahramani, Z.; Jaakkola, T. S.; and Saul, L. K., 1999. An introduction to variational methods for graphical models. *Machine learning*, 37, 2 (1999), 183–233. (Cited on page 3.)

Kaveh, A. and Rahami, H., 2008. Factorization for efficient solution of eigenproblems of adjacency and laplacian matrices for graph products. *International journal for numerical methods in engineering*, 75, 1 (2008), 58–82. (Cited on page 3.)

Kelesis, D.; Fotakis, D.; and Paliouras, G., 2025. Analyzing the effect of residual connections to oversmoothing in graph neural networks. *Machine Learning*, 114, 8 (2025), 184. (Cited on page 4.)

Kempe, D.; Kleinberg, J.; and Tardos, É., 2003. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 137–146. (Cited on pages 104 and 115.)

Keriven, N., 2022. Not too little, not too much: a theoretical analysis of graph (over) smoothing. *Advances in Neural Information Processing Systems*, 35 (2022), 2268–2281. (Cited on page 23.)

Keriven, N. and Peyré, G., 2019. Universal invariant and equivariant graph neural networks. *Advances in neural information processing systems*, 32 (2019). (Cited on page 16.)

Ketkar, N. and Moolayil, J., 2021. Convolutional neural networks. In *Deep learning with Python: learn best practices of deep learning models with PyTorch*, 197–242. Springer. (Cited on page 2.)

Khanam, K. Z.; Srivastava, G.; and Mago, V., 2023. The homophily principle in social network analysis: A survey. *Multimedia Tools and Applications*, 82, 6 (2023), 8811–8854. (Cited on page 21.)

Khoshraftar, S. and An, A., 2024. A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology*, 15, 1 (2024), 1–55. (Cited on page 9.)

Kiefer, S., 2020. *Power and limits of the Weisfeiler-Leman algorithm*. Ph.D. thesis, Dissertation, RWTH Aachen University, 2020. (Cited on page 21.)

Kimura, D. and Hayakawa, Y., 2008. Coevolutionary networks with homophily and heterophily. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 78, 1 (2008), 016103. (Cited on page 22.)

Kimura, M.; Saito, K.; and Motoda, H., 2009. Efficient estimation of influence functions for sis model on social networks. In *IJCAI*, 2046–2051. (Cited on page 104.)

Kingma, D., 2015. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*. (Cited on pages 44, 64, 85, 105, and 128.)

Kipf, T. N. and Welling, M., 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, (2016). (Cited on page 124.)

Kipf, T. N. and Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*. (Cited on pages 4, 6, 16, 30, 36, 43, 58, 63, 73, 81, 84, 93, 100, 104, and 127.)

Kruse, R.; Mostaghim, S.; Borgelt, C.; Braune, C.; and Steinbrecher, M., 2022. Multi-layer perceptrons. In *Computational intelligence: a methodological introduction*, 53–124. Springer. (Cited on page 17.)

Kumar, S.; Mallik, A.; Khetarpal, A.; and Panda, B. S., 2022. Influence maximization in social networks using graph embedding and graph neural network. *Information Sciences*, (2022). (Cited on pages 11, 28, 104, 115, and 116.)

Lancichinetti, A. and Fortunato, S., 2009. Community detection algorithms: a comparative analysis. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 80, 5 (2009). (Cited on page 124.)

Laugwitz, D., 2014. *Differential and Riemannian geometry*. Academic press. (Cited on page 24.)

Lee, J. B.; Rossi, R. A.; Kim, S.; Ahmed, N. K.; and Koh, E., 2019. Attention models in graphs: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13, 6 (2019). (Cited on page 115.)

Lee, J.-w. and Jung, J., 2023. Time-aware random walk diffusion to improve dynamic graph learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 8473–8481. (Cited on page 93.)

Lee, S. Y.; Bu, F.; Yoo, J.; and Shin, K., 2023. Towards deep attention in graph neural networks: Problems and remedies. In *International Conference on Machine Learning*, 18774–18795. PMLR. (Cited on page 27.)

Leonard, N. E.; Bizyaeva, A.; and Franci, A., 2024. Fast and flexible multiagent decision-making. *Annual Review of Control, Robotics, and Autonomous Systems*, 7 (2024). (Cited on page 27.)

Lerman, K. and Galstyan, A., 2008. Analysis of social voting patterns on digg. In *workshop on Online social networks*. (Cited on page 127.)

Leskovec, J.; Krause, A.; Guestrin, C.; Faloutsos, C.; VanBriesen, J.; and Glance, N., 2007. Cost-effective outbreak detection in networks. In *13th ACM SIGKDD international conference on Knowledge discovery and data mining*. (Cited on pages 115 and 130.)

Li, B.; Pan, E.; and Kang, Z., 2024a. Pc-conv: Unifying homophily and heterophily with two-fold filtering. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, 13437–13445. (Cited on page 22.)

Li, G.; Müller, M.; Ghanem, B.; and Koltun, V., 2021a. Training graph neural networks with 1000 layers. In *International conference on machine learning*, 6437–6449. PMLR. (Cited on page 93.)

Li, H.; Cao, J.; Zhu, J.; Liu, Y.; Zhu, Q.; and Wu, G., 2022a. Curvature graph neural network. *Information Sciences*, 592 (2022), 50–66. (Cited on page 52.)

Li, H.; Jiang, H.; Zheng, Y.; Sun, H.; and Gong, W., 2025. Unigo: A unified graph neural network for modeling opinion dynamics on graphs. In *Proceedings of the ACM on Web Conference 2025*, 530–540. (Cited on page 104.)

Li, H.; Xu, M.; Bhowmick, S. S.; Rayhan, J. S.; Sun, C.; and Cui, J., 2022b. Piano: Influence maximization meets deep reinforcement learning. *IEEE Transactions on Computational Social Systems*, 10, 3 (2022). (Cited on pages 28, 116, and 127.)

Li, H.; Yang, S.; Xu, M.; Bhowmick, S. S.; and Cui, J., 2023a. Influence maximization in social networks: A survey. *arXiv preprint arXiv:2309.04668*, (2023). (Cited on page 104.)

Li, S.; Kim, D.; and Wang, Q., 2023b. Restructuring graph for higher homophily via adaptive spectral clustering. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, 8622–8630. (Cited on page 23.)

Li, X.; Smith, J. D.; Dinh, T. N.; and Thai, M. T., 2019. Tiptop:(almost) exact solutions for influence maximization in billion-scale networks. *IEEE/ACM Transactions on Networking*, 27, 2 (2019). (Cited on page 115.)

Li, X.; Sun, L.; Ling, M.; and Peng, Y., 2023c. A survey of graph neural network based recommendation in social networks. *Neurocomputing*, 549 (2023), 126441. (Cited on page 115.)

Li, X.; Zhu, R.; Cheng, Y.; Shan, C.; Luo, S.; Li, D.; and Qian, W., 2022c. Finding global homophily in graph neural networks when meeting heterophily. In *International conference on machine learning*, 13242–13256. PMLR. (Cited on page 22.)

Li, Y.; Fan, J.; Wang, Y.; and Tan, K.-L., 2018. Influence maximization on social graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 30, 10 (2018), 1852–1872. (Cited on pages 11, 19, 28, and 126.)

Li, Y.; Wang, X.; Liu, H.; and Shi, C., 2024b. A generalized neural diffusion framework on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 8707–8715. (Cited on pages 18, 104, and 109.)

Li, Y.; Zemel, R.; Brockschmidt, M.; and Tarlow, D., 2016. Gated graph sequence neural networks. In *Proceedings of ICLR'16*. (Cited on page 4.)

Li, Y.-M.; Lai, C.-Y.; and Lin, C.-H., 2009. Discovering influential nodes for viral marketing. In *2009 42nd Hawaii International Conference on System Sciences*. IEEE. (Cited on page 115.)

Li, Z.; Liu, F.; Yang, W.; Peng, S.; and Zhou, J., 2021b. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33, 12 (2021), 6999–7019. (Cited on page 2.)

Liang, F.; Qian, C.; Yu, W.; Griffith, D.; and Golmie, N., 2022. Survey of graph neural networks and applications. *Wireless Communications and Mobile Computing*, 2022, 1 (2022), 9261537. (Cited on pages 1 and 3.)

Lim, D.; Hohne, F.; Li, X.; Huang, S. L.; Gupta, V.; Bhalerao, O.; and Lim, S. N., 2021. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in neural information processing systems*, 34 (2021), 20887–20902. (Cited on page 22.)

Lin, J.; Guo, X.; Zhang, S.; Zhu, Y.; and Shun, J., 2025. When heterophily meets heterogeneity: Challenges and a new large-scale graph benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 5607–5618. (Cited on page 22.)

Lin, Y.; Lu, L.; and Yau, S.-T., 2011. Ricci curvature of graphs. *Tohoku Mathematical Journal, Second Series*, 63, 4 (2011), 605–627. (Cited on pages 24 and 52.)

Lin, Y. and Yau, S.-T., 2010. Ricci curvature and eigenvalue estimate on locally finite graphs. *Mathematical research letters*, 17, 2 (2010), 343–356. (Cited on pages 55 and 56.)

Ling, C.; Jiang, J.; Wang, J.; Thai, M. T.; Xue, R.; Song, J.; Qiu, M.; and Zhao, L., 2023. Deep graph representation learning and optimization for influence maximization. In *International Conference on Machine Learning*. (Cited on pages xv, 11, 19, 28, 29, 63, 104, 115, 127, 129, and 137.)

Liu, M.; Gao, H.; and Ji, S., 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 338–348. (Cited on page 93.)

Liu, M.; Yu, H.; and Ji, S., 2024. Empowering gnns via edge-aware weisfeiler-leman algorithm. *Transactions on Machine Learning Research*, (2024). (Cited on page 63.)

Liu, X.; Ding, J.; Jin, W.; Xu, H.; Ma, Y.; Liu, Z.; and Tang, J., 2021a. Graph neural networks with adaptive residual. *Advances in Neural Information Processing Systems*, 34 (2021), 9720–9733. (Cited on pages 4, 27, and 104.)

Liu, X.; Yan, M.; Deng, L.; Li, G.; Ye, X.; and Fan, D., 2021b. Sampling methods for efficient training of graph convolutional networks: A survey. *IEEE/CAA Journal of Automatica Sinica*, 9, 2 (2021), 205–234. (Cited on page 4.)

Liu, Y.; Zhou, C.; Pan, S.; Wu, J.; Li, Z.; Chen, H.; and Zhang, P., 2023. Curvdrop: A ricci curvature based approach to prevent graph neural networks from over-smoothing and over-squashing. In *Proceedings of the ACM Web Conference 2023*, 221–230. (Cited on pages 8, 24, 25, and 52.)

Lou, V. Y.; Bhagat, S.; Lakshmanan, L. V.; and Vaswani, S., 2014. Modeling non-progressive phenomena for influence propagation. In *ACM conference on Online social networks*. (Cited on page 120.)

Lu, H.; Wang, L.; Ma, X.; Cheng, J.; and Zhou, M., 2025. A survey of graph neural networks and their industrial applications. *Neurocomputing*, 614 (2025), 128761. (Cited on page 4.)

Luan, S.; Hua, C.; Lu, Q.; Zhu, J.; Zhao, M.; Zhang, S.; Chang, X.-W.; and Precup, D., 2022. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35 (2022), 1362–1375. (Cited on pages 9, 22, and 71.)

Luo, B.; Wilson, R. C.; and Hancock, E. R., 2003. Spectral embedding of graphs. *Pattern recognition*, 36, 10 (2003), 2213–2230. (Cited on page 3.)

Luo, D.; Cheng, W.; Yu, W.; Zong, B.; Ni, J.; Chen, H.; and Zhang, X., 2021. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*, 779–787. (Cited on page 23.)

Ma, L.; Shao, Z.; Li, X.; Lin, Q.; Li, J.; Leung, V. C.; and Nandi, A. K., 2022a. Influence maximization in complex networks by using evolutionary deep reinforcement learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7, 4 (2022). (Cited on page 28.)

Ma, Y.; Liu, X.; Shah, N.; and Tang, J., 2022b. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*. (Cited on pages 23, 73, and 74.)

Malliaros, F. D.; Giatsidis, C.; Papadopoulos, A. N.; and Vazirgiannis, M., 2020. The core decomposition of networks: Theory, algorithms and applications. *The VLDB Journal*, 29, 1 (2020), 61–92. (Cited on page 40.)

Mannan, M. A.; Rahman, M. R.; Akter, H.; Nahar, N.; and Mondal, S., 2021. A study of banach fixed point theorem and it's applications. *American Journal of Computational Mathematics*, 11, 2 (2021), 157–174. (Cited on page 99.)

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y., 2019a. Provably powerful graph networks. *Advances in neural information processing systems*, (2019). (Cited on pages 40, 42, and 43.)

Maron, H.; Ben-Hamu, H.; Shamir, N.; and Lipman, Y., 2018. Invariant and equivariant graph networks. In *International Conference on Learning Representations*. (Cited on page 16.)

Maron, H.; Fetaya, E.; Segol, N.; and Lipman, Y., 2019b. On the universality of invariant networks. (2019). (Cited on page 21.)

Marquetoux, N.; Stevenson, M. A.; Wilson, P.; Ridler, A.; and Heuer, C., 2016. Using social network analysis to inform disease control interventions. *Preventive Veterinary Medicine*, 126 (2016). (Cited on page 115.)

McCallum, A. K.; Nigam, K.; Rennie, J.; and Seymore, K., 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3 (2000). (Cited on pages 62 and 127.)

McKay, B. D. and Piperno, A., 2014. Practical graph isomorphism, ii. *Journal of symbolic computation*, 60 (2014), 94–112. (Cited on page 37.)

McPherson, M.; Smith-Lovin, L.; and Cook, J. M., 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27, 1 (2001), 415–444. (Cited on pages 9 and 71.)

Medsker, L. and Jain, L. C., 1999. *Recurrent neural networks: design and applications*. CRC press. (Cited on page 2.)

Mondal, M.; Samal, A.; Münch, F.; and Jost, J., 2024. Bakry–émery–ricci curvature: an alternative network geometry measure in the expanding toolbox of graph ricci curvatures. *Journal of Complex Networks*, 12, 3 (2024), cnae019. (Cited on page 53.)

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M., 2020a. Tudataset: A collection of benchmark datasets for learning with graphs. *ICML workshop 'Graph Representation Learning and Beyond'*, (2020). (Cited on pages 42, 46, 49, 50, and 62.)

Morris, C.; Rattan, G.; and Mutzel, P., 2020b. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. *Advances in Neural Information Processing Systems*, (2020). (Cited on pages 7, 21, and 30.)

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M., 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*. (Cited on pages 7 and 30.)

Mueller, S. T. and Tan, Y.-Y. S., 2018. Cognitive perspectives on opinion dynamics: The role of knowledge in consensus formation, opinion divergence, and group polarization. *Journal of Computational Social Science*, 1, 1 (2018), 15–48. (Cited on page 26.)

Namata, G.; London, B.; Getoor, L.; Huang, B.; and Edu, U., 2012. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, vol. 8, 1. (Cited on page 62.)

Nguyen, K.; Hieu, N. M.; Nguyen, V. D.; Ho, N.; Osher, S.; and Nguyen, T. M., 2023. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International Conference on Machine Learning*, 25956–25979. PMLR. (Cited on pages 8, 24, 25, and 52.)

Ni, C.-C.; Lin, Y.-Y.; Gao, J.; Gu, X. D.; and Saucan, E., 2015. Ricci curvature of the internet topology. In *2015 IEEE conference on computer communications (INFOCOM)*, 2758–2766. IEEE. (Cited on pages 24 and 52.)

Nikolentzos, G.; Dasoulas, G.; and Vazirgiannis, M., 2020. k-hop graph neural networks. *Neural Networks*, 130 (2020), 195–205. (Cited on pages 7, 21, and 30.)

Ollivier, Y., 2007. Ricci curvature of metric spaces. *Comptes Rendus Mathematique*, 345, 11 (2007), 643–646. (Cited on pages 8, 24, and 52.)

Ollivier, Y., 2009. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256, 3 (2009), 810–864. (Cited on pages 8 and 24.)

Panagopoulos, G.; Malliaros, F. D.; and Vazirgianis, M., 2020a. Influence maximization using influence and susceptibility embeddings. In *International AAAI Conference on Web and Social Media*, vol. 14. (Cited on page 116.)

Panagopoulos, G.; Malliaros, F. D.; and Vazirgiannis, M., 2020b. Multi-task learning for influence estimation and maximization. *IEEE Transactions on Knowledge and Data Engineering*, 34, 9 (2020). (Cited on page 127.)

Panagopoulos, G.; Tziortziotis, N.; Vazirgiannis, M.; and Malliaros, F., 2023. Maximizing influence with graph neural networks. In *International Conference on Advances in Social Networks Analysis and Mining*. (Cited on pages 11, 28, 29, 104, and 115.)

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B., 2020a. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*. (Cited on page 63.)

Pei, H.; Wei, B.; Chang, K. C. C.; Lei, Y.; and Yang, B., 2020b. Geomgcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations*. (Cited on pages 22, 43, 71, 84, and 103.)

Perozzi, B.; Al-Rfou, R.; and Skiena, S., 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. (Cited on page 3.)

Platonov, O.; Kuznedelev, D.; Diskin, M.; Babenko, A.; and Prokhorenkova, L., 2023. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*. (Cited on pages xiv, 71, 85, 86, and 103.)

Pouryahya, M.; Elkin, R.; Sandhu, R.; Tannenbaum, S.; Georgiou, T.; and Tannenbaum, A., 2016. Bakry-émery ricci curvature on weighted graphs with applications to biological networks. In *Int. Symp. on Math. Theory of Net. and Sys*, vol. 22, 52. (Cited on page 53.)

Protter, M. H. and Weinberger, H. F., 2012. *Maximum principles in differential equations*. Springer Science & Business Media. (Cited on page 56.)

Puri, A.; Akansha, S.; et al., 2025. Message passing graph neural networks: A study. *Global & Stochastic Analysis*, 12, 1 (2025). (Cited on page 4.)

Qureshi, S. et al., 2023. Limits of depth: Over-smoothing and over-squashing in gnns. *Big Data Mining and Analytics*, 7, 1 (2023), 205–216. (Cited on pages 8 and 23.)

Rainer, H. and Krause, U., 2002. Opinion dynamics and bounded confidence: models, analysis and simulation. (2002). (Cited on pages 10, 27, and 94.)

Riaz, F. and Ali, K. M., 2011. Applications of graph theory in computer science. In *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, 142–145. IEEE. (Cited on page 1.)

Robbins, H. and Monro, S., 1951. A stochastic approximation method. *The annals of mathematical statistics*, (1951), 400–407. (Cited on page 98.)

Roberts, F., 2012. *Applications of combinatorics and graph theory to the biological and social sciences*, vol. 17. Springer Science & Business Media. (Cited on page 1.)

Rossi, E.; Charpentier, B.; Di Giovanni, F.; Frasca, F.; Günnemann, S.; and Bronstein, M. M., 2024. Edge directionality improves learning on heterophilic graphs. In *Learning on Graphs Conference*, 25–1. PMLR. (Cited on pages 63, 84, and 104.)

Rossi, R. and Ahmed, N., 2015. The network data repository with interactive graph analytics and visualization. In *AAAI conference on artificial intelligence*, vol. 29. (Cited on pages 62, 103, and 127.)

Rozemberczki, B.; Allen, C.; and Sarkar, R., 2021. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9, 2 (2021), cnab014. (Cited on page 62.)

Saha, A.; Mendez, O.; Russell, C.; and Bowden, R., 2023. Learning adaptive neighborhoods for graph neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 22541–22550. (Cited on page 28.)

Samal, A.; Sreejith, R.; Gu, J.; Liu, S.; Saucan, E.; and Jost, J., 2018. Comparative analysis of two discretizations of ricci curvature for complex networks. *Scientific reports*, 8, 1 (2018), 8650. (Cited on pages 24 and 52.)

Sastry, S., 2013. *Nonlinear systems: analysis, stability, and control*, vol. 10. Springer Science & Business Media. (Cited on page 121.)

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G., 2008. The graph neural network model. *IEEE transactions on neural networks*, 20, 1 (2008), 61–80. (Cited on page 16.)

Scholkemper, M.; Wu, X.; Jadbabaie, A.; and Schaub, M. T., 2025. Residual connections and normalization can provably prevent oversmoothing in gnns. In *The Thirteenth International Conference on Learning Representations*. (Cited on pages 4 and 23.)

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T., 2008. Collective classification in network data. *AI magazine*, 29, 3 (2008), 93–93. (Cited on pages 42 and 62.)

Shayestehfard, K., 2023. *Permutation Invariant Graph Learning*. Ph.D. thesis, Northeastern University. (Cited on page 4.)

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S., 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, (2018). (Cited on page 84.)

Shu, J.; Xi, B.; Li, Y.; Wu, F.; Kamhoua, C.; and Ma, J., 2022. Understanding dropout for graph neural networks. In *companion proceedings of the web conference 2022*, 1128–1138. (Cited on page 23.)

Singh, A. P. and Gordon, G. J., 2008. A unified view of matrix factorization models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 358–373. Springer. (Cited on page 3.)

Smith, K. M., 2019. On neighbourhood degree sequences of complex networks. *Scientific reports*, 9, 1 (2019), 8340. (Cited on page 41.)

Song, Y.; Zhou, C.; Wang, X.; and Lin, Z., 2023. Ordered gnn: Ordering message passing to deal with heterophily and over-smoothing. In *The Eleventh International Conference on Learning Representations*. (Cited on page 22.)

Southern, J.; Wayland, J.; Bronstein, M. M.; and Rieck, B., 2023. On the expressive power of ollivier-ricci curvature on graphs. (2023). (Cited on page 68.)

Soydaner, D., 2022. Attention mechanism in neural networks: where it comes and where it goes. *Neural Computing and Applications*, 34, 16 (2022), 13371–13385. (Cited on page 3.)

Sperduti, A. and Starita, A., 1997. Supervised neural networks for the classification of structures. *IEEE transactions on neural networks*, 8, 3 (1997), 714–735. (Cited on pages 3 and 16.)

Sreejith, R.; Mohanraj, K.; Jost, J.; Saucan, E.; and Samal, A., 2016. Forman curvature for complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2016, 6 (2016), 063206. (Cited on page 25.)

Stanković, L.; Mandic, D.; Daković, M.; Brajović, M.; Scalzo, B.; Li, S.; Constantinides, A. G.; et al., 2020. Data analytics on graphs part iii: Machine learning on graphs, from graph topology to applications. *Foundations and Trends® in Machine Learning*, 13, 4 (2020), 332–530. (Cited on page 1.)

Sun, C.; Li, C.; Lin, X.; Zheng, T.; Meng, F.; Rui, X.; and Wang, Z., 2023a. Attention-based graph neural networks: a survey. *Artificial intelligence review*, 56, Suppl 2 (2023), 2263–2310. (Cited on pages 3 and 4.)

Sun, H.; Li, X.; Wu, Z.; Su, D.; Li, R.-H.; and Wang, G., 2024. Breaking the entanglement of homophily and heterophily in semi-supervised node classification. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2379–2392. IEEE. (Cited on pages xiii, 9, 22, and 64.)

Sun, L.; Huang, Z.; Wu, H.; Ye, J.; Peng, H.; Yu, Z.; and Philip, S. Y., 2023b. Deepricci: Self-supervised graph structure-feature co-refinement for alleviating over-squashing. In *2023 IEEE International Conference on Data Mining (ICDM)*, 558–567. IEEE. (Cited on pages 24 and 52.)

Sun, R.; Dai, H.; and Yu, A. W., 2022. Does gnn pretraining help molecular representation? *Advances in Neural Information Processing Systems*, 35 (2022), 12096–12109. (Cited on page 20.)

Suresh, S.; Budde, V.; Neville, J.; Li, P.; and Ma, J., 2021. Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 1541–1551. (Cited on pages xiv, 85, and 86.)

Tang, J.; Sun, J.; Wang, C.; and Yang, Z., 2009. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 807–816. (Cited on page 62.)

Tang, J.; Tang, X.; Xiao, X.; and Yuan, J., 2018. Online processing algorithms for influence maximization. In *2018 international conference on management of data*. (Cited on page 127.)

Tang, Y.; Shi, Y.; and Xiao, X., 2015. Influence maximization in near-linear time: A martingale approach. In *2015 ACM SIGMOD international conference on management of data*. (Cited on pages 115 and 127.)

Tennison, B. R., 1975. *Sheaf theory*, vol. 21. Cambridge University Press. (Cited on pages 12 and 116.)

Thorpe, M.; Nguyen, T.; Xia, H.; Strohmer, T.; Bertozzi, A.; Osher, S.; and Wang, B., 2022a. Grand++: Graph neural diffusion with a source term. *ICLR*, (2022). (Cited on pages 5 and 10.)

Thorpe, M.; Nguyen, T. M.; Xia, H.; Strohmer, T.; Bertozzi, A.; Osher, S.; and Wang, B., 2022b. Grand++: Graph neural diffusion with a source term. In *International Conference on Learning Representations*. (Cited on pages 18 and 104.)

Tolles, J. and Luong, T., 2020. Modeling epidemics with compartmental models. *Jama*, 323, 24 (2020), 2515–2516. (Cited on page 139.)

Topping, J.; Di Giovanni, F.; Chamberlain, B. P.; Dong, X.; and Bronstein, M. M., 2022. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*. (Cited on pages 8, 25, 52, and 68.)

Traag, V. A.; Waltman, L.; and Van Eck, N. J., 2019. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9, 1 (2019). (Cited on page 123.)

Turing, A. M., 1990. The chemical basis of morphogenesis. *Bulletin of mathematical biology*, 52 (1990). (Cited on page 119.)

Vabalas, A.; Gowen, E.; Poliakoff, E.; and Casson, A. J., 2019. Machine learning algorithm validation with a limited sample size. *PloS one*, 14, 11 (2019), e0224365. (Cited on page 127.)

Van der Maaten, L. and Hinton, G., 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9, 11 (2008). (Cited on page 69.)

Vargas-Pérez, V. A.; Giráldez-Cru, J.; Mesejo, P.; and Cordón, O., 2024. Unveiling agents' confidence in opinion dynamics models via graph neural networks. *IEEE Transactions on Computational Social Systems*, (2024). (Cited on page 27.)

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y., 2018. Graph attention networks. In *International Conference on Learning Representations*. (Cited on pages 4, 17, 28, 43, 58, 63, 81, 84, 93, 104, and 127.)

Wang, C.; Liu, Y.; Gao, X.; and Chen, G., 2021a. A reinforcement learning model for influence maximization in social networks. In *International Conference on Database Systems for Advanced Applications*. Springer. (Cited on page 28.)

Wang, J.; Guo, Y.; Yang, L.; and Wang, Y., 2024a. Understanding heterophily for graph neural networks. In *Forty-first International Conference on Machine Learning*. (Cited on pages 9, 23, and 71.)

Wang, J.; Guo, Y.; Yang, L.; and Wang, Y., 2024b. Understanding heterophily for graph neural networks. In *International Conference on Machine Learning*, 50489–50529. PMLR. (Cited on page 73.)

Wang, K.; Yang, Y.; Saha, I.; and Allen-Blanchette, C., 2025. Resolving oversmoothing with opinion dissensus. *arXiv preprint arXiv:2501.19089*, (2025). (Cited on page 27.)

Wang, R.; Mou, S.; Wang, X.; Xiao, W.; Ju, Q.; Shi, C.; and Xie, X., 2021b. Graph structure estimation neural networks. In *Proceedings of the web conference 2021*, 342–353. (Cited on page 22.)

Wang, X. and Xu, Y., 2019. An improved index for clustering validation based on silhouette index and calinski-harabasz index. In *IOP Conference Series: Materials Science and Engineering*, vol. 569, 052024. IOP Publishing. (Cited on page 90.)

Wang, X. and Zhang, M., 2022a. Glass: Gnn with labeling tricks for subgraph representation learning. In *International Conference on Learning Representations*. (Cited on pages 7 and 30.)

Wang, X. and Zhang, M., 2022b. How powerful are spectral graph neural networks. In *International conference on machine learning*, 23341–23362. PMLR. (Cited on pages 4 and 22.)

Wang, Y.; Cong, G.; Song, G.; and Xie, K., 2010. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. (Cited on page 115.)

Wang, Y.; Yi, K.; Liu, X.; Wang, Y. G.; and Jin, S., 2022a. Acmp: Allen-cahn message passing with attractive and repulsive forces for graph neural networks. In *The Eleventh International Conference on Learning Representations*. (Cited on pages 18, 104, and 120.)

Wang, Z.; Cao, Q.; Shen, H.; Bingbing, X.; Zhang, M.; and Cheng, X., 2022b. Towards efficient and expressive gnns for graph classification via subgraph-aware weisfeiler-lehman. In *The First Learning on Graphs Conference*. (Cited on pages 7, 21, and 30.)

Weber, F. and Zacher, R., 2021. The entropy method under curvature-dimension conditions in the spirit of bakry-émery in the discrete setting of markov chains. *Journal of Functional Analysis*, 281, 5 (2021), 109061. (Cited on page 53.)

Wei, G. and Wylie, W., 2007. Comparison geometry for the smooth metric measure spaces. In *Proceedings of the 4th International Congress of Chinese Mathematicians*, vol. 2, 191–202. (Cited on page 56.)

Wei, G. and Wylie, W., 2009. Comparison geometry for the bakry-emery ricci tensor. *Journal of differential geometry*, 83, 2 (2009), 337–405. (Cited on page 53.)

Weisfeiler, B. and Leman, A., 1968. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2, 9 (1968), 12–16. (Cited on pages 6, 19, and 30.)

Wijesinghe, A. and Wang, Q., 2022. A new perspective on how graph neural networks go beyond weisfeiler-lehman? In *International Conference on Learning Representations*. (Cited on pages xiii, xiv, 7, 20, 30, 42, 44, 63, and 66.)

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K., 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, 6861–6871. PMLR. (Cited on pages 63 and 74.)

Wu, Z.; Chen, J.; Al-Sabri, R.; Oloulade, B. M.; and Gao, J., 2024. Depth-adaptive graph neural architecture search for graph classification. *Knowledge-Based Systems*, 301 (2024), 112321. (Cited on page 28.)

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S., 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32, 1 (2020), 4–24. (Cited on pages 1 and 3.)

Xia, W. et al., 2021. Deepis: Susceptibility estimation on social networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 839–847. (Cited on pages 5, 11, 23, 28, 63, 65, 103, 104, and 116.)

Xu, J.; Zhang, A.; Bian, Q.; Dwivedi, V. P.; and Ke, Y., 2024. Union subgraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 16173–16183. (Cited on pages 7 and 30.)

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S., 2018a. How powerful are graph neural networks? In *International Conference on Learning Representations*. (Cited on page 63.)

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S., 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*. (Cited on pages xiii, 17, 20, 30, 36, 37, 40, 42, 43, 44, and 58.)

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S., 2018b. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, 5453–5462. PMLR. (Cited on page 27.)

Xu, W.; Zhu, L.; Guan, J.; Zhang, Z.; and Zhang, Z., 2022. Effects of stubbornness on opinion dynamics. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2321–2330. (Cited on page 26.)

Yang, F.; Zhang, H.; Tao, S.; and Hao, S., 2022a. Graph representation learning via simple jumping knowledge networks. *Applied Intelligence*, 52, 10 (2022), 11324–11342. (Cited on page 23.)

Yang, K.; Tu, J.; and Chen, T., 2019. Homoscedasticity: An overlooked critical assumption for linear regression. *General psychiatry*, 32, 5 (2019), e100148. (Cited on page 73.)

Yang, L.; Peng, W.; Zhou, W.; Niu, B.; Gu, J.; Wang, C.; Guo, Y.; He, D.; and Cao, X., 2022b. Difference residual graph neural networks. In *Proceedings of the 30th ACM international conference on multimedia*, 3356–3364. (Cited on page 23.)

Yang, Z.; Cohen, W.; and Salakhudinov, R., 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 40–48. PMLR. (Cited on page 71.)

Ye, Z.; Liu, K. S.; Ma, T.; Gao, J.; and Chen, C., 2019. Curvature graph network. In *International conference on learning representations*. (Cited on pages 8, 25, and 52.)

Yin, S.; Yang, Y.; Zhang, J.; and Zhou, T., 2022. Adaptive graph convolutional networks based on decouple and residuals to relieve over-smoothing. In *2022 International Joint Conference on Neural Networks (IJCNN)*, 01–08. IEEE. (Cited on page 23.)

You, J.; Gomes-Selman, J. M.; Ying, R.; and Leskovec, J., 2021. Identity-aware graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*. (Cited on pages 21, 30, and 63.)

Younesian, T.; Thanapalasingam, T.; van Krieken, E.; Daza, D.; and Bloem, P., 2023. Grapes: Learning to sample graphs for scalable graph neural networks. In *NeurIPS Workshop: New Frontiers in Graph Learning*. (Cited on page 27.)

Yu, Y.; Si, X.; Hu, C.; and Zhang, J., 2019. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31, 7 (2019), 1235–1270. (Cited on page 2.)

Yu, Z.; Feng, B.; He, D.; Wang, Z.; Huang, Y.; and Feng, Z., 2024. Lg-gnn: local-global adaptive graph neural network for modeling both homophily and heterophily. In *Proceedings of the thirty-third international joint conference on artificial intelligence*, 2515–2523. (Cited on page 22.)

Zeng, D.; Liu, W.; Chen, W.; Zhou, L.; Zhang, M.; and Qu, H., 2023. Substructure aware graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 11129–11137. (Cited on page 21.)

Zhang, B.; Fan, C.; Liu, S.; Huang, K.; Zhao, X.; Huang, J.; and Liu, Z., 2024a. The expressive power of graph neural networks: A survey. *IEEE Transactions on Knowledge and Data Engineering*, (2024). (Cited on page 63.)

Zhang, B.; Gai, J.; Du, Y.; Ye, Q.; He, D.; and Wang, L., 2024b. Beyond weisfeiler-lehman: A quantitative framework for gnn expressiveness. In *The Twelfth International Conference on Learning Representations*. (Cited on pages 21 and 30.)

Zhang, C.; Bu, Y.; Ding, Y.; and Xu, J., 2018. Understanding scientific collaboration: Homophily, transitivity, and preferential attachment. *Journal of the Association for Information Science and Technology*, 69, 1 (2018), 72–86. (Cited on page 22.)

Zhang, L.; Yan, X.; He, J.; Li, R.; and Chu, W., 2023. Drgcn: Dynamic evolving initial residual for deep graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, 11254–11261. (Cited on page 27.)

Zhang, M. and Li, P., 2021. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34 (2021), 15734–15747. (Cited on pages 7 and 30.)

Zhang, S.; Tong, H.; Xu, J.; and Maciejewski, R., 2019. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6, 1 (2019). (Cited on page 115.)

Zhang, Z.; Wang, X.; and Zhu, W., 2021. Automated machine learning on graphs: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*. International Joint Conferences on Artificial Intelligence Organization. (Cited on page 1.)

Zhao, J.; Dong, Y.; Ding, M.; Kharlamov, E.; and Tang, J., 2021. Adaptive diffusion in graph neural networks. *Advances in neural information processing systems*, 34 (2021), 23321–23333. (Cited on pages 5, 17, and 93.)

Zhao, L. and Akoglu, L., 2020. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*. (Cited on page 23.)

Zhao, L.; Jin, W.; Akoglu, L.; and Shah, N., 2022a. From stars to subgraphs: Uplifting any gnn with local structure awareness. *International Conference on Learning Representations*, (2022). (Cited on pages xiii, 20, 30, 42, and 44.)

Zhao, L.; Shah, N.; and Akoglu, L., 2022b. A practical, progressively-expressive gnn. *Advances in Neural Information Processing Systems*, (2022). (Cited on page 21.)

Zhao, W.; Guo, T.; Yu, X.; and Han, C., 2023. A learnable sampling method for scalable graph neural networks. *Neural Networks*, 162 (2023), 412–424. (Cited on page 27.)

Zheng, X.; Wang, Y.; Liu, Y.; Li, M.; Zhang, M.; Jin, D.; Yu, P. S.; and Pan, S., 2022. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, (2022). (Cited on pages 9 and 71.)

Zheng, X.; Zhang, M.; Chen, C.; Zhang, Q.; Zhou, C.; and Pan, S., 2023a. Autoheg: Automated graph neural network on heterophilic graphs. In *Proceedings of the ACM Web Conference 2023*, 611–620. (Cited on pages 9 and 71.)

Zheng, Y.; Zhang, H.; Lee, V.; Zheng, Y.; Wang, X.; and Pan, S., 2023b. Finding the missing-half: Graph complementary learning for homophily-prone and heterophily-prone graphs. In *International Conference on Machine Learning*, 42492–42505. PMLR. (Cited on pages xiv, 22, 85, and 86.)

Zhong, G.; Wang, L.-N.; Ling, X.; and Dong, J., 2016. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2, 4 (2016), 265–278. (Cited on page 2.)

Zhou, B.; Lv, O.; Wang, J.; Xiao, X.; and Zhao, W., 2024. Odnet: Opinion dynamics-inspired neural message passing for graphs and hypergraphs. *Transactions on Machine Learning Research*, (2024). (Cited on pages 27 and 104.)

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M., 2020a. Graph neural networks: A review of methods and applications. *AI open*, 1 (2020), 57–81. (Cited on page 1.)

Zhou, J.; Feng, J.; Wang, X.; and Zhang, M., 2023. Distance-restricted folklore weisfeiler-leman gnns with provable cycle counting power. *Advances in Neural Information Processing Systems*, 36 (2023), 14293–14337. (Cited on page 37.)

Zhou, K.; Huang, X.; Li, Y.; Zha, D.; Chen, R.; and Hu, X., 2020b. Towards deeper graph neural networks with differentiable group normalization. *Advances in neural information processing systems*, 33 (2020), 4917–4928. (Cited on page 23.)

Zhu, J.; Rossi, R. A.; Rao, A.; Mai, T.; Lipka, N.; Ahmed, N. K.; and Koutra, D., 2021. Graph neural networks with heterophily. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 11168–11176. (Cited on pages 9, 22, and 71.)

Zhu, J.; Yan, Y.; Heimann, M.; Zhao, L.; Akoglu, L.; and Koutra, D., 2023a. Heterophily and graph neural networks: Past, present and future. *IEEE Data Engineering Bulletin*, (2023). (Cited on pages 9, 22, and 71.)

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D., 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33 (2020), 7793–7804. (Cited on pages xiii, 9, 22, 64, and 71.)

Zhu, X.; Fu, J.; and Chen, C., 2023b. Matrix completion of adaptive jumping graph neural networks for recommendation systems. *IEEE Access*, 11 (2023), 88433–88450. (Cited on page 23.)