UCLA
UCLA Electronic Theses and Dissertations

Title

Explainable Artificial Intelligence for Graph Data

Permalink

https://escholarship.org/uc/item/6bf1g6dc

Author

Zhang, Shichang

Publication Date

2024

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
Los Angeles

Explainable Artificial Intelligence for Graph Data

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Shichang Zhang

2024

© Copyright by
Shichang Zhang
2024

ABSTRACT OF THE DISSERTATION

Explainable Artificial Intelligence for Graph Data

by

Shichang Zhang
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Yizhou Sun, Chair

The development of artificial intelligence (AI) has significantly impacted our daily lives and even driven new scientific discoveries. However, the modern AI models based on deep learning remain opaque "black boxes" and raise a critical "why question" – why are these AI models capable of achieving such remarkable outcomes? Answering this question leads to research on Explainable AI (XAI), which offers numerous benefits, such as enhancing model performance, establishing user trust, and extracting deeper insights from data. While XAI has been explored for some data modalities like images and text, relevant research on graph data, a more complex data modality that represents both entities and their relationships, is underdeveloped. Given the ubiquity of graph data and their prevalent applications across main domains including science, business, and healthcare, XAI for graph data becomes a critical research direction.

This thesis aims to address the gap in XAI for graph data from three complementary and equally important perspectives: model, user, and data. Accordingly, my research advances XAI for graph data by developing: (1) Model-oriented explanation techniques that illuminate the mechanism and enhance the performance of state-of-the-art AI models on graph data. (2)

User-oriented explanation approaches that offer intuitive visualizations and natural language explanations to establish user trust in graph AI models for real-world applications. (3) Data-oriented explanation methods that identify key patterns and extract insights from graph data, potentially leading to new scientific discoveries. By integrating these three perspectives, this thesis enhances the transparency, trustworthiness, and insightfulness of AI for graph data across domains and applications.

The dissertation of Shichang Zhang is approved.

Christos Faloutsos

Cho-Jui Hsieh

Wei Wang

Yizhou Sun, Committee Chair

University of California, Los Angeles

2024

TABLE OF CONTENTS

1 Introduction **1**

1.1 Background 1

1.2 Thesis Summary 2

I Explanations Elucidate Model Mechanisms **5**

2 Explain GNNs with Structure-aware Cooperative Games **6**

2.1 Introduction 6

2.2 Preliminaries 8

2.2.1 Graph Neural Networks 8

2.2.2 Cooperative Game Theory 9

2.3 GStarX: Graph Structure-aware Explanation 11

2.3.1 GNN Explanation via Feature Importance Scoring 11

2.3.2 Scoring Functions from Cooperative Games 12

2.3.3 The HN Value 14

2.3.4 GNN Message Passing and The HN Surplus Allocation 17

2.3.5 The GStarX Algorithm 18

2.3.6 GStarX for Node Classification 20

2.4 Evaluation 20

2.4.1 Experiment Settings 20

2.4.2 Evaluation Results 24

2.4.3 Ablation Studies and Analysis 28

2.5 Related Work 30
2.6 Discussion 32

II Explanations Establish User Trust 33

3 Path-based GNN Explanation for Link Prediction 34
3.1 Introduction 34
3.2 Related Work 37
3.3 Preliminaries 39
3.4 PaGE-Link: Path-based GNN Explanation for Link Prediction 41
3.4.1 Link-Prediction Explanation 41
3.4.2 K-core Pruning 44
3.4.3 Heterogeneous Path-Enforcing Mask Learning 45
3.4.4 Mask Optimization and Path Generation 48
3.4.5 Complexity Analysis 49
3.5 Evaluation 50
3.5.1 Datasets 51
3.5.2 Experiment Settings 53
3.5.3 Algorithmic Evaluation 54
3.5.4 Human Evaluation 56
3.6 Discussion 57

III Explanations Extract Data Insights 58

4 Predicting and Interpreting Energy Barriers of Metallic Glasses with GNNs 59

4.1 Introduction 59

4.2 Related Work 63

4.2.1 GNN Explanation 64

4.3 Problem Setup and Preliminaries 65

4.3.1 EB Prediction with GNNs 65

4.3.2 Orthogonality and Invariance 65

4.3.3 GNNExplainer 67

4.4 SymGNN: Symmetrized GNNs 67

4.4.1 Theory of Symmetrization Over O(3) 68

4.4.2 Symmetrized GNN 69

4.4.3 Computation Time Analysis 71

4.4.4 Expressiveness Analysis 72

4.4.5 Explanations for Structure-EB Relationship 73

4.5 Prediction Evaluation 74

4.5.1 Dataset 74

4.5.2 Experiment Settings 76

4.5.3 Prediction Results 78

4.5.4 Computation Time Comparison 79

4.5.5 Abalations and Further Comparisons 79

4.6 Explanation Evaluation and Analysis 82

4.6.1 Explanation Visualization and MRO 82

4.6.2 Edge Importance Explanation and TDA 83

4.7 Discussion 84

5 Motif Mining via Clustering Representations of Graphs 86

5.1 Introduction 86

5.2 Related Work 88

5.3 Micro-Graph: Motif Mining via Clustering Representations of Graphs 90

5.3.1 Probabilistic Modeling 90

5.3.2 Motif Descriptor Inference 91

5.3.3 Visualize and Interpret Mined Motifs 92

5.3.4 MICRO-Graph Parameter Learning 93

5.4 Evaluation 96

5.4.1 Datasets 97

5.4.2 Model Configuration and Implementation 98

5.4.3 Qualitative Evaluation 99

5.4.4 Quantitative Evaluation 101

5.4.5 Ablation Study and Analysis 103

5.5 Discussion 104

6 Automated Molecular Concept Generation and Labeling 106

6.1 Introduction 106

6.2 Related Work 109

6.3 AutoMolCo: Automated Molecular Concept Generation and Labeling 111

6.3.1 Step 1: Concept Generation 111

6.3.2 Step 2: Concept Labeling 112

6.3.3 Step 3: CM Fitting and Concept Selection. 116
6.3.4 Iterative Concepts Refinement 117
6.4 Evaluation 117
6.4.1 Experiment Settings 117
6.4.2 AutoMolCo-induced CM Performance 119
6.4.3 RQ1: Can AutoMolCo Generate Meaningful Molecular Concepts? . . 119
6.4.4 RQ2: Can AutoMolCo Assign Molecules Reasonable Concept Labels Using Each Strategy? 120
6.4.5 RQ3: Can AutoMolCo Produced Concepts and Labels Be Utilized To Build an Effective CM? 122
6.4.6 RQ4: Does Iterative Refinement Boost The Performance of AutoMolCo-induced CM? 123
6.4.7 RQ5: Does The AutoMolCo-induced CM Facilitate Explainable Molecular Science? 123
6.4.8 Ablation Studies 126
6.5 Discussion 131

7 Conclusion 133

A Appendices 135

A.1 Chapter 2 Appendices 135
A.1.1 The Myerson Value, C-Shapley Value, and L-hop Graph Cutoff 135
A.1.2 More Explanation Visualizations 142
A.2 Chapter 3 Appendices 142
A.2.1 Proof of Proposition 3.4.1 142

A.2.2 Theorem 3.4.3: A Detailed Version 143
A.3 Chapter 4 Appendices 144
A.3.1 Proof of Lemma 4.4.1 144
A.3.2 Proof of Theorem 4.4.3 145
A.3.3 More Explanation Visualizations 146
A.4 Chapter 5 Appendices 146
A.4.1 Derivations of The Log-likelihood and The Posterior Probability . . . 146
A.4.2 Derivation of The Log-likelihood Lower Bound with Optimal Transport 148
A.5 Chapter 6 Appendices 150
A.5.1 Decision Tree Visualizations 150

References **153**

LIST OF FIGURES

1.1 Ubiquitous applications of AI on graph data. 2
1.2 Three complementary perspectives of XAI. 3

2.1 Explanations on graphs with structure-aware values 8
2.2 Explanations on sentences from GraphSST2 24
2.3 Explanations on a mutagenic molecule in MUTAG 25
2.4 H-Fidelity vs. sparsity for GStarX and baselines 26

3.1 Path-based explanations generated by PaGE-Link 35
3.2 Accuracy and run-time comparison between PaGE-Link and baselines. 37
3.3 Illustration of the PaGE-Link framework. 41
3.4 The proposed augmented graph AugCitation and the synthetic graph UserItemAttr. 50
3.5 Explanation visualization and comparison between PaGE-Link and baselines. . . 52
3.6 Top paths selected by PaGE-Link 53

4.1 EBs, mobility, and MG physical properties. 60
4.2 Example graphs demonstrating model expressiveness. 61
4.3 Illustration of the SymGNN framework. 70
4.4 Global and local explanation visualizations for SymGNN on MG node 1501. . . 82
4.5 Analysis of explanations for SymGNN prediction. 83

5.1 Difference between the traditional motif definition and our motif definition. . . . 88
5.2 Probabilistic graphical model of the graph generating process in MICRO-Graph. 90
5.3 The MICRO-Graph framework. 91

5.4 Frequent motifs mined from the HIV chemical compound dataset. 100
5.5 Visualization of the top 3 most similar subgraphs corresponding to motifs. . . . 101
5.6 Frequent motifs mined from the DD protein dataset. 102
5.7 MICRO-Graph ablation on K 105

6.1 The prediction process of molecule properties is greatly illuminated with AutoMolCo. 107
6.2 The AutoMolCo framework. 111
6.3 Prompts for concept generation and labeling on FreeSolv. 114
6.4 Prompts for generating concept labeling functions in Python code on FreeSolv. . 114
6.5 Prompts for calling the external tool RDKit to label concepts on FreeSolv. . . . 115
6.6 RQ1: Concepts selected by AutoMolCo in three refinement iterations on FreeSolv. 121
6.7 RQ4: Iterative refinement improves CM performance for classification tasks. . . 123
6.8 RQ5: Coefficients of the logistic regression model on BBBP with concepts refined by AutoMolCo after three iterations. 125
6.9 RQ5: Coefficients of the linear regression model and the decision tree on FreeSolv. 126
6.10 Intervention on logP of diphenylamine for predicting solubility with MLP 127
6.11 Correlation between the ground truth labels and concept labels generated using molecule names or SMILES strings. 129

A.1 Marginal contribution coefficients comparison 139
A.2 More explanations on mutagenic molecules from the MUTAG 140
A.3 More explanations on sentences from GraphSST2. 141
A.4 Local and global explanation visualizations for SymGNN on more MG nodes. . . 151
A.5 The decision tree for AutoMolCo-induced CM classification on BBBP. 152
A.6 The decision tree for AutoMolCo-induced CM classification on BACE. 152

LIST OF TABLES

2.1 GStarX experiment dataset statistics. 21

2.2 GStarX experiment GCN hyperparameters. 22

2.3 GStarX experiment GIN and GAT hyperparameters 22

2.4 The best H-Fidelity of different Sparsity for each dataset. 27

2.5 Ablation on GNN architectures for GStarX. 28

2.6 Comparison of average running time on 50 graphs in BBBPbetween GStarX and baselines. 28

2.7 The entropy-based sparsity scores of GStarX 30

3.1 Methods and desired explanation properties 39

3.2 PaGE-Link notations. 40

3.3 Time complexity of PaGE-Link and baseline methods. 49

3.4 Hyperparameters for constructing AugCitation and UserItemAttr 52

3.5 Performance comparison in ROC-AUC scores of learned masks between PaGE-Link and baselines. 54

3.6 Performance comparison in path hit rates between PaGE-Link and baselines. . . 55

4.1 Characteristic comparison of different methods. 62

4.2 Theoretical time complexity of SymGNN and baselines. 72

4.3 Testing scores comparison of SymGNN, the molecular dynamics (MD) method, and other ML methods. 78

4.4 Training time comparison for one epoch between SymGNN and baselines. 79

4.5 Inference time comparison on an MG with 3,000 atoms between SymGNN and baselines. 79
4.6 Performance comparison of the original and new dataset splits between SymGNN and SchNet. 80
4.7 SymGNN performance ablated on different number of orthogonal transformations. 81
4.8 SymGNN explanation and optimal cycles. 84

5.1 Performance comparison between MICRO-Graph and baselines on chemical compound datasets. 103
5.2 Performance comparison between MICRO-Graph and baselines on protein datasets. 103
5.3 MICRO-Graph ablation on $\mathcal{L}_{cut}$ 104

6.1 Performance comparison of the AutoMolCo-induced CM with baselines. 120
6.2 RQ2: Percentage of concepts with high correlations with the ground-truth. . . . 122
6.3 RQ3: AutoMolCo-induced CM performance with different labeling strategies and prediction models. 122
6.4 AutoMolCoablation on LLMs (GPT-3.5 vs. Claude-2). 127
6.5 AutoMolCoablation on input formats (SMILES strings vs. molecule names). . . 128
6.6 AutoMolCowith combined labeling strategies. 130
6.7 Performance comparison of the best AutoMolCo-induced CM vs. LLM ICL in accuracy on BBBP and BACE. 130
6.8 Performance comparison of the AutoMolCo-induced CM with different prediction models vs. LLM ICL in accuracy on BH and SM. 131

ACKNOWLEDGMENTS

This thesis represents five years of work with the support of many people along the way, without their support, the work would not have been possible. The first and foremost among them is my Ph.D. advisor, Yizhou Sun. I am profoundly grateful to Yizhou for giving me enough flexibility to explore whatever research topics I got interested in, encouraging me to ask questions that I did not know how to begin to answer, and providing hands-on guidance whenever I needed it. Her mentorship has been invaluable. She has been beyond the best advisor I could have imagined before starting my Ph.D. journey. I thank her for everything.

I also want to extend my gratitude to my committee members: Wei Wang, Cho-Jui Hsieh, and Christos Faloutsos, for their advice and mentorship. Their thoughtful feedback and unwavering support have been crucial to my research progress.

I would like to give special thanks to several people to whom I am deeply grateful. The first is John (Junghoo) Cho, who I consider my unofficial committee member. If John were still healthy and with us, I would have definitely invited him to join my committee. I can still recall many enjoyable discussions I had with him in my first year of Ph.D., where John showed his passion and meticulousness about research. John set a role model for me of what a true intellectual looks like. I hope in 10 to 20 years, I can become an intellectual like him.

The second is Ziniu Hu, who is my good friend and long-term collaborator. Ziniu taught me many practical research skills that helped me start my first research project. He has been continually sharing exciting research ideas with me to date. Our friendship is marked by countless nights stayed up late together in the ScAI lab until way past midnight, and numerous intense research discussions happened while studying, eating, and walking together. Back in the day, we disagreed about 50% of the time, and throughout the past five years, that percentage has been getting lower and lower.

The third is Chumeng Cheng, who I married during my Ph.D. and will spend the rest of my life with. Unlike those already mentioned, She has not helped me directly with my

academic research, but she has been supporting and loving me in all other aspects of life. Her optimism and liveliness have carried me through these years. I would not know how to live a fulfilling life without her.

This thesis is a joint effort of many collaborators, including Soji Adeshina, Haoyu Li, Yozen Liu, Neil Shah, Xiang Song, Arjun Subramonian, Fang Sun, Longwen Tang, Qianli Wu, Botao Xia, Jiani Zhang, Zimin Zhang, Da Zheng, and Yizhou, Christos, and Ziniu mentioned above. Thanks to their intellectual input and time commitment. Special thanks to Neil for his mentorship on multiple projects. I truly enjoyed our collaboration and have passed down the skills he taught me to my mentees. Another special thanks to Jiani for her mentorship and the joyful activities she organized, which provided a refreshing break from work.

Also, I owe many thanks to the wonderful ScAI lab members. Thanks to Yunsheng Bai, Xuelu Chen, and Junheng Hao for helping me to embark on my graduate studies. Thanks to Xiusi Chen, Kewei Cheng, Song Jiang, Ziniu Hu, Zijie Huang, Roshni Iyer, Xiao Luo, Mingyu Ma, Arjun Subramonian, Zongyue Qin, Yewen Wang, Derek Xu, and Zhiping Xiao for engaging in daily research and relaxing discussions. Thanks to Yuanzhou Chen, Jingru Gan, Weikai Li, Zongyu Lin, Fang Sun, Xiaoxuan Wang, Yijia Xiao, Zeyuan Xu, Chenchen Ye, and Yanqiao Zhu for bringing freshness and joy to my workdays. Thanks to Ling Ding for the spiritual fellowship; I hope she finds joy and peace in the love of God. Thank everyone for their friendship and for those nice meals and great adventures we had together. It has been an incredible journey being part of the ScAI lab. The ScAI lab has truly become a family. A family that I am so lucky to be a part of, and a family that is so difficult to say goodbye to.

Thanks to the J.P. Morgan Chase AI Fellowship and the Amazon Science Hub Fellowship for supporting my research. Thanks to Yanlong Ma for being a companion since middle school, who has attended the same six different schools with me and supported me for the past 17 years. Thanks to my parents for their faith and support in whatever I choose to do. Thanks to the brothers and sisters from the Alhambra Christian Fellowship and Peninsula Church in Christ for their care, support, and for bringing me into God's love. Glory to God.

VITA

Expected	Ph.D. in Computer Science, University of California, Los Angeles, CA
Apr. 2019	M.S. in Statistics, Stanford University, Stanford, CA
May 2017	B.A. in Statistics, University of California, Berkeley, CA Honors: Honors in Statistics, High Distinction

PUBLICATIONS

Haoyu Li*, **Shichang Zhang***, Longwen Tang, Yizhou Sun. "Predicting and Interpreting Energy Barriers of Metallic Glasses with Graph Neural Networks" (**ICML 2024, *equal contribution**)

Xiaoxuan Wang*, Ziniu Hu*, Pan Lu*, Yanqiao Zhu*, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, **Shichang Zhang**, Yizhou Sun, Wei Wang. "SciBench Evaluating College-LevelScientific Problem-Solving Abilities of Large Language Models" (**ICML 2024, *equal contribution**)

Yewen Wang, **Shichang Zhang**, Junghoo Cho, Yizhou Sun. "Laplacian Score Benefit Adaptive Filter Selection for Graph Neural Networks" (**SDM 2024**)

Zhichun Guo, William Shiao, **Shichang Zhang**, Yozen Liu, Nitesh Chawla, Neil Shah, Tong Zhao. "Linkless Link Prediction via Relational Distillation" (**ICML 2023**)

Shichang Zhang, Jiani Zhang, Xiang Song, Soji Adeshina, Da Zheng, Christos Faloutsos, Yizhou Sun. "PaGE-Link: Graph Neural Network Explanation for Heterogeneous Link Prediction" (**WWW**2023)

Shichang Zhang, Yozen Liu, Neil Shah, Yizhou Sun. "Explaining Graph Neural Networks with Structure-Aware Cooperative Games" (**NeurIPS 2022**)

Shichang Zhang, Yozen Liu, Yizhou Sun, Neil Shah. " Graph-less Neural Networks, Teach Old MLPs New Tricks via Distillation" (**ICLR 2022**)

Wei Jin, Lingxiao Zhao, **Shichang Zhang**, Yozen Liu, Jiliang Tang, Neil Shah. " Graph Condensation for Graph Neural Networks" (**ICLR 2022**)

Shichang Zhang*, Ziniu Hu*, Arjun Subramonian, Yizhou Sun. "Motif-driven Contrastive Learning of Graph Representations" (**TKDE, *equal contribution**)

CHAPTER 1

Introduction

1.1 Background

Artificial Intelligence (AI) has made remarkable advancements and achieved human-level or even superhuman performance in diverse domains, ranging from image recognition and chatbots to game-playing and new material structure predictions. These developments have significantly impacted our daily lives and even driven new scientific discoveries. However, the impressive AI achievements are largely based on the development of deep learning, which makes AI models remain opaque and difficult to interpret, often referred to as "black boxes". The lack of explainability in these black-box AI models has raised a critical "why question" that concerns researchers, practitioners, and policymakers – why are these AI models capable of achieving such remarkable outcomes? As AI models become prevalent and increasingly influence critical decision-making processes in various domains, understanding them becomes imperative.

Answering the why question leads to research on Explainable AI (XAI), which offers numerous benefits, such as enhancing model performance, establishing user trust, and extracting deeper insights from data. While the field of XAI has gained traction in recent years, with researchers exploring various techniques to explain AI models, the focus has predominantly been on data modalities such as images and text. However, the domain of graph data, which represents not only entities but also their relationships, has received little attention in the context of XAI. Given the ubiquity of graph data in many domains with

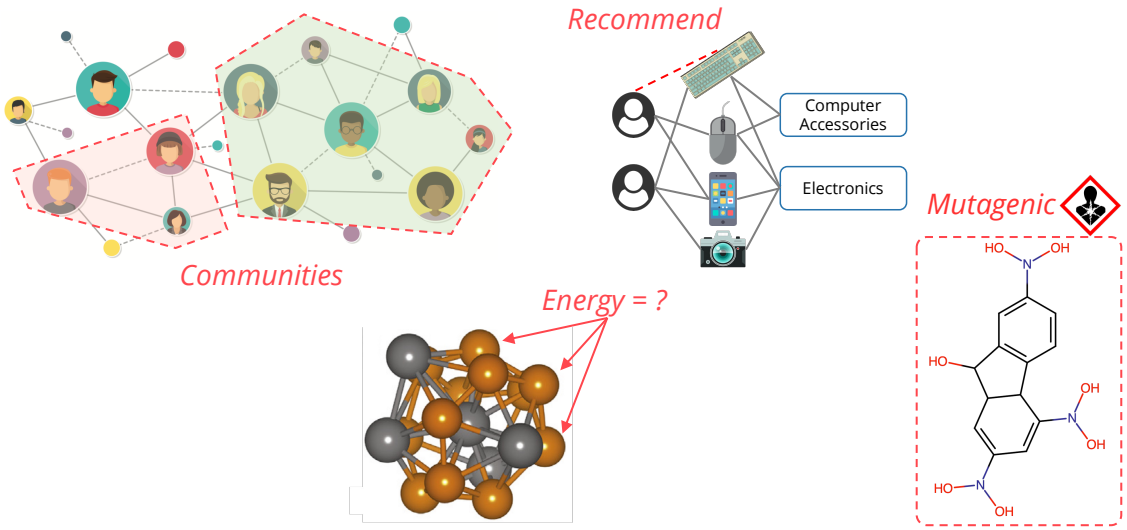


Figure 1.1: **Ubiquitous applications of AI on graph data.** Detect communities on social networks. Recommend products to customers. Predict the atom energy of materials. Classify properties of molecules.

numerous real-world applications, such as analysis of social networks, recommendations on e-commerce graphs, property prediction of molecule graphs, and many more (Figure 1.1), XAI for graph data emerges as a critical research direction.

1.2 Thesis Summary

This thesis aims to address the gap in XAI for graph data from three complementary and equally important perspectives: model, user, and data (Figure 1.2). These perspectives are integral to the entire AI pipeline and form the foundational pillars of the XAI research. Firstly, the model serves as the core for prediction, generation, or any other tasks for which it is designed. Unveiling its mechanism is a key challenge that XAI needs to address, where results can be used for debugging models and further enhancing model performance. Next, the user, as the stakeholder of the explanations, must find them comprehensible. A guiding principle for XAI researchers, akin to "Explain the moon landing to a 6-year-old", underscores that an explanation is considered successful only if it is clear enough to establish trust among users,

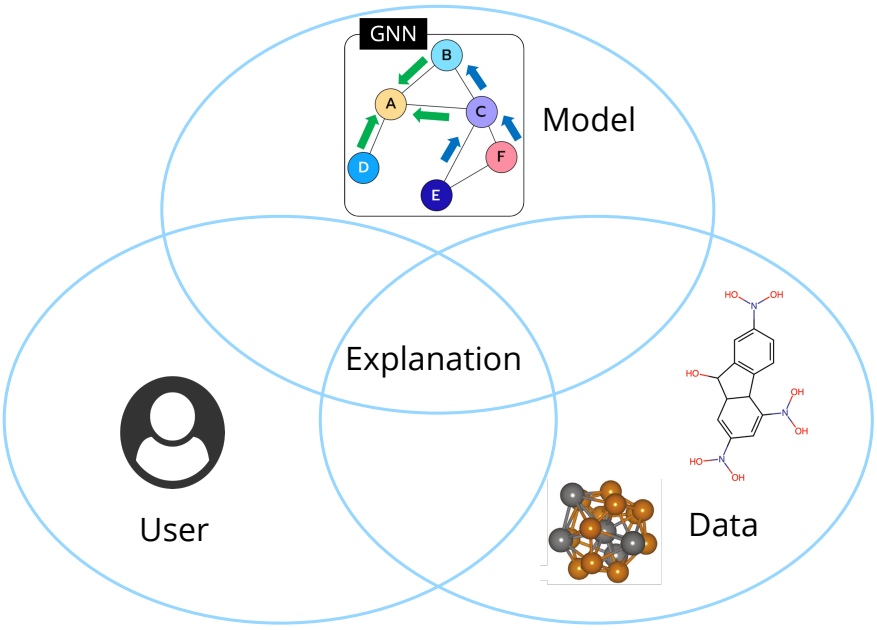


Figure 1.2: Three complementary perspectives of XAI.

including those without any technical background. Finally, the data underpins the model and provides the knowledge that the model learns. Upon revealing the model's underlying mechanisms, explanations are expected to delve deeper and extract insights from the data, especially insights that domain experts had not anticipated. This aspect is particularly crucial for scientific data that often appears as graphs, where such unexpected discoveries could lead to significant breakthroughs.

My research advances XAI for graph data by developing novel methods and frameworks regarding these three perspectives:

1. Model-oriented explanation techniques that illuminate the mechanisms and enhance the performance of AI models on graph data. Particularly, I focus on the state-of-the-art Graph Neural Network (GNN) models and propose a novel structure-aware approach based on cooperative game theory to explain the GNN prediction process, especially how GNN message passing works.

2. User-oriented explanation approaches that offer intuitive visualizations and natural language explanations to establish user trust. A specific application involves recommending items to users as link predictions on e-commerce graphs. My explanations take the form as intuitive paths connecting the source and target of the predicted link and can be clearly expressed in natural language for non-technical users.

3. Data-oriented explanation methods that identify key patterns and extract insights from graph data. For this perspective, I design both post hoc explanation methods as well as inherently explainable models tailored for scientific graph data. These methods have been applied to material science and molecular science problems, producing purely AI-based explanations without any human knowledge inputs besides data but match domain experts' knowledge.

By integrating these three complementary perspectives, this thesis contributes to XAI by enhancing the transparency, trustworthiness, and insightfulness of AI models for graph data across domains and applications. Moreover, it can facilitate effective model debugging and error analysis, ensure fairness in the decision-making processes, promote the responsible deployment of AI systems, and potentially lead to new scientific discoveries.

Part I

Explanations Elucidate Model Mechanisms

CHAPTER 2

Explain GNNs with Structure-aware Cooperative Games

2.1 Introduction

Explainability is crucial for complex machine learning (ML) models in sensitive applications, helping establish user trust and providing insights for potential model improvements. Many efforts focus on explaining models on images, text, and tabular data. In contrast, the explainability of models on graph data is yet underexplored. Since explainability can be especially critical for many graph tasks like drug discovery, and interest in deep graph models is growing rapidly, further investigation of graph explainability is warranted. In this work, we study graph ML explanation with GNNs as the target models, given their popularity and widespread use for graph machine learning tasks [YHC18, SLY21, WSZ20, TLS20, TLH22, ZJS21].

In ML explainability, important features are identified, and the Shapley value [Sha53] has been deemed as a "fair" scoring function for computing feature importance. Originally from cooperative game theory, many values, including the Shapley value, have been proposed for allocating a total payoff to players in a game. When used for scoring the feature importance of a data instance, the model prediction is treated as the total payoff and the features are considered as players. In particular, for an instance with n features $\{\boldsymbol{x}_1, \dots \boldsymbol{x}_n\}$, the Shapley value of its ith feature $\boldsymbol{x}_i$ is computed via aggregating $m(i, S)$, which are the marginal contributions of $\boldsymbol{x}_i$ to sets of other features $\boldsymbol{x}_S \subseteq \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \setminus \{\boldsymbol{x}_i\}$. Each $\boldsymbol{x}_S$ is called a *coalition*. Each $m(i, S)$ is computed as the difference between model outputs for $\boldsymbol{x}_S \cup \{\boldsymbol{x}_i\}$ and $\boldsymbol{x}_S$, e.g., difference of probability belonging to a target class for these two set of features,

and it is meant to capture the interaction between $\boldsymbol{x}_i$ and $\boldsymbol{x}_S$. The Shapley value is widely used for explaining ML models on images, text, and tabular data, when the features are pixels, words, and attributes [LC01, LL17].

The Shapley value has recently been extended to explain GNNs on graphs through feature importance scoring as above, where features are nodes [DM21] or supernodes [YYW21]. We argue that the Shapley value is a non-ideal choice for (super)node importance scoring because its contribution aggregation is **non-structure-aware**. The Shapley value aggregation assumes no structural relationship between $\boldsymbol{x}_i$ and $\boldsymbol{x}_S$ even though they are both parts of the input graph (a review of the Shapley value is in Section 2.2.2). Since the graph structure generally contains critical information and is crucial to the success of GNNs, we consider properly leveraging the structure with a better **structure-aware** scoring function.

We propose Graph Structure-aware eXplanation (GStarX), where we construct a structure-aware node importance scoring function based on the Hamiache-Navarro (HN) value [HN20] from cooperative game theory. Recall that GNNs make predictions via message passing, during which node representations are learned by aggregating messages from neighbors. Message passing aggregates both feature and structure information, resulting in powerful structure-aware models [CCV20]. The HN value shares a similar idea to message passing by allocating the payoff surplus generated from the cooperation between neighboring players (nodes). When used as a scoring function to explain node importance, the HN value captures both features and structural interactions between nodes (details in Section 2.3). Figure 2.1**(a)** shows an example comparing the Shapley value and the HN value. In this example, their difference boils down to different aggregation weights of marginal contributions, where the former is uniform and the latter is structure-aware (details in Section 2.3.2). In summary, our contributions are:

- Identify the non-structure-aware limitation of the Shapley value for GNN explanation.
- Introduce the structure-aware HN value from cooperative game theory to the graph machine learning community and connect it to the GNN message passing and GNN explanation.

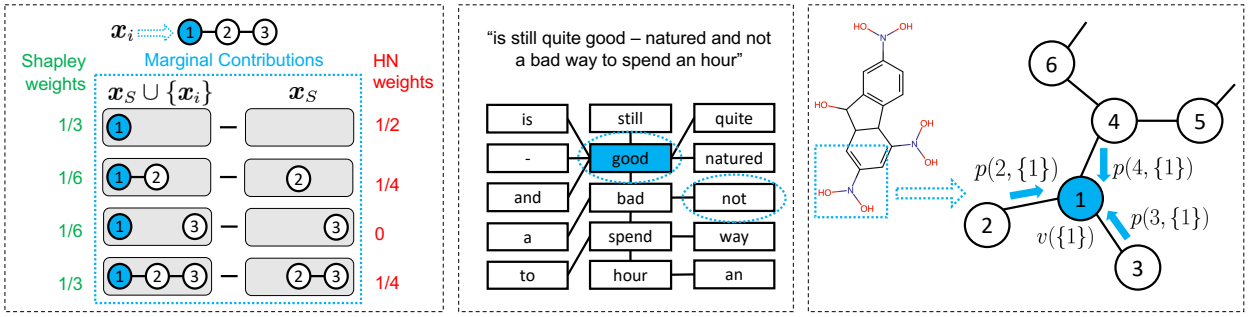


Figure 2.1: **Explanations on graphs with structure-aware values.** Explanations on graphs with structure-aware values (like HN) offer advantages over non-structure-aware values (like Shapley). **(a) Synthetic graph (left):** The Shapley value assigns weights to $m(i, S)$ only based on size of $\boldsymbol{x}_S$, while the HN value assigns weights considering structures and in particular gives zero weight to the disconnected $\boldsymbol{x}_S$. **(b) Text graph (middle):** For a sentence classified as positive, the {"not", "good"} coalition shouldn't be considered when they are not connected by "bad". **(c) Chemical graph (right):** For a chemical graph with mutagenic functional group -NO2, the importance of the atom N (node 1) is better recognized if decided locally within the functional group.

- Propose a new HN-value-based GNN explanation method GStarX, and demonstrate the superiority of GStarX over strong baselines for explaining GNNs on chemical and text graphs.

2.2 Preliminaries

2.2.1 Graph Neural Networks

Consider a graph $\mathcal{G}$ with (feature-enriched) nodes $\mathcal{V}$ and edges E. We denote $\mathcal{G}$ as $\mathcal{G} = (\mathcal{V}, \boldsymbol{X}, \boldsymbol{A})$, where $\boldsymbol{X} \in \mathcal{R}^{n \times d}$ denotes d-dimensional features of n nodes in $\mathcal{V}$, and $\boldsymbol{A} \in \{0,1\}^{n \times n}$ denotes the adjacency matrix specifying edges in E. GNNs make predictions on $\mathcal{G}$ by learning representations via the *message-passing* mechanism. During message passing, the representation of each node $u \in \mathcal{V}$ is updated by aggregating its own representation and representations (messages) from its neighbors. We denote the set of neighbors as $\mathcal{N}(u)$. This aggregation is recursively applied, so u can collect messages from its multi-hop neighbors and

produce structure-aware representations [CCV20]. With $\boldsymbol{h}_i^{(l)}$ denotes the representation of node i at iteration l, and $\text{AGGR}(\cdot,\cdot)$ denotes the aggregation operation, e.g. summation, the representation update is shown in Equation 2.1.

$$\boldsymbol{h}_u^{(l)} = \text{AGGR}(\boldsymbol{h}_u^{(l-1)}, \{\boldsymbol{h}_i^{(l-1)} | i \in \mathcal{N}(u)\}) \tag{2.1}$$

2.2.2 Cooperative Game Theory

A cooperative game denoted by (N, v), is defined by a set of players $N = \{1, \ldots, n\}$, and a *characteristic function* $v : 2^N \to \mathcal{R}$. v takes a subset of players $S \subseteq N$, called a *coalition*, and maps it to a payoff $v(S)$, where $v(\emptyset) \coloneqq 0$. A *solution function* ϕ is a function maps each given game (N, v) to $\phi(N, v) \in \mathcal{R}^n$. The vector $\phi(N, v)$, called a *solution*, represents a certain allocation of the total payoff $v(N)$ generated by all players to each individual, with the ith coordinate $\phi_i(N, v)$ being the payoff attributed to player i. $\phi(N, v)$ is also called the "value" of the game when it satisfies certain properties, and different values were proposed to name solutions with different properties [Sha53, Tel94].

The Shapley value is one popular solution of cooperative games. The main idea is to assign each player a "fair" share of the total payoff by considering all possible player interactions. For example, when player i cooperates with a coalition S, the total payoff $v(S \cup \{i\})$ may be very different from $v(S) + v(\{i\})$ because of i's interaction with S. Thus the marginal contribution of i to S is defined as by $m(i, S) = v(S \cup \{i\}) - v(S)$. Then the formula of the Shapley value for i is shown in Equation 2.3, where marginal contributions to all possible coalitions $S \subseteq N \backslash \{i\}$ are aggregated. The first identify in Equation 2.3 shows that the aggregation weights are first uniformly distributed among coalition sizes k (outer average),

then uniformly distributed among all coalitions with the same size (inner average).

$$\phi_i(N,v) = \overbrace{\frac{1}{n}\sum_{k=0}^{n-1}}^{\text{Average over } k} \overbrace{\frac{1}{\binom{n-1}{k}}\sum_{\substack{S\subseteq N\setminus\{i\}\\|S|=k}}}^{\text{Average over } S \text{ s.t. } |S|=k} m(i,S) \tag{2.2}$$

$$= \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!(n-|S|-1)!}{n!} m(i,S) \tag{2.3}$$

The Shapley value was proposed as the unique solution of a game (N,v) that satisfies three properties shown below, i.e. *efficiency*, *symmetry*, and *additivity* [Sha53]. These three properties together are referred as an axiomatic characterization of the Shapley value. The *associated consistency* properties introduced in Section 2.3.3 provides a different axiomatic characterization.

Property 2.2.1 (**Efficiency**)**.**

$$\sum_{i\in N}\phi_i(N,v) = v(N)$$

Property 2.2.2 (**Symmetry**)**.** If $v(S\cup\{i\}) = v(S\cup\{j\})$ for all $S\in N\setminus\{i,j\}$, then

$$\phi_i(N,v) = \phi_j(N,v)$$

Property 2.2.3 (**Additivity**)**.** Given two games (N,v) and (N,w),

$$\phi(N,v+w) = \phi(N,v) + \phi(N,w)$$

The efficiency property states that the value should fully distribute the payoff of the game. The symmetry property states that if two players make equal contributions to all possible coalitions formed by other players (including the empty coalition), then they should have the same value. The additivity property states that the value of two independent games should be added player by player. It is the most useful for a system of independent games.

Games with communication structures. Although the Shapley value is widely used for cooperative games, its assumption of fully flexible cooperation among all players may not be achievable. Some coalitions may be preferred over others and some may even be impossible due to limited communication among players. Thus, [Mye77] uses a graph $\mathcal{G}$ as the *communication structure* of players to represent cooperation preference. A game with a communication structure is defined by a triple $(N, v, \mathcal{G})$, with N being the node set of $\mathcal{G}$. This game formulation is more practical than fully flexible cooperation when cooperation preference is available. Several values with different properties have been proposed for such games [Mye77, BOT92, Ham99, KKU06] including the HN value [HN20].

2.3 GStarX: Graph Structure-aware Explanation

2.3.1 GNN Explanation via Feature Importance Scoring

A general approach to formalize an ML explanation problem is through feature importance scoring [LL17, DG17], where features may refer to pixels of images, words of text, or nodes/edges/subgraphs of graphs. Let $f(\cdot)$ denote a to-be-explained GNN, $\mathcal{G} = (\mathcal{V}, \boldsymbol{X}, \boldsymbol{A})$ denote an input graph, and $0 < \gamma < 1$ denote a sparsity constraint to enforce concise explanation. GNN explanation via subgraph scoring is aimed to find a subgraph g that maximizes a given evaluation metric $\textsc{Eval}\,(\cdot,\cdot,\cdot)$, which measures the faithfulness of g to $\mathcal{G}$ regarding making predictions with $f(\cdot)$, i.e.

$$g^* = \underset{g \subseteq \mathcal{G}, |g| \leq \gamma |\mathcal{G}|}{\arg\max}\ \textsc{Eval}(f(\cdot), \mathcal{G}, g) \tag{2.4}$$

When the task is graph classification and $f(\cdot)$ outputs a one-sum vector $f(\mathcal{G}) \in [0,1]^C$ containing probabilities for $\mathcal{G}$ belongs to C classes, an example $\textsc{Eval}$ can be the prediction probability drop for removing g from $\mathcal{G}$, i.e. $\textsc{Eval}(f(\cdot), \mathcal{G}, g) = [f(\mathcal{G})]_{c^*} - [f(\mathcal{G} \backslash g)]_{c^*}$ with $c^* = \arg\max_c [f(\mathcal{G})]_c$.

In practice, since the number of subgraphs is combinatorial in the number of nodes, the objective is often relaxed to finding a set of important nodes or edges first and then inducing the subgraph [YBY19, LCX20, DM21]. A more tractable objective of finding the optimal set of nodes $S^* \subseteq \mathcal{V}$ [1] is given by

$$S^* = \underset{S \subseteq \mathcal{V}, |S| \leq \gamma |\mathcal{V}|}{\arg\max} \sum_{i \in S} \textsc{Score}(f(\cdot), \mathcal{G}, i) \tag{2.5}$$

Existing methods often boil down to Equation 2.5 with different scoring functions (Score), and finding a proper Score is non-trivial. One example of Score is to evaluate each node i directly as $\textsc{Score}(f(\cdot), \mathcal{G}, i) = [f(\{i\})]_{c^*}$. However, this choice misses interactions between nodes and corresponds to a trivial case in GNNs where no message-passing is performed for $\{i\}$. Another possibility is to use Eval as Score, e.g., $\textsc{Score}(f(\cdot), \mathcal{G}, i) = [f(\mathcal{G})]_{c^*} - [f(\mathcal{G} \backslash \{i\})]_{c^*}$. However, this again fails to capture interactions between nodes; for example, two nodes i and j may be both important but also complimentary, so their contribution to $\mathcal{G}$ can only be observed when they are missing simultaneously.

2.3.2 Scoring Functions from Cooperative Games

Given the challenges for defining a proper Score, solutions to cooperative games, like the Shapley value, have been proposed with $f(\cdot)$ as the characteristic function, i.e. $\textsc{Score}(f(\cdot), \mathcal{G}, i) = \phi_i(|\mathcal{G}|, f(\cdot))$ [YYW21, DM21]. However, existing works only use the non-structure-aware Shapley value. In contrast, values defined on games $(N, v, \mathcal{G})$ with communication structures $\mathcal{G}$ are naturally structure-aware but were never considered GNN explanation. Below we discuss the non-structure-aware limitation of the Shapley value in detail and motivating structure-aware values with practical examples in GNN explanation.

[1] A similar objective can be defined as S over edges E. We define it over nodes as nodes often contain richer features than edges and are more flexible. One advantage of this choice will be made clear in Section 2.4.2

The Shapley value is defined on games (N, v), which by definition takes no graph structures. It assumes flexible cooperation between players and uniform distribution of coalition importance that only depends on $|S|$ (see Equation 2.3). Even if a $\mathcal{G}$ is given and the game is defined as $(N, v, \mathcal{G})$, the Shapley value will overlook $\mathcal{G}$ when aggregating $m(i, S)$. In contrast, structure-aware values on $(N, v, \mathcal{G})$ can be interpreted as a weighted aggregation of coalitions with more reasonable weights. Although different solutions $\phi(N, v, \mathcal{G})$ have their nuances in weight adjustments [Ham99, HN20, Mye77, KKU06], they share two key properties: **(1)** the weight is zero if i and S are disconnected because they are interpreted as players without communication channels [Mye77], and **(2)** the weight is impacted by the nature of connections between i and S because it is easier for better-connected nodes to communicate.

A synthetic example. We take the HN value (definition in Section 2.3.3) as an example structure-aware value and compare it to the Shapley value in a simple graph in Figure 2.1**(a)**. To compute $\phi_1(N, v, \mathcal{G})$, both values aggregates $m(1, S)$ for $S \in \{\emptyset, \{2\}, \{3\}, \{2, 3\}\}$. The Shapley value first assigns a uniform weight $\frac{1}{3}$ to three different $|S|$, and then splits weights uniformly for the $|S| = 1$ case to be $\frac{1}{6}$. However, the HN value assigns weight zero for $S = \{3\}$ because 1 and 3 are disconnected in coalition $\{1, 3\}$ and are assumed to be two independent graphs that shouldn't interact (property **(1)**). Their interaction is rather captured in the $S = \{2, 3\}$ case, when 1 and 3 are connected by the bridging node 2, and this case is also down-weighted from $\frac{1}{3}$ to $\frac{1}{4}$, as 3 is relatively far from 1 (property **(2)**).

A practical example. The good properties of structure-aware values can help explain graph tasks. The example in Figure 2.1**(b)** is from GraphSST2 (dataset description in Section 2.4.1), where the graph for sentiment classification is constructed from the sentence *"is still quite good-natured and not a bad way to spend an hour"* with edges generated by the Biaffine parser [GGN18]. Assuming a model can correctly classify it as positive. Intuitively, "good" and "not a bad" are central to the human explanation. To compute the Shapley value of the word "good", the coalition "not good" will diminish the positive importance of "good", despite the two words lacking any direct connection. A structure-aware value can instead eliminate

the {"not", "good"} coalition, and only consider interactions between "not" and "good" (in fact, "not" and any other word) when the bridging "bad" appears, hence better binding "not" with "bad" and improving the salience of "good". In Section 2.4.2, we revisit this example to observe impacts of structure-awareness empirically.

2.3.3 The HN Value

Let $(N, v, \mathcal{G})$ be a game with a communication structure $\mathcal{G}$ and $S \subseteq N$ be a coalition. Let $\bar{S} = \cup_{i \in S}\{\mathcal{N}(i)\} \cup S$ to be the union of S and its neighbors in $\mathcal{G}$. Let $S/\mathcal{G}$ be the partition of S containing connected components in $\mathcal{G}$, i.e.,
$S/\mathcal{G} = \{\{i | i = j$ or i and j are connected in S by E of $\mathcal{G}\} | j \in S\}$. Let $\mathcal{G}[S]$ be the induced subgraph of S in $\mathcal{G}$. For example, in Figure 2.1**(b)**, when $S =$ {"is", "an", "hour"}, $\bar{S}$ will be {"is", "good", "an", "hour", "spend"}, $S/\mathcal{G}$ will be {{"is"}, {"an", "hour"}}, and $\mathcal{G}[S]$ will be the subgraph with a two-node component an-hour and a single node component is.

Definition 2.3.1 (**Surplus**)**.** The surplus $p(j, S)$ generated by a coalition S cooperating with its neighbor j is defined as

$$p(j, S) = v(S \cup \{j\}) - v(S) - v(\{j\}) \tag{2.6}$$

Intuitively, $p(j, S)$ is generated because S is actively cooperating. Thus, when evaluating a fair payoff to S, a portion of $p(j, S)$ should be added to its own payoff $v(S)$. This idea leads to the next definition of associated games regarding the original games, where surplus allocation is performed.

Definition 2.3.2 (**HN Associated Game**)**.** Given $0 \leq \tau \leq 1$ representing the portion of surplus that will be allocated to a coalition S for its cooperation with other players. The HN

associated game $(N, v^*_\tau, \mathcal{G})$ of $(N, v, \mathcal{G})$ is defined as

$$v^*_\tau(S) = \begin{cases} v(S) + \tau \sum_{j \in \bar{S} \setminus S} p(j, S) & \text{if } |S/\mathcal{G}| = 1 \quad (2.7) \\ \sum_{T \in S/\mathcal{G}} v^*_\tau(T) & \text{otherwise} \quad (2.8) \end{cases}$$

The HN value is a solution on $(N, v, \mathcal{G})$. It is computed by iteratively constructing a series of HN associated games until it converges to a *limit game* $(N, \tilde{v}, \mathcal{G})$. In other words, we first construct v^*_τ from v by surplus allocation. Then we construct v^{**}_τ from v^*_τ by allocating the surplus generated from the v^*_τ and so on. The convergence of the limit game is guaranteed and the result $\tilde{v}$ is independent of τ under mild conditions as shown in [HN20]. The HN value of each player is uniquely determined by applying $\tilde{v}$ to that player, i.e. $\phi_i(N, v, \mathcal{G}) = \tilde{v}(\{i\})$.

Consistency and associated games. One reason for the Shapley value's popularity is its *axiomatic characterization*, indicating that it is the unique solution that satisfies a set of desirable properties. Then [Ham01] proposed a new axiomatic characterization of the Shapley value based on a different *associated consistency* property. The *consistency* property is a common analysis tool used in game theory [HM89, DM65, Sob75, Pel86]. The idea is to analyze a game (N, v) by defining other reduced games (S, v_S) for $S \subseteq N$, and a solution function ϕ is called *consistent* when $\phi(N, v)$ yields the same payoff as $\phi(S, v_S)$ on each S. When (S, v_S) is defined with desired properties, these good properties can be enforced for a solution by requiring consistency. The associated consistency in [Ham01] is a special case of consistency between (N, v) and only one other game (N, v^*), which is called the *associated game*. [Ham01] shows that a carefully designed associated game uniquely characterizes the Shapley value. Associated consistency is also the key idea of the HN value.

Limit game and the axiomatic characterization. The HN value is established on a special associated game. We can actually write this associated game in a more compact

matrix form, where we slightly abuse notation and use v and v^*_τ to represent vectors of payoffs for all $S \subseteq N$ under the original and associated game respectively. In other words, $v(S)$, which is used to represent evaluating the coalition S using the characteristic function v, now can also be interpreted as indexing the vector v with index S.

Lemma 2.3.3. *A matrix form of the associated game $(N, v^*_\tau, \mathcal{G})$ is given by*

$$v^*_\tau = \boldsymbol{H}_{\{\tau,n,\mathcal{G}\}} v \tag{2.9}$$

The matrix $\boldsymbol{H}_{\{\tau,n,\mathcal{G}\}}$ depends on the hyperparameter τ, number of players n, and the graph $\mathcal{G}$. When these variables are clear from the context, we drop them and write $v^*_\tau = \boldsymbol{H}v$. Please refer to [HN20] for the proof of Lemma 2.3.3.

With the matrix form, we can define the limit game.

Definition 2.3.4. Given a game $(N, v, \mathcal{G})$, its limit game $(N, \tilde{v}, \mathcal{G})$ is defined by

$$\tilde{v} = \lim_{p\to\infty} \boldsymbol{H}^p v \tag{2.10}$$

Notice that although the matrix $\boldsymbol{H}$ is constructed from the associated game and depends on τ, the powers of $\boldsymbol{H}$ actually converge to a limit independent from τ, when τ is sufficiently small. The general condition depends on the actual graph, but $0 < \tau < \frac{2}{n}$ is proven to be sufficient for the complete graph case [Ham01]. As we discussed in Section 2.3.3, the limit game can be seen as constructing associated games repeatedly until the characteristic function converges.

An axiomatic characterization of the HN value regarding its uniqueness is given by the following theorem based on the limit game. The associated consistency is the core property related to this work. We encourage the readers to check [HN20] for the other two properties.

Theorem 2.3.5. *There exists a unique solution ϕ that verifies the associated consistency,*

Algorithm 1 The Compute-HN-MC Function

Input: Graph instance $\mathcal{G}$ with nodes $\mathcal{V} = \{u_1, \ldots, u_n\}$, characteristic function v, hyperparameter τ, maximum sample size m, number of samples J
Let $\psi_1, \ldots, \psi_n$ be n empty lists
for $j = 1$ **to** J **do**
  Sample g_{S^j} from $\mathcal{G}$ s.t. $S^j = \{u_{j_1}, \ldots, u_{j_l}\}$ and $l < m$
  $\phi^j = \text{Compute-HN}(g_{S^j}, S^j, v(\cdot), \tau)$
  for $k = 1$ **to** l **do**
    Append ϕ^j_k to ψ_{j_k}
  end for
end for
Set ϕ_i to be the mean of ψ_i
Return: ϕ

i.e. $\phi_i(N, v, \mathcal{G}) = \phi_i(N, v^*_\tau, \mathcal{G})$*, inessential game, and continuity.* ϕ *is given by*

$$\phi_i(N, v, \mathcal{G}) = \tilde{v}(\{i\}) \tag{2.11}$$

We show the algorithm for Compute-HN-MC (Algorithm 1), which is a combination of Equation 2.9, 2.10, and 2.12.

2.3.4 GNN Message Passing and The HN Surplus Allocation

Both the GNN message passing (MP) and the associated game surplus allocation (SA) are iterative aggregation algorithms, with considerable alignment. In fact, SA on each singular node set $S = \{i\}$ is exactly MP: Equation 2.7 becomes an instantiation of Equation 2.1 with $\text{AGGR}(a, \boldsymbol{b}) = a + \tau \sum_j \boldsymbol{b}_j$ on a scalar node value a and a neighbor set $\boldsymbol{b}$. These algorithms differ in that SA applies more broadly to $|S| \geq 1$ cases; it treats S as a supernode when nodes in S form a connected component in $\mathcal{G}$, and handles disconnected S component-wise via Equation 2.8.

We illustrate SA using a real chemical graph example. The molecule shown in Figure 2.1**(c)** is taken from MUTAG (dataset description in Section 2.4.1). It is known to be

classified as *mutagenic* because of the -NO2 group (nodes 1, 2, and 3) [DCD91]. When we compute $v^*_\tau(\{1\})$, the surplus $p(2,\{1\})$, $p(3,\{1\})$, and $p(4,\{1\})$ are allocated to node 1 (like messages passed to a central node in GNN). Then surplus are aggregated together with $v(\{1\})$ following Equation 2.7 to form $v^*_\tau(\{1\})$.

For graphs, the SA approach has two advantages over the uniform aggregation approach used in the Shapley value: **(1)** The aggregated payoff in each v^*_τ is structure-aware, like representations learned by GNNs [CCV20], and **(2)** the iterative computation preserves locality, which is preserved by GNNs [BZS13]. In other words, these two properties mean close neighbors heavily influence each other due to cooperation in many iterations, while far away nodes less influence each other due to little cooperation. In the MUTAG example, since the local -NO2 generates a high payoff for the mutagenicity classification, locally allocating the payoff helps us better understand the importance of the nitrogen atom and the oxygen atoms. Whereas aggregating over many unnecessary coalitions with far-away carbon atoms can obscure the true contribution of -NO2. We will revisit this example in Section 2.4.2.

2.3.5 The GStarX Algorithm

We now state our algorithm for explaining GNNs with GStarX. Notice that GStarX scores nodes in a graph but not each dimension of node features. Feature dimension importance explanation is an orthogonal perspective that can be added on top of GStarX. We leave this extension as a future work. GStarX formulates the GNN explanation problem as a feature importance scoring problem, where nodes are scored to find the optimal node-induced subgraph. It essentially implements and solves the objective in Equation 2.5, where an HN-value-based SCORE is used. To use such SCORE, we need to define the players and the characteristic function of the game, and then apply the formula in Equation 2.7 and 2.8. Suppose the inputs are a graph $\mathcal{G}$ with nodes $\mathcal{V} = \{u_1, \ldots, u_n\}$ and label $y \in \{1, \ldots, C\}$, a GNN $f(\cdot)$ outputs a probability vector $f(\mathcal{G}) \in [0,1]^C$, and the predicted class $c^* = \arg\max_c [f(\mathcal{G})]_c$. Let $\mathcal{V}$ be players, and let the normalized probability of the

Algorithm 2 GStarX: Graph Structure-Aware Explanation

Input: Graph $\mathcal{G}$ with nodes $\mathcal{V} = \{u_1, \ldots, u_n\}$, trained GNN $f(\cdot)$, empirical expectation f^0, hyperparameter τ, max sample size m, number of samples J, sparsity γ.
Get the predicted class $c^* = \arg\max_c [f(\mathcal{G})]_c$
Define characteristic function $v(S) = [f(g_S)]_{c^*} - f^0_{c^*}$
if $n \leq m$ **then**
  $\phi = \text{Compute-HN}(\mathcal{G}, \mathcal{V}, v(\cdot), \tau)$
else
  $\phi = \text{Compute-HN-MC}(\mathcal{G}, \mathcal{V}, v(\cdot), \tau, m, J)$
end if
Sort ϕ in descending order with indices $\{\pi_1, \ldots, \pi_n\}$
$k = \lfloor \gamma |\mathcal{V}| \rfloor$
Return: $S^* = \{u_{\pi_1}, \ldots, u_{\pi_k}\}$

Algorithm 3 The Compute-HN Function

Input: Graph instance $\mathcal{G}$ with nodes $\mathcal{V} = \{u_1, \ldots, u_n\}$, characteristic function v, hyperparameter τ.
for S in 2^N **do**
  Compute payoff $v(S)$ {Eq.(2.12)}
end for
Construct matrix $\boldsymbol{H}_{\{\tau,n,\mathcal{G}\}}$
repeat
  $\boldsymbol{H} = \boldsymbol{H}\boldsymbol{H}$
until $\boldsymbol{H}$ converges
Get the limit game $\tilde{v} = \boldsymbol{H}v$
Assign the first n entries of $\tilde{v}$ to ϕ
Return: ϕ

predicted class be the characteristic function v:

$$v(S) = [f(\mathcal{G}[S])]_{c^*} - f^0_{c^*} \quad \forall S \subseteq \mathcal{V} \tag{2.12}$$

Here the normalization term $f^0_{c^*} = \mathbb{E}\left[[f(G)]_{c^*}\right]$ is the expectation over a random variable G representing a general graph. In practice, we approximate it using the empirical expectation over all $\mathcal{G}$ in the dataset. SCORE will be the HN value of the game, i.e., $\text{SCORE}(f(\cdot), \mathcal{G}, i) = \phi_i(\mathcal{V}, v, \mathcal{G}) = \tilde{v}(\{i\})$.

Given SCORE, we solve the objective by first computing the scores $\phi \in \mathcal{R}^n$ then selecting the top $\lfloor \gamma |\mathcal{V}| \rfloor$ scores greedily as in Algorithm 2. Practically, like other game-theoretic methods, the exact computation of the HN value is infeasible when the number of players n is large. We thus do an exact computation for small graphs (the if-branch) and Monte-Carlo sampling for large graphs (the else-branch). The Compute-HN function is shown in Algorithm 3, where the $\boldsymbol{H}$ stands for a matrix form of the associated game defined in Definition 2.3.2. Also, even though the algorithm is stated for graph classification, GStarX works for node classification as well. This can be easily seen since GNNs classify nodes u_i by processing an ego-graph centered at u_i, so the task can be converted to graph classification with the label of u_i used as the label of the ego-graph.

2.3.6 GStarX for Node Classification

Even though the GStarX algorithm is stated for graph classification, it works for node classification as well. This can be easily seen as the GNN node classification can be covert to classify an ego-graph. Given a graph $\mathcal{G}$ with $\mathcal{V} = \{u_1, \ldots, u_n\}$. Node classification on u_i with an L-layer GNN can be converted to a graph classification. The target graph to classify will be the L-hop ego-graph centered at u_i, because this is the receptive field of the GNN for classifying u_i and nodes further away won't influence the result. The label of the graph will be the label of u_i. In this case, the final readout layer of the GNN will be indexing u_i instead of pooling. Given this kind of conversion, everything we showed above follows.

2.4 Evaluation

2.4.1 Experiment Settings

Datasets. We conduct experiments on datasets from different domains including synthetic graphs, chemical graphs, and text graphs. A brief description of the datasets is shown below

Table 2.1: GStarX experiment dataset statistics.

Dataset	#Graphs	#Test Graphs	#Nodes (avg)	#Edges (avg)	#Features	#Classes
MUTAG	188	20	17.93	19.79	7	2
BACE	1,513	152	34.01	73.72	9	2
BBBP	2,039	200	24.06	25.95	9	2
GraphSST2	70,042	1821	9.20	10.19	768	2
Twitter	6,940	692	21.10	40.20	768	3
BA2Motifs	1,000	100	25	25.48	10	2

with detailed statistics in Table 2.1.

1. **Chemical graph property prediction.** MUTAG [DCD91], BACE and BBBP [WRF18] contain chemical molecule graphs for graph classification, with atoms as nodes, bonds as edges, and chemical properties as graph labels.

2. **Text graph sentiment classification.** GraphSST2 and Twitter [YYG20] contain graphs constructed from text. Nodes are words with pre-trained BERT embeddings as features. Edges are generated by the Biaffine parser [GGN18]. Graphs are labeled as positive or negative sentiment.

3. **Synthetic graph motif detection.** BA2Motifs [LCX20] contains graphs with a Barabasi-Albert (BA) base graph of size 20 and a 5-node motif in each graph. Node features are 10-dimensional all-one vectors. The motif can be either a house-like structure or a cycle. Graphs are labelled in two classes based on which motif they contain.

GNNs and explanation baselines. We evaluate GStarX by explaining GCNs [KW16] on all datasets in our major experiment in Section 2.4.2. In the ablation study in Section 2.4.3, we further evaluate on GIN [XHL18] and GAT [VCC17] on certain datasets following [YYW21]. We compare with 5 strong baselines representing the SOTA methods for GNN explanation: GNNExplainer [YBY19], PGExplainer [LCX20], SubgraphX [YYW21], GraphSVX [DM21],

Table 2.2: GStarX experiment GCN hyperparameters.

Dataset	#Layers	#Hidden	Pool	Test Acc
BA2Motifs	3	20	mean	0.9800
BACE	3	128	max	0.8026
BBBP	3	128	max	0.8634
MUTAG	3	128	mean	0.8500
GraphSST2	3	128	max	0.8808
Twitter	3	128	max	0.6908

Table 2.3: GStarX **experiment GIN and GAT hyperparameters.** For GAT, we use 10 attention heads with 10 dimensions each, and thus 100 hidden dimensions.

Dataset	#Layers	#Hidden	Pool	Test Acc
GraphSST2(GAT)	3	10 $\times$10	max	0.8814
MUTAG(GIN)	3	128	max	1.0

and OrphicX [LLW22]. In particular, SubgraphX and GraphSVX use Shapley-value-based scoring functions.

In Table 2.2, we provided the hyperparameters and test accuracy for the GCN model used in our major experiments. In Table 2.5, we provided the hyperparameters and test accuracy for the GIN and GAT model used in our analysis experiment. Most parameters are following [YYW21], with small changes to further boost the test accuracy.

We run all experiments on a machine with 80 Intel(R) Xeon(R) E5-2698 v4 @ 2.20GHz CPUs, and a single NVIDIA V100 GPU with 16GB RAM. Our implementations are based on Python 3.8.10, PyTorch 1.10.0, PyTorch-Geometric 1.7.1 [FL19], and DIG [LLW21]. We adapt the GNN implementation and most baseline explainer implementation from the DIG library, except for GraphSVX and OrphicX where we adapt the official implementation. For the baseline hyperparameters, we closely follow the setting in [YYW21] and [DM21] for a fair comparison.

Evaluation metrics. Evaluating explanations is non-trivial due to the lack of ground truth. We follow [YYW21, YYG20] to employ Fidelity, Inverse Fidelity (Inv-Fidelity), and

Sparsity as our evaluation metrics. Fidelity and Inv-Fidelity measure whether the prediction is faithfully important to the model prediction by removing the selected nodes or only keeping the selected nodes respectively. Sparsity promotes fair comparison by controlling explanations to have similar sizes, since including more nodes generally improves Fidelity and Inv-Fidelity, and explanations with different sizes are not directly comparable. Ideal explanations should have high Fidelity, low Inv-Fidelity, and high Sparsity, indicating relevance and conciseness. Equations 2.13-2.15 show their formulas.

$$\text{Fidelity}(\mathcal{G}, g) = [f(\mathcal{G})]_{c^*} - [f(\mathcal{G} \backslash g)]_{c^*} \tag{2.13}$$

$$\text{Inv-Fidelity}(\mathcal{G}, g) = [f(\mathcal{G})]_{c^*} - [f(g)]_{c^*} \tag{2.14}$$

$$\text{Sparsity}(\mathcal{G}, g) = 1 - |g|/|\mathcal{G}| \tag{2.15}$$

Fidelity and Inv-Fidelity are complementary and are both important for a good explanation g. Fidelity justifies the necessity for g to be included to predict correctly. Inv-Fidelity justifies the sufficiency of a standalone g to predict correctly. As they are analogous to precision and recall, we draw an analogy to the F1 score to propose a single-scalar-metric "harmonic fidelity" (H-Fidelity), where we normalize them by Sparsity and take their harmonic mean.

In Equation 2.16, 2.17, and 2.18, we show formulas for normalized fidelity (N-Fidelity), normalized inverse fidelity (N-Inv-Fidelity), and harmonic fidelity (H-Fidelity). Both the N-Fidelity and N-Inv-Fidelity are in $[-1, 1]$. The H-Fidelity flips N-Inv-Fidelity, rescales both values to be in $[0, 1]$, and takes their harmonic mean.

$$\text{N-Fidelity}(\mathcal{G}, g) = \text{Fidelity}(\mathcal{G}, g) \cdot (1 - \frac{|g|}{|\mathcal{G}|}) \tag{2.16}$$

$$\text{N-Inv-Fidelity}(\mathcal{G}, g) = \text{Inv-Fidelity}(\mathcal{G}, g) \cdot (\frac{|g|}{|\mathcal{G}|}) \tag{2.17}$$

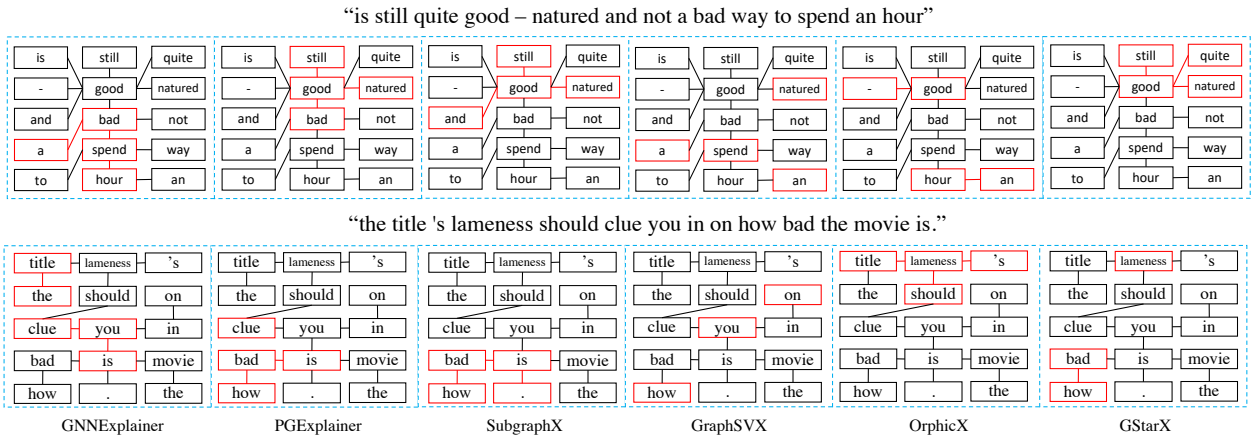


Figure 2.2: **Explanations on sentences from** GraphSST2**.** We show the explanation of one positive sentence (upper) and one negative sentence (lower). Red outlines indicate the selected nodes/edges as the explanation. GStarX identifies the sentiment words more accurately compared to baselines.

Let $m1 = \text{N-Fidelity}(\mathcal{G}, g)$, $m2 = \text{N-Inv-Fidelity}(\mathcal{G}, g)$

$$\text{H-Fidelity}(\mathcal{G}, g) = \frac{2}{(\frac{1+m1}{2})^{-1} + (\frac{1-m2}{2})^{-1}} = \frac{(1+m1)\cdot(1-m2)}{(2+m1-m2)}$$

Hyperparameters. GStarX includes three hyperparameters: τ for the allocated surplus in the associated game, m as the maximum graph size to perform exact HN value calculation, and J as the number of samples for the MC approximation. In our experiments, we choose $\tau = 0.01$ since we need $\tau < \frac{2}{n}$ for convergence and all graphs in the datasets above have less than 200 nodes. For m and J, bigger values should be better for the MC approximation, and we found $m = 10$ and $J = n$ work well empirically.

2.4.2 Evaluation Results

Quantitative studies. We report averaged test set H-Fidelity in Table 2.4. We conduct 8 different runs to get results with Sparsity ranging from 0.5-0.85 in 0.05 increments (Sparsity cannot be precisely guaranteed, hence it has minor variations across methods) and report the

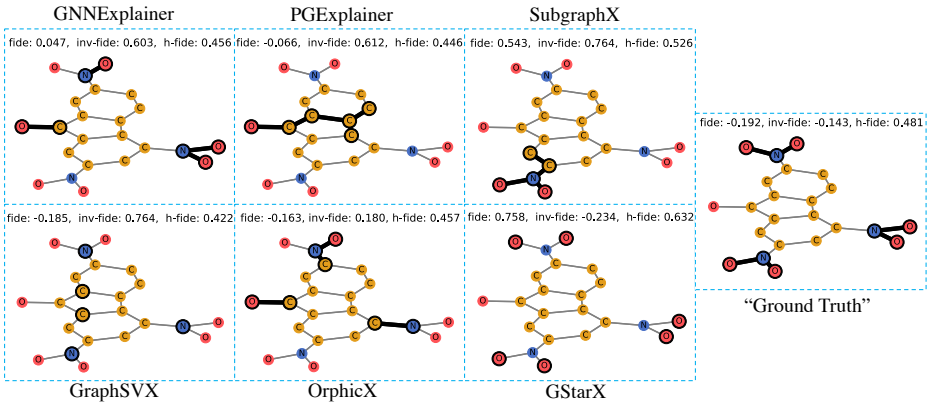


Figure 2.3: **Explanations on a mutagenic molecule in** MUTAG**.** Carbon atoms (C) are in yellow, nitrogen atoms (N) are in blue, and oxygen atoms are in red (O). Dark outlines indicate the selected nodes/edges as the explanation. We report the explanation Fidelity (fide), Inv-Fidelity (inv-fide), and H-Fidelity (h-fide). GStarX gives a significantly better explanation than other methods in terms of these metrics.

best H-Fidelity for each method. GStarX outperforms others on 4/6 datasets and has the highest average. We also follow [YYW21] to show the Fidelity vs. Sparsity plots in Figure 2.4 row1. Note that GraphSVX tends to give sparse explanations on some datasets, we still pick 8 different sparsities for it but mostly on the higher end. We also show the *1 - Inv-Fidelity* vs. sparsity plots and the H-Fidelity vs. sparsity plots. Curves in all three plots are the higher the better.

Qualitative studies. We visualize the explanations of graphs in GraphSST2 in Figure 2.2 and compare them qualitatively. We show explanations selected with high and comparable Sparsity on a positive (upper) graph and a negative (lower) graph. GStarX concisely captures the important words for sentiment classification without including extraneous ones for both sentences. Baseline methods generally select some-but-not-all important sentiment words, with extra neutral words as well. Among baselines, SubgraphX gives more reasonable results. However, it cannot cover two groups of important nodes with a limited budget because it can only select a connected subgraph as the explanation; e.g. to cover the negative word "lameness" in the lower sentence, SubgraphX needs at least three more nodes along the way,

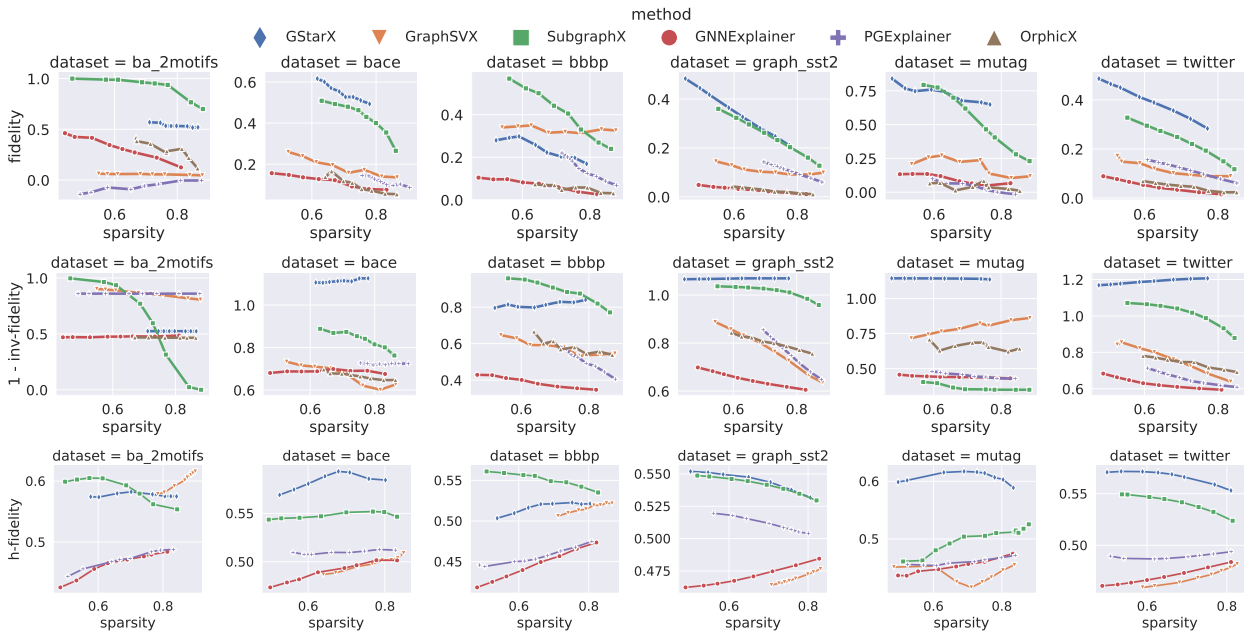


Figure 2.4: H-Fidelity **vs. sparsity for** GStarX **and baselines.** Fidelity (row1), 1 - Inv-Fidelity (row2), and H-Fidelity (row3) vs. Sparsity on all datasets corresponding to the results shown in Table 2.4. All three metrics are the higher the better. We see that GStarX outperforms the other methods

which will significantly decrease Sparsity while including undesirable, neutral words. Moreover, we discussed in Section 2.3.2 that the Shapley value will downgrade the positive importance of the word "good" for the upper sentence. Comparing the normalized contribution scores of our HN-value-based method GStarX and the Shapley-based method GraphSVX, contribution of "good" is higher in ours: 0.1152 vs. 0.0371.

We visualize explanations selected with high and comparable Sparsity of a mutagenic molecule from MUTAG in Figure 2.3. Explanations on chemical graphs are harder to evaluate than text graphs as they require domain knowledge. MUTAG has been widely used as a benchmark for evaluating GNN explanations because human experts recognize -NO2 as mutagenic [DCD91], which makes MUTAG a dataset with "ground truth"[2]. Surprisingly, we

[2]Carbon rings were also claimed as mutagenic by human experts, but we found it is not discriminative as they exist in both mutagenic and non-mutagenic molecules in MUTAG.

Table 2.4: **The best** H-Fidelity **(higher is better) of 8 different** Sparsity **for each dataset.** GStarX shows higher H-Fidelity on average and on 4/6 datasets.

Dataset	GNNExplainer	PGExplainer	SubgraphX	GraphSVX	OrphicX	GStarX
BA2Motifs	0.4841	0.4879	**0.6050**	0.5017	0.5087	0.5824
BACE	0.5016	0.5127	0.5519	0.5067	0.4960	**0.5934**
BBBP	0.4735	0.4750	**0.5610**	0.5345	0.4893	0.5227
GraphSST2	0.4845	0.5196	0.5487	0.5053	0.4924	**0.5519**
MUTAG	0.4745	0.4714	0.5253	0.5211	0.4925	**0.6171**
Twitter	0.4838	0.4938	0.5494	0.4989	0.4944	**0.5716**
Average	0.4837	0.4934	0.5569	0.5114	0.4952	**0.5732**

found that GStarX generates much better H-Fidelity/Fidelity/Inv-Fidelity than other methods and even the "ground truth" by only selecting the -O in -NO2 as explanations. In particular, the -0.234 Inv-Fidelity of GStarX means the selected subgraph has an even better prediction result than the original whole graph (0 Inv-Fidelity) and the ground truth (-0.143 Inv-Fidelity) because nodes not significant to the GNN prediction are removed. Fidelity metrics of baselines are inferior to GStarX because they include other non-discriminative carbon atoms despite they capture -NO2 to some extent. This suggests that even though **human experts identify -NO2** as the "ground truth" of mutagenicity, the **GNN only needs -O** to classify mutagenic molecules. With the goal being understand model behavior, GStarX explanation is better. Moreover, SubgraphX is the only baseline that has better H-Fidelity than the "ground truth", but it can only capture one -NO2 because its search algorithm requires the explanation to be connected, so its Inv-Fidelity is not optimal. In fact, GNNExplainer, PGExplainer, and SubgraphX can never generate explanations including only disconnected -O without -N like GStarX, because the former two solve the explanation problem by optimizing edges (as opposed to Equation 2.5), and the latter requires connectedness. More MUTAG explanation visualizations are provided in Appendix A.1.2.

Table 2.5: **Ablation on GNN architectures for** GStarX**.** GStarX shows higher H-Fidelity for both GAT on GraphSST2 and GIN on MUTAG.

Dataset	GNNExplainer	PGExplainer	SubgraphX	GraphSVX	OrphicX	GStarX
GraphSST2	0.4951	0.4918	0.5484	0.5132	0.4997	**0.5542**
MUTAG	0.5042	0.4993	0.5264	0.5592	0.5152	**0.6064**

Table 2.6: Comparison of average running time on 50 graphs in BBBPbetween GStarX and baselines.

Method	GNNExp	PGExp	SubgX	GraphSVX	OrphicX	GStarX
Time(s)	11.92	0.03 (train 720)	75.96	3.06	0.15 (train 915)	31.24

2.4.3 Ablation Studies and Analysis

Model-agnostic explanation. GStarX makes no assumptions about the model architecture and can be applied to explain various GNN backbones. We use GCN for all datasets in the major experiment above for consistency, and we now further investigate performance on two more popular GNNs: GIN and GAT. We follow [YYW21] to train GIN on MUTAG and GAT on GraphSST2[3], and show results in Table 2.5. For both settings, GStarX outperforms the baselines, which is consistent with results on GCN.

Efficiency study. The GStarX algorithm scales in $O(J)$ with practical $J \propto |\mathcal{V}|$. Following [YYW21], we study the empirical efficiency of GStarX by explaining 50 randomly selected graphs from BBBP. We report the average run time in Table 2.6. Our results for the baselines are similar to [YYW21]. GStarX is not the fastest method, but it is more than two times faster than SubgraphX. Since explanation usually doesn't have strict efficiency requirements in real applications, considering GStarX generates higher-quality explanations than the baselines, we believe the time complexity of GStarX is acceptable.

Explanation sparsity study. To further study whether the obtained scores by GStarX are sparse, we follow [FKA21] to evaluate an entropy-based sparsity measure on model

[3]As some baselines take over 24 hours on full GraphSST2, we randomly select 30 graphs for this analysis.

output scores. We show the average GStarX entropy-based sparsity on all datasets, and compare them with three reference score distributions on all n nodes in a graph. 1) An upper bound: Uniform(n), which represents the least sparse output. 2) A practical lower bound: Uniform(0.25*n) which represents very sparse outputs with only top 25% of nodes. 3) Poisson(0.25*n), which is a more realistic version of case 2). Results in Table 2.7 show the average entropy-based sparsity of GStarX is much lower than Uniform(n) and close to Poisson(0.25*n), which justifies the GStarX outputs are indeed sparse.

The entropy-based sparsity, as defined in Definition 2 in [FKA21], is shown in the Equation 2.18 below. Here ϕ is the model output scores for a data instance, and $\tilde{\phi}_i = \frac{\phi_i}{\sum_i \phi_i}$ represent normalized scores.

$$H(\tilde{\phi}) = -\sum_{i \in n} \tilde{\phi}_i \log \tilde{\phi}_i \tag{2.18}$$

The entropy-based sparsity helps us to understand how sparse an explanation is, before the scores are turned into hard explanation by thresholding or selecting top k. In Table 2.7, we show the average scores for GStarX on all datasets, and compare them with three reference cases. 1) The entropy of uniform distribution over all n nodes in a graph, i.e., Uniform(n), which represents the least sparse output and is an upper bound of entropy-based sparsity. 2) The entropy of uniform distribution over the top 25% nodes in a graph, i.e., Uniform(0.25*n), where probabilities of the bottom 75% nodes are set to zero. This case is very sparse since 75% of nodes are deterministically excluded, which can be treated as a practical lower bound of entropy-based sparsity. 3) The entropy of Poisson distribution with mean 0.25*n, i.e. Poisson(0.25*n), which is a more realistic version of the sparse output in case 2). Instead of setting all 75% of nodes to have probability zero, we assume the probabilities for tail nodes decrease exponentially as a Poisson distribution while the mean is kept the same as in case 2). Results in Table 2.7 show that the average entropy-based sparsity of GStarX is between Uniform(0.25*n) and Uniform(n) and close to Poisson(0.25*n), which justifies the GStarX

Table 2.7: **The entropy-based sparsity scores of** GStarX vs. three reference distributions, which shows GStarX outputs are indeed sparse.

Dataset	BA2Motifs	BACE	BBBP	GraphSST2	MUTAG	Twitter
GStarX	2.1352	2.4481	2.3290	2.3282	2.2434	2.2114
Uniform(n)	3.2189	3.5080	3.0728	2.8698	2.8612	2.9833
Uniform(0.25*n)	1.8326	2.1217	1.6893	1.4855	1.4749	1.5970
Poisson(0.25*n)	2.3204	2.4686	2.2416	2.1336	2.1323	2.1945

outputs are indeed sparse.

2.5 Related Work

GNN explanation aims to produce an explanation for a GNN prediction on a given graph, usually as a subgraph induced by important nodes or edges. Many existing methods work by scoring nodes or edges and are thus similar to this work. For example, the scoring function of GNNExplainer [YBY19] is the mutual information between a masked graph and the prediction on the original graph, where soft masks on edges and node features are generated by direct parameter learning. PGExplainer [LCX20] uses the same scoring function as [YBY19] but generates a discrete mask on edges by training an edge mask predictor. SubgraphX [YYW21] uses the Shapley value as its scoring function on subgraphs selected by Monte Carlo Tree Search (MCTS), and GraphSVX [DM21] uses a least-square approximation to the Shapley value to score nodes and their features. While SubgraphX and GraphSVX were shown to perform better than prior alternatives, as we show in Section 2.3.1, the Shapley value they try to approximate is non-ideal as it is non-structure-aware. Although SubgraphX and GraphSVX use L-hop subgraphs and thus technically they use the graph structure, such structure usage are very limited in achieving structure-awareness as we show in Appendix A.1.1.3.

Besides the perturbation-based methods, there are several other types of approaches for GNN explanation. Gradient-based methods are widely used for explaining ML models

on images and text. The key idea is to use the gradients as the approximations of input importance. Such methods as contrastive gradient-based (CG) saliency maps, Class Activation Mapping (CAM), and gradient-weighted CAM (Grad-CAM) have been generalized to graph data in [PKR19]. Decomposition-based methods are a popular way to explain deep NNs for images. They measure the importance of input features by decomposing the model predictions and regard the decomposed terms as importance scores. Decomposition methods including Layer-wise Relevance Propagation (LRP) and Excitation Backpropagation (EB) have also been extended to graphs [PKR19, BA19]. Surrogate-based methods work by approximating a complex model using an explainable model locally. Possible options to approximate GNNs include linear model as in GraphLIME [HYT20], additive feature attribution model with the Shapley value as in GraphSVX [DM21], and Bayesian networks as in [VT20]. GNN explainability has also been studied from the causal perspective. In [LLL21, LLW22], generative models were constructed to learn causal factors, and explanations were produced by analyzing the cause-effect relationship in the causal graph.

Cooperative game theory originally studies how to allocate payoffs among a set of players in a cooperative game. Recently, certain ideas from this domain have been successfully used in feature importance scoring for ML model explanation [LC01, SK14, LL17]. When used for model explanation, data features becomes players in the game, e.g. pixels for images, and the value of the game gives feature importance scores. The vast majority of works in this line, like the ones cited above, deem the Shapley value [Sha53] to be the only choice. In fact, there are many other values with different properties and used in different situations in cooperative game theory. However, to the best of our knowledge, only [CSW19] mentions the Myerson value [Mye77] in the context of proposing a connected Shapley (C-Shapley) value for explaining sequence data, and it is not directly comparable to ours for graph data. A detailed discussion of the Myerson value and the C-Shapley value can be found in Appendix A.1.1. Our work follows the cooperative game theory approach to explain models on graph data using the HN value [HN20], which as we show is a better choice than the Shapley value

given its structure-awareness.

2.6 Discussion

In summary, we study GNN explanation on graphs via node importance scoring. We identify the non-structure-aware challenge of existing Shapley-value-based approaches and propose GStarX to assign importance scores to each node via a structure-aware HN value. We also build connections between the HN value surplus allocation and GNN message passing. GStarX demonstrates its superiority over strong baselines on chemical and text graph classifications. A limitation of GStarX is that the importance of different node feature dimensions is not explained. One future work is to add this extension, which could be done by scoring a subset of nodes together with a subset of features each time. Another future direction is to exploit the rich cooperative game theory literature. Beyond the Shapley value, more values are possible for explaining ML models. For graph data, edge-based values like [BOT92] can potentially be applied to an alternative edge-based objective like Equation 2.5. Other values may be appropriate to more data types beyond graphs. A relevant discussion of Myerson values, C-Shapley values, and L-hop graph cut-off for approximating the Shapley value is provided in Appendix A.1.1.

Part II

Explanations Establish User Trust

CHAPTER 3

Path-based GNN Explanation for Link Prediction

3.1 Introduction

Transparency and accountability are significant concerns when researchers advance black-box ML models [SP19, LOL18]. Good explanations of model behavior improve model transparency. For end users, explanations make them trust the predictions and increase their engagement and satisfaction [HKR00, BM05]. For researchers and developers, explanations enable them to understand the decision-making process and create accountable ML models. GNNs [WPC20, ZCH20] have recently achieved state-of-the-art performance on many graph ML tasks and attracted increased interest in studying their explainability [YBY19, LCX20, ZLS22, YYG22]. However, to our knowledge, GNN explanation for link prediction (LP) is missing in the literature. LP is an essential task of many vital Web applications like recommendation [ZSZ19, MZX21, WSZ20] and sponsored search [LPL21, HCS21]. GNNs are widely used to solve LP problems [ZC18, ZZX21], and generating good GNN explanations for LP will benefit these applications, e.g., increasing user satisfaction with recommended items.

Existing GNN explanation methods have addressed node/graph-level tasks on homogeneous graphs. Given a data instance, most methods generate an explanation by learning a mask to select an edge-induced subgraph [YBY19, LCX20] or searching over the space of subgraphs [YYW21]. However, explaining GNNs for LP is a new and more challenging task. Existing node/graph-level explanation methods do not generalize well to LP for three challenges. 1) *Connection Interpretability*: LP involves a pair of the source node and the

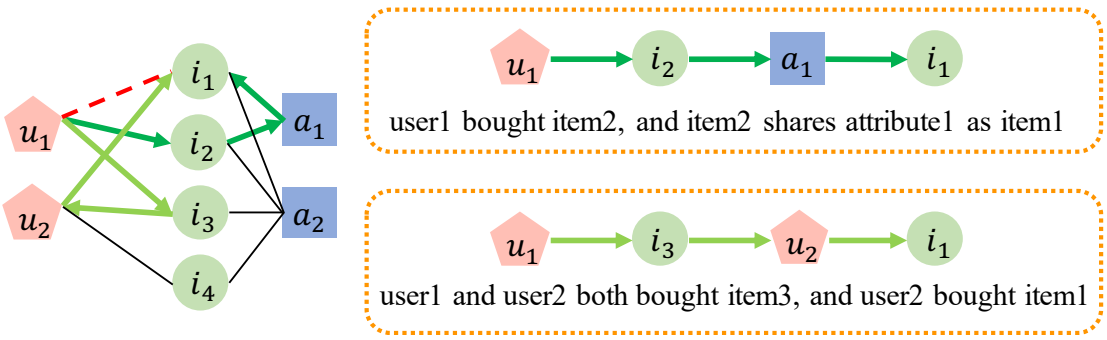


Figure 3.1: **Path-based explanations generated by** PaGE-Link**.** Given a GNN model and a predicted link (u_1, i_1) (dashed red) on a heterogeneous graph of user u, item i, and attribute a (left). PaGE-Link generates two path explanations (green arrows). We illustrate the interpretations on the right.

target node rather than a single node or graph. Desired interpretable explanations for a predicted link should reveal connections between the node pair. Existing methods generate subgraphs with no format constraints, so they are likely to output subgraphs disconnected from the source, the target, or both. Without revealing connections between the source and the target, these subgraph explanations are hard for humans to interpret and investigate. 2) *Scalability*: For LP, the number of edges involved in GNN computation almost grows from m to $\sim 2m$ compared to the node prediction task because neighbors of both the source and the target are involved. Since most existing methods consider all (edge-induced) subgraphs, the increased edges will scale the number of subgraph candidates by a factor of $O(2^m)$, which makes finding the optimal subgraph explanation much harder. 3) *Heterogeneity*: Practical LP is often on heterogeneous graphs with rich node and edge types, e.g., a graph for recommendations can have user->buys->item edges and item->has->attribute edges, but existing methods only work for homogeneous graphs.

In light of the importance and challenges of GNN explanation for LP, we formulate it as a post hoc and instance-level explanation problem and generate explanations for it in the form of important paths connecting the source node and the target node. Paths have played substantial roles in graph ML and are the core of many non-GNN LP methods [LK07, Kat53, JW02, SHY11]. Paths as explanations can solve the connection interpretability and

scalability challenges. Firstly, paths connecting two nodes naturally explain connections between them. Figure 3.1 shows an example on a graph for recommendations. Given a GNN and a predicted link between user u_1 and item i_1, human-interpretable explanations may be based on the user's preference of attributes (e.g., user u_1 bought item $i2$ that shared the same attribute a_1 as item i_1) or collaborative filtering (e.g, user u_1 had a similar preference as user u_2 because they both bought item i_3 and user u_2 bought item i_1, so that user u_1 would like item i_1). Both explanations boil down to paths. Secondly, paths have a considerably smaller search space than general subgraphs. As we will see in Proposition 3.4.1, compared to the expected number of edge-induced subgraphs, the expected number of paths grows strictly slower and becomes negligible. Therefore, path explanations exclude many less-meaningful subgraph candidates, making the explanation generation much more straightforward and accurate.

To this end, we propose Path-based GNN Explanation for heterogeneous Link prediction (PaGE-Link), which achieves a better explanation AUC and scales linearly in the number of edges (see Figure 3.2). We first perform *k-core pruning* [Bol84] to help find paths and improve scalability. Then we do *heterogeneous path-enforcing* mask learning to determine important paths, which handles heterogeneity and enforces the explanation edges to form paths connecting source to target. In summary, the contributions of our method are:

- Connection Interpretability: PaGE-Link produces more interpretable explanations in path forms and quantitatively improves explanation AUC over baselines.
- Scalability: PaGE-Link reduces the explanation search space by magnitudes from subgraph finding to path finding and scales linearly in the number of graph edges.
- Heterogeneity: PaGE-Link works on heterogeneous graphs and leverages edge-type information to generate better explanations.

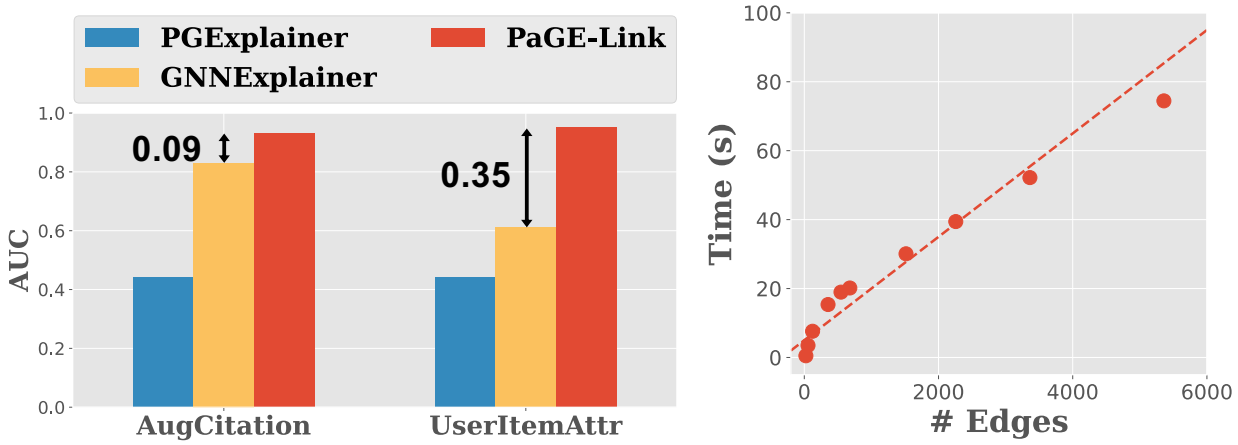


Figure 3.2: **Accuracy and run-time comparison between** PaGE-Link **and baselines.** (a) PaGE-Link outperforms GNNExplainer and PGExplainer in terms of explanation AUC on the citation graph and the user-item graph. (b) The running time of PaGE-Link scales linearly in the number of graph edges.

3.2 Related Work

We review relevant research on (a) GNNs (b) GNN explanation (c) recommendation explanation and (d) paths for LP. We summarize the properties of PaGE-Link vs. representative methods in Table 3.1.

GNNs GNNs are a family of ML models on graphs [KW16, VCC17, XHL18]. They take graph structure and node/edge features as input and output node representations by transforming and aggregating features of nodes' (multi-hop) neighbors. The node representations can be used for LP and achieved great results on LP applications [ZLX20, ZC18, ZSZ19, MZX21, WSZ20, ZLW22, GSZ22]. We review GNN-based LP models in Section 3.3.

GNN explanation GNN explanation was studied for node and graph classification, where the explanation is defined as an important subgraph. Existing methods majorly differ in their definition of importance and subgraph selection methods. GNNExplainer [YBY19] selects edge-induced subgraphs by learning fully parameterized masks on graph edges and

node features, where the mutual information (MI) between the masked graph and the prediction made with the original graph is maximized. PGExplainer [LCX20] adopts the same MI importance but trains a mask predictor to generate a discrete mask instead. Other popular importance measures are game theory values. SubgraphX [YYW21] uses the Shapley value [Sha53] and performs Monte Carlo Tree Search (MCTS) on subgraphs. GStarX [ZLS22] uses a structure-aware HN value [HN20] to measure the importance of nodes and generates the important-node-induced subgraph. There are more studies from other perspectives that are less related to this work, i.e., surrogate models [HYT20, VT20], counterfactual explanations [LTT22], and causality [LLL21, LLW22], for which [YYG20] provides a good review. While these methods produce subgraphs as explanations, what makes a good explanation is a complex topic, especially how to meet "stakeholders' desiderata" [LOS21]. Our work differs from all above since we focus on a new task of explaining heterogeneous LP, and we generate paths instead of unrestricted subgraphs as explanations. The interpretability of paths makes our method advantaged especially when stakeholders have less ML background.

Recommendation explanation This line of works explains why a recommendation is made [ZC20]. J-RECS [PKF20] generates recommendation explanations on product graphs using a justification score that balances item relevance and diversity. PRINCE [GBS20] produces end-user explanations as a set of minimal actions performed by the user on graphs with users, items, reviews, and categories. The set of actions is selected using counterfactual evidence. Typically, recommendations on graphs can be formalized as an LP task. However, the recommendation explanation problem differs from explaining GNNs for LP because the recommendation data may not be graphs, and the models to be explained are primarily not GNN-based [WWX19]. GNNs have their unique message passing procedure, and GNN-based LP corresponds to more general applications beyond recommendation, e.g., drug repurposing [IZK20], and knowledge graph completion [NMT15, CYZ21]. Thus, recommendation explanation is related to but not directly comparable to GNN explanation.

Table 3.1: **Methods and desired explanation properties.** A question mark (**?**) means "unclear", or "maybe, after non-trivial extensions". "Rec. Exp." stands for the general recommendation explanation methods.

Methods	GNNExp [YBY19]	PGExp [LCX20]	SubgraphX [YYW21]	J-RECS [PKF20]	PRINCE [GBS20]	Rec. Exp. [ZC20]	PaGE-Link
On Graphs	✓	✓	✓	✓	✓	?	✓
Explains GNN	✓	✓	✓				✓
Explains LP	?	?	?	✓	✓	✓	✓
Connection				?	?	?	✓
Scalability	✓	✓		✓	?	?	✓
Heterogeneity			✓	✓	✓	?	✓

Paths Paths are important in graph ML, and many LP methods are path-based, such as graph distance [LK07], Katz index [Kat53], SimRank [JW02], and PathSim [SHY11]. Paths have also been used to capture the relationship between a pair of nodes. For example, the "connection subgraphs" [FMT04] find paths between the source and the target based on electricity analogs. In general, although black-box GNNs recently outperform path-based methods in LP accuracy, we embrace paths for their interpretability for LP explanation.

3.3 Preliminaries

In this section, we define necessary notations, summarize them in Table 3.2, and review the GNN-based LP models.

Definition 3.3.1. A heterogeneous graph is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathrm{E})$ associated with a node type mapping function $\phi : \mathcal{V} \to \mathcal{A}$ and an edge type mapping function $\tau : \mathrm{E} \to \mathcal{R}$. Each node $v \in \mathcal{V}$ belongs to one node type $\phi(v) \in \mathcal{A}$ and each edge $e \in \mathrm{E}$ belongs to one edge type $\tau(e) \in \mathcal{R}$.

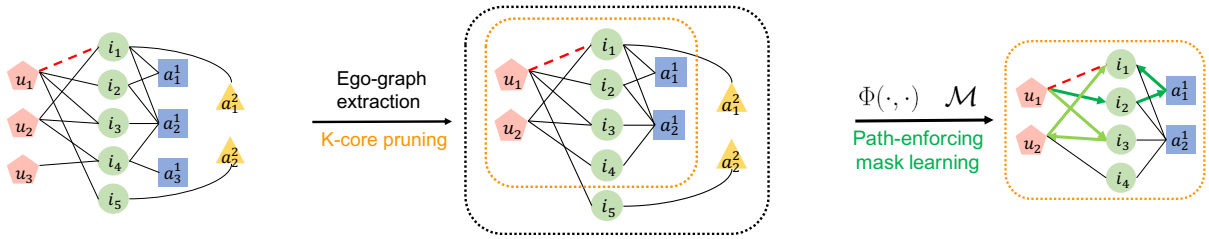
Let $\Phi(\cdot, \cdot)$ denote a trained GNN-based model for predicting the missing links in $\mathcal{G}$, where

Table 3.2: PaGE-Link notations.

Notation	Definition and description
$\mathcal{G} = (\mathcal{V}, \mathrm{E})$	a heterogeneous graph $\mathcal{G}$, node set $\mathcal{V}$, and edge set E
$\phi : \mathcal{V} \to \mathcal{A}$	a node type mapping function
$\tau : \mathrm{E} \to \mathcal{R}$	an edge type mapping function
D_v	the degree of node $v \in \mathcal{V}$
E^r	edges with type $r \in \mathcal{R}$, i.e., $\mathrm{E}^r = \{e \in \mathrm{E} \vert \tau(e) = r\}$
$\Phi(\cdot, \cdot)$	the GNN-based LP model to explain
(s, t)	the source and target node for the predicted link
$\boldsymbol{h}_s$ & $\boldsymbol{h}_t$	the node representations for s & t
$Y = \Phi(\mathcal{G}, (s, t))$	the link prediction of the node pair (s, t)
$\mathcal{G}_c = (\mathcal{V}_c, \mathrm{E}_c)$	the computation graph, i.e., L-hop ego-graph of (s, t)

a prediction $Y = \Phi(\mathcal{G}, (s, t))$ denotes the predicted link between a source node s and a target node t. The model Φ learns a conditional distribution $P_\Phi(Y|\mathcal{G}, (s, t))$ of the binary random variable Y. The commonly used GNN-based LP models [ZC18, ZZX21, ZLW22] involve two steps. The first step is to generate node representations $(\boldsymbol{h}_s, \boldsymbol{h}_t)$ of (s, t) with an L-hop GNN encoder. The second step is to apply a prediction head on $(\boldsymbol{h}_s, \boldsymbol{h}_t)$ to get the prediction of Y. An example prediction head is an inner product.

To explain $\Phi(\mathcal{G}, (s, t))$ with an L-Layer GNN encoder, we restrict to the *computation graph* $\mathcal{G}_c = (\mathcal{V}_c, \mathrm{E}_c)$. $\mathcal{G}_c$ is the L-hop ego-graph of the predicted pair (s, t), i.e., the subgraph with node set $\mathcal{V}_c = \{v \in V | dist(v, s) \leq L \text{ or } dist(v, t) \leq L\}$. It is called a computation graph because the L-layer GNN only collects messages from the L-hop neighbors of s and t to compute $\boldsymbol{h}_s$ and $\boldsymbol{h}_t$. The LP result is thus fully determined by $\mathcal{G}_c$, i.e., $\Phi(\mathcal{G}, (s, t)) \equiv \Phi(\mathcal{G}_c, (s, t))$. Figure 3.3b shows a 2-hop ego-graph of u_1 and i_1, where u_3 and a_3^1 are excluded since they are more than 2 hops away from either u_1 or i_1.



(a) A GNN predicted link (u_1, i_1) (dashed red) that needs explanation.

(b) Extract 2-hop ego-graph of (u_1, i_1) excluding u_3 and a_3^1 (black box). Then prune it to get the k-core excluding i_5, a_1^2, and a_2^2 (orange box).

(c) Human-interpretable path explanations (u_1, i_2, a_1^1, i_1) and (u_1, i_3, u_2, i_1) (green arrows) that capture the connection between u_1 and i_1.

Figure 3.3: **Illustration of the** PaGE-Link **framework.** PaGE-Link on a graph with user nodes u, item nodes i, and two attribute types a^1 and a^2.

3.4 PaGE-Link: Path-based GNN Explanation for Link Prediction

3.4.1 Link-Prediction Explanation

In this work, we address a *post hoc* and *instance-level* GNN explanation problem. The post hoc means the model $\Phi(\cdot, \cdot)$ has been trained. To generate explanations, we won't change its architecture or parameters. The instance level means we generate an explanation for the prediction of each instance (s, t). Specifically, the explanation method answers the question of why a missing link is predicted by $\Phi(\cdot, \cdot)$. In a practical web recommendation system, this question can be "*why an item is recommended to a user by the model*".

An explanation for a GNN prediction should be some substructure in $\mathcal{G}_c$, and it should also be concise, i.e., limited by a size budget B. This is because an explanation with a large size is often neither informative nor interpretable, for example, an extreme case is that $\mathcal{G}_c$ could be a non-informative explanation for itself. Also, a fair comparison between different explanations should consume the same budget. In the following, we define budget B as the maximum number of edges included in the explanation.

We list three desirable properties for a GNN explanation method on heterogeneous LP: capturing the connection between the source node and the target node, scalable to

large graphs, and addressing graph heterogeneity. Using a path-based method inherently possesses all the properties. Paths capture the connection between a pair of nodes and can be transferred to human-interpretable explanations. Besides, the search space of paths with the fixed source node and the target node is greatly reduced compared to edge-induced subgraphs. Given the ego-graph $\mathcal{G}_c$ of s and t, the number of paths between s and t and the number of edge-induced subgraphs in $\mathcal{G}_c$ both rely on the structure of $\mathcal{G}_c$. However, they can be estimated using random graph approximations. The next proposition on random graphs shows that the expected number of paths grows strictly slower than the expected number of edge-induced subgraphs as the random graph grows. Also, the expected number of paths becomes insignificant for large graphs.

Proposition 3.4.1. *Let $\mathcal{G}(n, d)$ be a random graph with n nodes and density d, i.e., there are $m = d\binom{n}{2}$ edges chosen uniformly randomly from all node pairs. Let $Z_{n,d}$ be the expected number of paths between any pair of nodes. Let $S_{n,d}$ be the expected number of edge-induced subgraphs. Then $Z_{n,d} = o(S_{n,d})$, i.e., $\lim_{n\to\infty} \frac{Z_{n,d}}{S_{n,d}} = 0$.*

Proof. In Appendix A.2.1. □

Paths are also a natural choice for LP explanations on heterogeneous graphs. On homogeneous graphs, features are important for prediction and explanation. A s-t link may be predicted because of the feature similarity of node s and node t. However, the heterogeneous graphs we focus on, as defined in Definition 3.3.1, often do not store feature information but explicitly model it using new node and edge types. For example, for the heterogeneous graph in Figure 3.3a, instead of making it a user-item graph and assigning each item node a two-dimensional feature with attributes a^1 and a^2, the attribute nodes are explicitly created and connected to the item nodes. Then an explanation like "i_1 and i_2 share node feature a_1^1" on a homogeneous graph is transferred to "i_1 and i_2 are connected through the attribute node a_1^1" on a heterogeneous graph.

Given the advantages of paths over general subgraphs on connection interpretability,

scalability, and their capability to capture feature similarity on heterogeneous graphs, we use paths to explain GNNs for heterogeneous LP. Our design principle is that a good explanation should be concise and informative, so we define the explanation to contain only *short* paths *without high-degree* nodes. Long paths are less desirable since they could correspond to unnecessarily complicated connections, making the explanation neither concise nor convincing. For example, in Figure 3.3c, the long path $(u_1, i_3, a_2^1, i_2, a_1^1, i_1)$ is not ideal since it takes four hops to go from item i_3 to the item i_1, making it less persuasive to be interpreted as "item1 and item3 are similar so item1 should be recommended". Paths containing high-degree nodes are also less desirable because high-degree nodes are often generic, and a path going through them is not as informative. In the same figure, all paths containing node a_2^1 are less desirable because a_2^1 has a high degree and connects to all the items in the graph. A real example of a generic attribute is the attribute "grocery" connecting to both "vanilla ice cream" and "vanilla cookie". When "vanilla ice cream" is recommended to a person who bought "vanilla cookie", explaining this recommendation with a path going through "grocery" is not very informative since "grocery" connects many items. In contrast, a good informative path explanation should go through the attribute "vanilla", which only connects to vanilla-flavored items and has a much lower degree.

We formalize the GNN explanation for heterogeneous LP as:

Problem 3.4.2. Generating path-based explanations for a predicted link between node s and t:

- **Given**
  - a trained GNN-based LP model $\Phi(\cdot, \cdot)$,
  - a heterogeneous computation graph $\mathcal{G}_c$ of s and t,
  - a budget B of the maximum number of edges in the explanation,
- **Find** an explanation $\mathrm{P} = \{\ p | p$ is a s-t path with maximum length l_{max} and degree of each node less than $D_{max}\ \}$, $|\mathrm{P}| l_{max} \leq B$,

- **By optimizing** $p \in \mathrm{P}$ to be influential to the prediction, concise, and informative.

3.4.2 K-core Pruning

The *k-core pruning* module of PaGE-Link reduces the complexity of $\mathcal{G}_c$. The k-core of a graph is defined as the unique maximal subgraph with a minimum node degree k [Bol84]. We use the superscript k to denote the k-core, i.e., $\mathcal{G}_c^k = (\mathrm{E}_c^k, \mathcal{V}_c^k)$ for the k-core of $\mathcal{G}_c$. The k-core pruning is a recursive algorithm that removes nodes $v \in \mathcal{V}$ such that their degrees $D_v < k$, until the remaining subgraph only has nodes with $D_v \geq k$, which gives the k-core. The difference in nodes between a $(k+1)$-core and a k-core is called the k-shell. The nodes in the orange box of Figure 3.3b is an example of a 2-core pruned from the 2-hop ego-graph, where node a_1^2 and a_2^2 are pruned in the first iteration because they are degree one. Node i_5 is recursively pruned because it becomes degree one after node a_2^2 is pruned. All those three nodes belong to the 1-shell. We perform k-core pruning to help path finding because the pruned k-shell nodes are unlikely to be part of meaningful paths when k is small. For example, the 1-shell nodes are either leaf nodes or will become leaf nodes during the recursive pruning, which will never be part of a path unless s or t are one of these 1-shell nodes. The k-core pruning module in PaGE-Link is modified from the standard k-core pruning by adding a condition of never pruning s and t.

The following theorem shows that for a random graph $\mathcal{G}(n, d)$, k-core will reduce the expected number of nodes by a factor of $\delta_{\mathcal{V}}(n, d, k)$ and reduce the expected number of edges by a factor of $\delta_{\mathrm{E}}(n, d, k)$. Both factors are functions of n, d, and k. We defer the exact expressions of these two factors in Appendix A.2.2, since they are only implicitly defined based on Poisson distribution. Numerically, for a random $\mathcal{G}(n, d)$ with average node degree $d(n-1) = 7$, its 5-core has $\delta_{\mathcal{V}}(n, d, 5)$ and $\delta_{\mathrm{E}}(n, d, 5)$ both ≈ 0.69.

Theorem 3.4.3 (Pittel, Spencer and Wormald [PSW96])**.** *Let $\mathcal{G}(n, d)$ be a random graph with m edges as in Proposition 3.4.1. Let $\mathcal{G}^k(n, d) = (\mathcal{V}^k(n, d), \mathrm{E}^k(n, d))$ be the nonempty*

k-core of $\mathcal{G}(n,d)$. Then $\mathcal{G}^k(n,d)$ contain $\delta_{\mathcal{V}}(n,d,k)n$ nodes and $\delta_{\mathrm{E}}(n,d,k)m$ edges with high probability for large n, i.e., $|\mathcal{V}^k(n,d)|/n \xrightarrow{p} \delta_{\mathcal{V}}(n,d,k)$ and $|\mathrm{E}^k(n,d)|/m \xrightarrow{p} \delta_{\mathrm{E}}(n,d,k)$ ($\xrightarrow{p}$ stands for convergence in probability).

Proof. Please refer to Appendix A.2.2 and [PSW96]. □

The k-core pruning helps reduce the graph complexity and accelerates path finding. One concern is whether it prunes too much and disconnects s and t. We found that such a situation is very unlikely to happen in practice. To be specific, we focus on explaining positively predicted links, e.g. why an item is recommended to a user by the model. Negative predictions, e.g., why an arbitrary item is not recommended to a user by the model, are less useful in practice and thus not in the scope of our explanation. (s,t) node pairs are usually connected by many paths in a practical $\mathcal{G}$ [WS98], and positive link predictions are rarely made between disconnected or weakly-connected (s,t). Empirically, we observe that there are usually too many paths connecting a positively predicted (s,t) instead of no paths, even in the k-core. Therefore, an optional step to enhance pruning is to remove nodes with super-high degrees. As we discussed in Section 3.4.1, high-degree nodes are often generic and less informative. Removing them can be a complement to k-core to further reduce complexity and improve path quality.

3.4.3 Heterogeneous Path-Enforcing Mask Learning

The second module of PaGE-Link learns heterogeneous masks to find important path-forming edges. We perform mask learning to select edges from the k-core-pruned computation graph. For notation simplicity in this section, we use $\mathcal{G} = (\mathcal{V}, \mathrm{E})$ to denote the graph for mask learning to save superscripts and subscripts, and $\mathcal{G}_c^k$ is the actual graph in the complete version of our algorithm.

The idea is to learn a mask over all edges of all edge types to select the important edges. Let $\mathrm{E}^r = \{e \in \mathrm{E} | \tau(e) = r\}$ be edges with type $r \in \mathcal{R}$. Let $\mathcal{M} = \{\mathcal{M}^r\}_{r=1}^{|\mathcal{R}|}$ be learnable

masks of all edge types, with $\mathcal{M}^r \in \mathbb{R}^{|\mathrm{E}^r|}$ corresponds type r. We denote applying $\mathcal{M}^r$ on its corresponding edge type by $\mathrm{E}^r \odot \sigma(\mathcal{M}^r)$, where σ is the sigmoid function, and $\odot$ is the element-wise product. Similarly, we also overload the notation $\odot$ to indicate applying the set of masks on all types of edges, i.e., $\mathrm{E} \odot \sigma(\mathcal{M}) = \cup_{r \in \mathcal{R}}\{\mathrm{E}^r \odot \sigma(\mathcal{M}^r)\}$. We call the graph with the edge set $\mathrm{E} \odot \sigma(\mathcal{M})$ a *masked graph*. Applying a mask on graph edges will change the edge weights, which makes GNNs pass more information between nodes connected by highly-weighted edges and less on others. The general idea of mask learning is to learn an $\mathcal{M}$ that produces high weights for important edges and low weights for others. To learn an $\mathcal{M}$ that better fits the LP explanation, we measure edge importance from two perspectives: important edges should be both influential for the model prediction and form meaningful paths. Below, we introduce two loss terms $\mathcal{L}_{pred}$ and $\mathcal{L}_{path}$ for achieving these two measurements.

$\mathcal{L}_{pred}$ is to learn to select influential edges for model prediction. The idea is to do a perturbation-based explanation, where parts of the input are considered important if perturbing them changes the model prediction significantly. In the graph sense, if removing an edge e significantly influences the prediction, then e is a critical counterfactual edge that should be part of the explanation. This idea can be formalized as maximizing the mutual information between the masked graph and the original graph prediction Y, which is equivalent to minimizing the prediction loss

$$\mathcal{L}_{pred}(\mathcal{M}) = -\log P_\Phi(Y = 1 | \mathcal{G} = (\mathcal{V}, \mathrm{E} \odot \sigma(\mathcal{M})), (s, t)). \tag{3.1}$$

$\mathcal{L}_{pred}(\mathcal{M})$ has a straightforward meaning, which says the masked subgraph should provide enough information for predicting the missing link (s, t) as the whole graph. Since the original prediction is a constant, $\mathcal{L}_{pred}(\mathcal{M})$ can also be interpreted as the performance drop after the mask is applied to the graph. A well-masked graph should give a minimum performance drop. Regularizations of the mask entropy and mask norm are often included in $\mathcal{L}_{pred}(\mathcal{M})$ to encourage the mask to be discrete and sparse.

$\mathcal{L}_{path}$ is the loss term for $\mathcal{M}$ to learn to select path-forming edges. The idea is to first identify a set of candidate edges denoted by E_{path} (specified below), where these edges can form concise and informative paths, and then optimize $\mathcal{L}_{path}(\mathcal{M})$ to enforce the mask weights for $e \in \mathrm{E}_{path}$ to increase and mask weights for $e \notin \mathrm{E}_{path}$ to decrease. We considered a weighted average of these two forces balanced by hyperparameters α and β,

$$\mathcal{L}_{path}(\mathcal{M}) = -\sum_{r \in \mathcal{R}} (\alpha \sum_{\substack{e \in \mathrm{E}_{path} \\ \tau(e)=r}} \mathcal{M}_e^r - \beta \sum_{\substack{e \in \mathrm{E}, e \notin \mathrm{E}_{path} \\ \tau(e)=r}} \mathcal{M}_e^r). \tag{3.2}$$

The key question for computing $\mathcal{L}_{path}(\mathcal{M})$ is to find a good E_{path} containing edges of concise and informative paths. As in Section 3.4.1, paths with these two desired properties should be short and without high-degree generic nodes. We thus define a score function of a path p reflecting these two properties as below

$$Score(p) = \log \prod_{\substack{e \in p \\ e=(u,v)}} \frac{P(e)}{D_v} = \sum_{\substack{e \in p \\ e=(u,v)}} Score(e), \tag{3.3}$$

$$Score(e) = \log \sigma(\mathcal{M}_e^{\tau(e)}) - \log(D_v). \tag{3.4}$$

In this score function, $\mathcal{M}$ gives the probability of e to be included in the explanation, i.e., $P(e) = \sigma(\mathcal{M}_e^{\tau(e)})$. To get the importance of a path, we first use a mean-field approximation for the joint probability by multiplying $P(e)$ together, and we normalize each $P(e)$ for edge $e = (u, v)$ by its target node degree D_v. Then, we perform log transformation, which improves numerical stability for multiplying many edges with small $P(e)$ or large D_v and break down a path score to a summation of edge scores $Score(e)$ that are easier to work with. This path score function captures both desired properties mentioned above. A path score will be high if the edges on it have high probabilities and these edges are linked to nodes with low degrees. Finding paths with the highest $Score(p)$ can be implemented using Dijkstra's shortest path algorithm [Dij59], where the distance represented by each edge is set to be the negative score

Algorithm 4 PaGE-Link

Input: heterogeneous graph $\mathcal{G}$, trained GNN-based LP model $\Phi(\cdot, \cdot)$, predicted link (s, t), size budget B, k for k-core, hyperparameters α and β, learning rate η, maximum iterations T.
Output: Explanation as a set of paths P.
Extract the computation graph $\mathcal{G}_c$;
Prune $\mathcal{G}_c$ for the k-core $\mathcal{G}_c^k$;
Initialize $\mathcal{M}^{(0)}$;
$t = 0$;
while $\mathcal{M}^{(t)}$ not converge and $t < T$ **do**
  Compute $\mathcal{L}_{pred}(\mathcal{M}^{(t)})$; { Eq.(3.1)}
  Compute $Score(e)$ for each edge e; { Eq.(3.4)}
  Construct E_{path} by finding shortest paths on $\mathcal{G}_c^k$ with edge distance $-Score(e)$;
  Compute $\mathcal{L}_{path}(\mathcal{M}^{(t)})$ according to E_{path}; { Eq.(3.2)}
  $\mathcal{M}^{(t+1)} = \mathcal{M}^{(t)} - \eta\nabla(\mathcal{L}_{pred}(\mathcal{M}^{(t)}) + \mathcal{L}_{path}(\mathcal{M}^{(t)}))$;
  t += 1;
end while
P = Under budget B, the top shortest paths on $\mathcal{G}_c^k$ with edge distance $-Score(e)$;
Return: P.

of the edge, i.e., $-Score(e)$. We let E_{path} be the set of edges in the top five shortest paths found by Dijkstra's algorithm.

3.4.4 Mask Optimization and Path Generation

We optimize $\mathcal{M}$ with both $\mathcal{L}_{pred}$ and $\mathcal{L}_{path}$. $\mathcal{L}_{pred}$ will increase the weights of the prediction-influential edges. $\mathcal{L}_{path}$ will further increase the weights of the path-forming edges that are also highly weighted by the current $\mathcal{M}$ and decrease other weights. Finally, after the mask learning converges, we run one more shortest-path algorithm to generate paths from the final $\mathcal{M}$ and select the top paths according to budget B to get the explanation P defined in Section 3.4.1. A pseudo-code of PaGE-Link is shown in Algorithm 4.

Table 3.3: Time complexity of PaGE-Link and baseline methods.

GNNExp [YBY19]	PGExp [LCX20]	SubgraphX [YYW21]	PaGE-Link (ours)
$O(\|\mathrm{E}_c\|T)$	$O(\|\mathrm{E}\|T)$ / $O(\|\mathrm{E}_c\|)$	$\Theta(\|\mathcal{V}_c\|\hat{D}^{2B_{node}-2})$	$O(\|\mathrm{E}_c\| + \|\mathrm{E}_c^k\|T)$

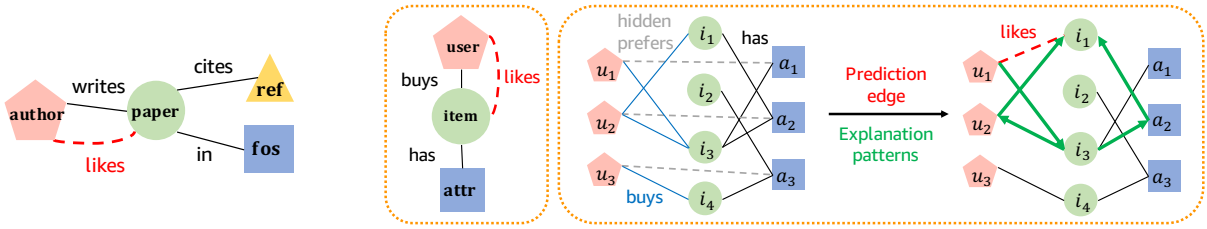
3.4.5 Complexity Analysis

In Table 3.3, we summarize the time complexity of PaGE-Link and representative existing methods for explaining a prediction with computation graph $\mathcal{G}_c = (\mathcal{V}_c, \mathrm{E}_c)$ on a full graph $\mathcal{G} = (\mathcal{V}, \mathrm{E})$. Let T be the mask learning epochs. GNNExplainer has complexity $|\mathrm{E}_c|T$ as it learns a mask on E_c. PGExplainer has a training stage and an inference stage (separated by / in the table). The inference stage is linear in $|\mathrm{E}_c|$, but the training stage covers edges in the entire graph and thus scales in $O(|\mathrm{E}|T)$. SubgraphX has a much higher time complexity exponential in $|\mathcal{V}_c|$, so a size budget of B_{node} nodes is forced to replace $|\mathcal{V}_c|$, and $\hat{D} = \max_{v\in\mathcal{V}} D_v$ denotes the maximum degree (derivation in next paragraph). For PaGE-Link, the k-core pruning step is linear in $|\mathrm{E}_c|$. The mask learning with Dijkstra's algorithm has complexity $|\mathrm{E}_c^k|T$. PaGE-Link has a better complexity than existing methods since $|\mathrm{E}_c^k|$ is usually smaller than $|\mathrm{E}_c|$ (see Theorem 3.4.3), and PaGE-Link often converges faster, i.e., has a smaller T, as the space of candidate explanations is smaller (see Proposition 3.4.1) and noisy nodes are pruned.

The search-based methods often have much higher time complexity exponential in the number of nodes or edges. Thus, a budget is forced instead of searching subgraphs with all sizes. For example, SubgraphX finds all connected subgraphs with at most B_{node} nodes, which has complexity $\Theta(|\mathcal{V}_c|\hat{D}^{2B_{node}-2})$ for a graph with maximum degree $\hat{D} = \max_{v\in\mathcal{V}} D_v$. This complexity can be shown using the following two lemmas.

Lemma 3.4.4. *For a graph $\mathcal{G}$ with n vertices, the number of the connected subgraph of $\mathcal{G}$ having B_{node} nodes is bounded below by the number of trees in $\mathcal{G}$ having B_{node} nodes.*

Proof. Each connected subgraph has a spanning tree. □



(a) Schema of AugCitation. "writes", "cites", "has" and "in" edges are original. The "likes" edges (dashed red) are augmented for prediction.

(b) Schema of UserItemAttr (the left box) and its generation process (the right box). Three types of base edges are generated first, i.e., "has" (black), "hidden prefers" (dashed gray), and "buys" (blue). The solid "has" and "buys" edges are then used to generate "likes" edges (dashed red) for prediction and the ground truth explanation patterns (green arrows).

Figure 3.4: The proposed augmented graph AugCitation and the synthetic graph UserItem-Attr.

Lemma 3.4.5. *For a graph $\mathcal{G}$ with node set $\mathcal{V}$, the number of trees in $\mathcal{G}$ having B_{node} tree nodes is $\Theta(|\mathcal{V}|\hat{D}^{2B_{node}-2})$.*

Proof. See [fil18] for proof using an encoding procedure. □

3.5 Evaluation

In this section, we conduct empirical studies to evaluate explanations generated by PaGE-Link. Evaluation is a general challenge when studying model explainability since standard datasets do not have ground truth explanations. Many works [YBY19, LCX20] use synthetic benchmarks, but no benchmarks are available for evaluating GNN explanations for heterogeneous LP. Therefore, we generate an augmented graph and a synthetic graph to evaluate explanations. They allow us to generate ground truth explanation patterns and evaluate explainers quantitatively.

3.5.1 Datasets

The augmented graph AugCitation is constructed by augmenting the AMiner citation network [TZY08]. A graph schema is shown in Figure 3.4a. The original AMiner graph contains four node types: author, paper, reference (ref), and field of study (fos), and edge types "cites", "writes", and "in". We construct AugCitation by augmenting the original graph with new (author, paper) edges typed "likes" and define a paper recommendation task on AugCitation for predicting the "like" edges. A new edge (s, t) is augmented if there is at least one concise and informative path p between them. In our augmentation process, we require the paths p to have lengths shorter than a hyperparameter l_{max} and with degrees of nodes on p (excluding s & t) bounded by a hyperparameter D_{max}. We highlight these two hyperparameters because of the conciseness and informativeness principles discussed in Section 3.4.1. The augmented edge (s, t) is used for prediction. The ground truth explanation is the set of paths satisfying the two hyperparameter requirements. We only take the top P_{max} paths with the smallest degree sums if there are many qualified paths. We train a GNN-based LP model to predict these new "likes" edges and evaluate explainers by comparing their output explanations with these path patterns as ground truth.

The synthetic graph UserItemAttr is generated to mimic graphs with users, items, and attributes for recommendations. Figure 3.4b shows the graph schema and illustrates the generation process. We include three node types: "user", "item", and item attributes ("attr") in the synthetic graph, and we build different types of edges step by step. Firstly, the "has" edges are created by randomly connecting items to attrs, and the "hidden prefers" edges are created by randomly connecting users to attrs. These edges represent items having attributes and user preferences for these attributes. Next, we randomly sample a set of items for each user, and we connect a (user, item) pair by a "buys" edge, if the user "hidden prefers" any attr the item "has". The "hidden prefers" edges correspond to an intermediate step for generating the observable "buys" edges. We remove the "hidden prefers" edges after "buys" edges are

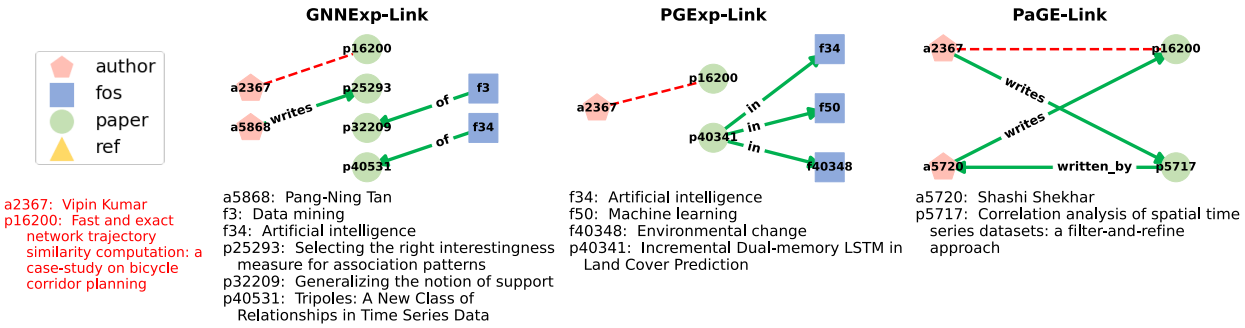


Figure 3.5: **Explanations visualization and comparison between** PaGE-Link **and baselines.** Explanations (green arrows) by different explainers for the predicted link $(a2367, p16200)$ (dashed red). The explanation generated by PaGE-Link explains the recommendation by co-authorship, whereas baseline explanations are less interpretable.

generated since we cannot observe 'hidden prefers" information in reality. An example of the rationale behind this generation step is that items have certain attributes, like the item "ice cream" with the attribute "vanilla". Then given that a user likes the attribute "vanilla" as hidden information, we observe that the user buys "vanilla ice cream". The next step is to generate more 'buys" edges between randomly picked (user, item) pairs if a similar user (two users with many shared item neighbors) buys this item. The idea is like collaborative filtering, which says similar users tend to buy similar items. The final step is generating edges for prediction and their corresponding ground truth explanations, which follows the same augmentation process described above for AugCitation. For UserItemAttr, we have "has" and "buys" as base edges to construct the ground truth, and we create "likes" edges between users and items for prediction.

We show the hyperparameters for constructing the datasets in Table 3.4.

Table 3.4: Hyperparameters for constructing AugCitation and UserItemAttr

	l_{max}	D_{max}	P_{max}
AugCitation	3	30	5
UserItemAttr	3	15	5

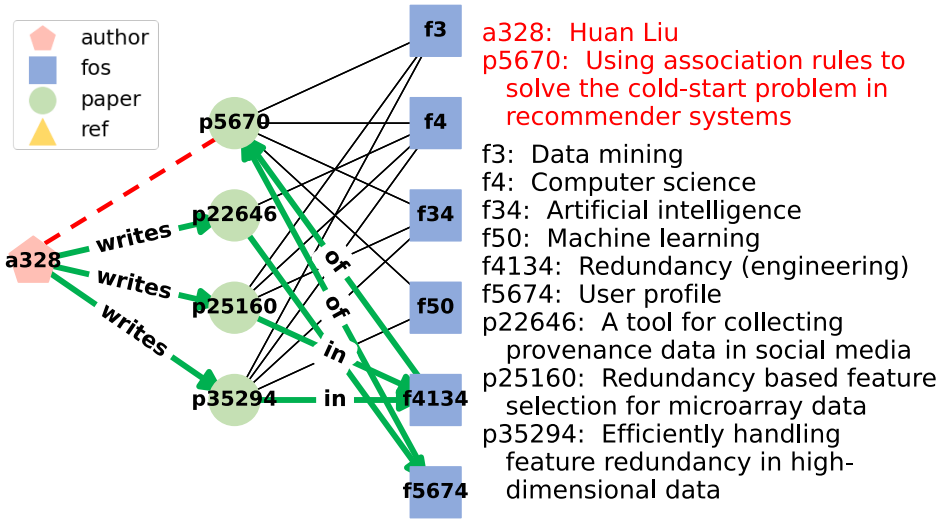


Figure 3.6: **Top paths selected by** PaGE-Link**.** Top three paths (green arrows) for explaining the predicted link $(a328, p5670)$ (dashed red). The selected paths are short and do not go through a generic field of study like "Computer Science".

3.5.2 Experiment Settings

The GNN-based LP model As described in Section 3.3, the LP model involves a GNN encoder and a prediction head. We use RGCN [SKB18] as the encoder to learn node representations on heterogeneous graphs and the inner product as the prediction head. We train the model using the cross-entropy loss. On each dataset, our prediction task covers one edge type r. We randomly split the observed edges of type r into train:validation:test = 7:1:2 as positive samples and draw negative samples from the unobserved edges of type r. Edges of other types are used for GNN message passing but not prediction.

Explainer baselines. Existing GNN explanation methods cannot be directly applied to heterogeneous LP. Thus, we extend the popular GNNExplainer [YBY19] and PGExplainer [LCX20] as our baselines. We re-implement a heterogeneous version of their mask matrix and mask predictor similar to the heterogeneous mask learning module in PaGE-Link. For these baselines, we perform mask learning using their original objectives, and we generate

Table 3.5: **Performance comparison in ROC-AUC scores of learned masks.** PaGE-Link outperforms baselines.

	GNNExp-Link	PGExp-Link	PaGE-Link (ours)
AugCitation	0.829	0.586	**0.928**
UserItemAttr	0.608	0.578	**0.954**

edge-induced subgraph explanations from their learned mask. We refer to these two adapted explainers as GNNExp-Link and PGExp-Link below. We do not compare to other search-based explainers like SubgraphX [YYW21] because of their high computational complexity (see Section 3.4.5). They work well on small graphs as in the original papers, but they are hard to scale to large and dense graphs we consider for LP.

3.5.3 Algorithmic Evaluation

Quantitative evaluation. Both the ground truth and the final explanation output of PaGE-Link are sets of paths. In contrast, the baseline explainers generate edge masks $\mathcal{M}$. For a fair comparison, we take the intermediate result PaGE-Link learned, also the mask $\mathcal{M}$, and we follow [LCX20] to compare explainers by their masks. Specifically for each computation graph, edges in the ground truth paths are treated as positive, and other edges are treated as negative. Then weights in $\mathcal{M}$ are treated as the prediction scores of edges and are evaluated with the ROC-AUC metric. A high ROC-AUC score reflects that edges in ground truth are precisely captured by the mask. The results are shown in Table 3.5, where PaGE-Link outperforms both baseline explainers.

For scalability, we showed PaGE-Link scales linearly in $O(|\mathrm{E}_c^k|)$ in Section 3.4.5. Here we evaluate its scalability empirically by generating ten synthetic graphs with various sizes from 20 to 5,500 edges in $\mathcal{G}_c$. The results are shown in Figure 3.2b, which suggests the computation time scales linearly in the number of edges.

Besides ROC-AUC scores, another way to evaluate the explanations is through the path

Table 3.6: **Performance comparison in path hit rates (HR) between** PaGE-Link **and baselines.** GStarXhas high HR with a small budget B. Baselines achieve nonzero HR for large B.

	B	GNNExp-Link	PGExp-Link	GStarX(ours)
AugCitation	10	0.000	0.000	**0.007**
	50	0.002	0.000	**0.194**
	100	0.019	0.000	**0.425**
	200	0.064	0.002	**0.645**
UserItemAttr	10	0.000	0.000	**0.163**
	50	0.008	0.032	**0.705**
	100	0.016	0.039	**0.790**
	200	0.046	0.101	**0.907**

hit rate (HR). Specifically, we fix the budget of B edges and evaluate whether an explanation can hit any complete path in the ground truth. Note that the ground truth for each link (s, t) only has the top P_{max} shortest paths with the smallest degree sums, so hitting a long path or a less informative path with high-degree generic nodes will not count.

For a fair comparison with baselines, we take the generated explanation mask $\mathcal{M}$ for each method, select the top B weighted edges to compare against the ground truth. We show results with different budget B in Table 3.6. Explanations generated by GStarXhave higher path HR than baselines on both datasets. In contrast, GNNExp-Link and PGExp-Link can barely hit any path in the ground truth for B less than 50.

Note that the actual explanation output of GStarXis a set of paths P. If we evaluate P instead of the top cut of the intermediate output mask $\mathcal{M}$. Then GStarXcan achieve perfect path HR (=1) when the budget |P| gets large.

Qualitative evaluation. A critical advantage of PaGE-Link is that it generates path explanations, which can capture the connections between node pairs and enjoy better interpretability. In contrast, the top important edges found by baseline methods are often disconnected from the source, the target, or both, which makes their explanations hard

for humans to interpret and investigate. We conduct case studies to visualize explanations generated by PaGE-Link on the paper recommendation task on AugCitation.

Figure 3.5 shows a case in which the model recommends the source author "Vipin Kumar" the recommended target paper titled "Fast and exact network trajectory similarity computation: a case-study on bicycle corridor planning". The top path explanation generated by PaGE-Link goes through the coauthor "Shashi Shekhar", which explains the recommendation as Vipin Kumar and Shashi Shekhar coauthored the paper "Correlation analysis of spatial time series datasets: a filter-and-refine approach", and Shashi Shekhar wrote the recommended paper. Given the same budget of three edges, explanations generated by baselines are less interpretable.

Figure 3.6 shows another example with the source author "Huan Liu" and the recommended target paper titled "Using association rules to solve the cold-start problem in recommender systems". PaGE-Link generates paths going through the common fos of the recommended paper and three other papers written by Huan Liu: $p22646$, $p25160$, and $p35294$. We show the PaGE-Link explanation with the top three paths in green. We also show other unselected fos shared by the $p22646$, $p25160$, and $p35294$ and the target paper. Note that the explanation paths all have length three, even though there are many paths with length five or longer, e.g., $(a328, p22646, f4, p25260, f4134, p5670)$. Also, the explanation paths go through the fos "Redundancy (engineering)" and "User profile" instead of generic fos like "Artificial intelligence" and "Computer science". This case demonstrates that explanation paths selected by PaGE-Link are more concise and informative.

3.5.4 Human Evaluation

The ultimate goal of model explanation is to improve model transparency and help human decision-making. Human evaluation is thus the best way to evaluate the effectiveness of an explainer, which has been a standard evaluation approach in previous works [SCD17, RSG16, GBS20]. We conduct a human evaluation by randomly picking 100 predicted links from

the test set of AugCitation and generate explanations for each link using GNNExp-Link, PGExp-Link, and PaGE-Link. We design a survey with single-choice questions. In each question, we show respondents the predicted link and those three explanations with both the graph structure and the node/edge type information, similarly as in Figure 3.5 but excluding method names. The survey is sent to people across graduate students, postdocs, engineers, research scientists, and professors, including people with and without background knowledge about GNNs. We ask respondents to "please select the best explanation of *'why the model predicts this author will like the recommended paper?'* ". At least three answers from different people are collected for each question. In total, 340 evaluations are collected and 78.79% of them selected explanations by PaGE-Link as the best.

3.6 Discussion

In this work, we study model transparency and accountability on graphs. We investigate a new task: GNN explanation for heterogeneous LP. We identify three challenges for the task and propose a new path-based method, i.e. PaGE-Link, that produces explanations with *interpretable connections*, is *scalable*, and handles graph *heterogeneity*. PaGE-Link explanations quantitatively improve ROC-AUC by 9 - 35% over baselines and are chosen by 78.79% responses as qualitatively more interpretable in human evaluation.

Part III

Explanations Extract Data Insights

CHAPTER 4

Predicting and Interpreting Energy Barriers of Metallic Glasses with GNNs

4.1 Introduction

Metallic glasses (MGs) combine good properties of metals and plastics in one material, making them stronger than steel while being shapeable as plastic [SHK11]. Their extensive applications span various industries including aerospace, sports equipment, luxury goods, biomedical devices, and many more [TT10]. The unique properties of MGs lie in their non-crystalline amorphous atomic structure, which sets them apart from the crystalline structure found in traditional metals [TT10, BD13]. Despite extensive research on MGs, the details of their structure-property relationship are still not well understood [SSD02, DPF14, PVF16, CLK18].

One promising approach for studying the structure-property relationship of MGs is through a special property called *Energy Barrier (EB)*. EBs describe the local roughness of the energy landscape by comparing the average energy difference around an atom's local neighbors. Many studies have shown that understanding EBs can act as an important intermediary step for studying the MG physical properties [DS01, YSW12, TLM21]. As shown in Figure 4.1, EBs represent mobility, which can influence the MG dynamics and further their physical properties like glass transition and ductility [BB11, KCZ22]. However, the precise simulation of EBs is challenging and often requires time-consuming computation [BM96, MBB12, JGS22]. For example, even with a high-performance computing (HPC) cluster and the advanced

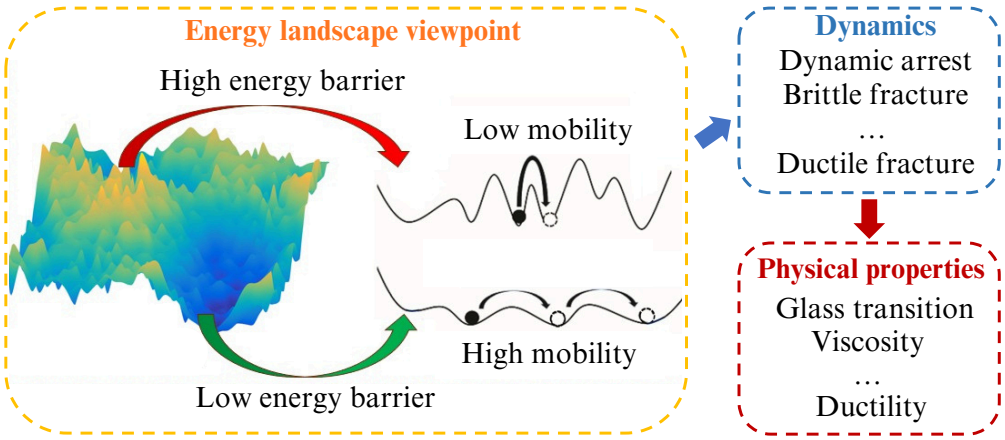


Figure 4.1: **EBs, mobility, and MG physical properties.** EBs represent mobility, which can further influence the MG dynamics and their physical properties.

Activation-Relaxation Technique nouveau (ARTn) [CLM09], calculating EBs for an MG system with 3,000 atoms can take 41 days.

Given the usefulness and computational difficulty of EBs, we explore machine learning (ML) approaches to efficiently predict them from MG atomic structures. Similar to recent ML investigations on glassy systems [BKG20, RNE22], we phrase the EB prediction problem as a graph ML problem and solve it using GNNs. Under this formalization, atoms become nodes in a graph, and edges are constructed between nearby nodes to represent the atomic structure. Atom types are used as node features. Displacement vectors constructed from 3D node coordinates are used as edge features. Then EB prediction becomes a node regression task on graphs.

We simulate MG systems and employ ARTn to calculate some EBs as training labels. Given the challenge of collecting data, a more data-efficient model with a stronger inductive bias is desired. In particular, the EB prediction problem exhibits E(3)-invariance, i.e., invariance to graph structure transformations including translations, rotations, reflections, and their combinations. We aim for a GNN that can handle such invariance, but general message-passing-based GNNs like GCN [KW17] cannot. Some specially designed models are

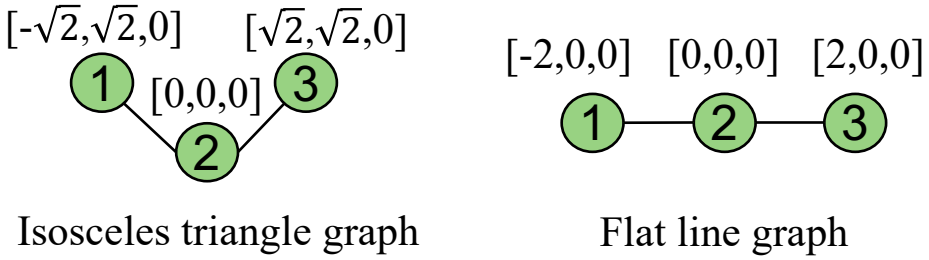


Figure 4.2: **Example graphs demonstrating model expressiveness.** SchNet cannot distinguish the embeddings of node 1 in these two graphs but SymGNN can.

E(3)-invariant [SKS17, LLW19, GGG20, TD22, LS22, BKS22, BMS22], but, to the best of our knowledge, none of the existing methods can achieve *invariance*, *expressiveness*, and *scalability* at the same time as we show in Table 4.1.

To achieve an invariant model that is both expressive and scalable, we propose a simple but effective *Symmetrized GNN (SymGNN)*, which is E(3)-invariant in expectation. SymGNN achieves E(3) invariance by introducing a symmetrization module to aggregate embeddings produced under different orthogonal transformations of the graph structure. It is expressive as there is no higher-order information loss caused by "feature scalarization" as in models like SchNet [SKS17]. For example, for the two graphs in Figure 4.2, SchNet will pass the same message to the central node of an isosceles triangle as if the middle node of a flat line, but SymGNN will easily distinguish between these two configurations (details in Section 4.4.4). Also, SymGNN does not involve complex equivariant calculation, so it is much more scalable than methods relying on equivariant feature extraction [LS22, BKS22, BMS22]. In our experiments, we demonstrate that when applied to MG graphs, SymGNN outperforms a variety of widely used GNNs.

Moreover, to better understand the EB prediction and benefit MG research, we also generate explanations along with the model prediction. Our proposed explanation method extends GNNExplainer [YBY19] to the node regression task to generate edge-based structure explanations. It helps us to identify and visualize the importance of each edge in predicting

Table 4.1: **Characteristic comparison of different methods.** ✓ means the model performs well evaluated by the corresponding category, ✗ means not, and ? means "unclear", or "possible after non-trivial extensions". The E indicates that the model satisfies the property in expectation.

Methods	Invariance	Expressiveness	Scalability
GCN	✗	✓	✓
EGNN	?	✓	✓
SchNet	✓	✗	✓
MGCN	✓	✗	✓
FAENet	✓	✗	✓
DimeNet	✓	✓	✗
Torch-MD Net	✓	✓	✗
Equiformer	✓	✓	✗
SymGNN (ours)	E	✓	✓

an EB. We also show that the generated explanations match the medium-range order (MRO) hypothesis of MGs and possess unique topological properties that correlate with the optimal-volume cycles in a persistent diagram [Oba18]. Our findings provide further insights into the understanding of EBs, and our explanations can potentially benefit new scientific discoveries. We summarize our contributions as the following:

1. We formulate a material science research problem of predicting MG EBs as an ML problem of node regression on graphs.
2. We collect MG data for ML research, with precisely simulated EBs using ARTn.
3. We propose a simple but effective SymGNN model that exhibits E(3)-invariance in expectation and predicts MG EBs accurately and fast.
4. We generate explanations for EB predictions that match the MRO hypothesis, express unique topological properties, and provide insights for scientific discoveries.

4.2 Related Work

ML in Material Science. The application of ML in materials science has seen significant advancements recently, with various models to tackle different aspects of material science problems. Among these, GNNs have emerged as a powerful tool for representing and analyzing materials at the atomic level, owing to their ability to capture the complex relationships and interactions between atoms in a material. For example, estimating the propensity of individual atoms [BKG20], potential energy exhibited by a system of atoms [SKS17]. In these settings, inductive bias of equivariance and invariance often plays a key role in the generalizability of network. For example, in our problem the EB only depends on local molecule configuration and thus are invariant on translation, rotation, and reflection of graphs. To incorporate this physical inductive bias, various invariant and equivariant GNNs have been proposed. Invariant GNNs often restrict graph features to be rotationally invariant, such as edge distances and angles, or reducing the inputs by projecting it onto PCA frames [SKS17, GGG20, GGM20, DSH23], whereas equivariant networks are proposed to leverage tensorial transformation that can extract equivariant node features [SUG21, LS22, BKS22, BMS22, TD22]. Several evaluation benchmarks for equivariant ML models on molecular dynamics [BMP24] and solid-state materials systems [CGR20, LGN23] have been proposed. EGRaffBench [BMP24] provides insights into the performance of various equivariant architectures in predicting forces in molecular systems, highlighting their potentials in simulating atomistic interactions. Moreover, JARVIS [CGR20] and the MatSciML benchmark [LGN23] represent significant efforts in benchmarking MLs to solid-state materials systems. These works demonstrate the potential of ML models, including various GNNs, in predicting properties and behaviors of solid-state materials, thereby aiding in materials design and discovery. Furthermore, a comprehensive overview of geometric GNNs for 3D atomic systems is provided by Duval et al. in their recent review [DMJ23]. This guide offers valuable insights into the development and application of GNNs in materials science, emphasizing the importance of geometric considerations in modeling atomic systems.

MGs and EBs. Understanding the relationship between the atomic structure and physical properties of MGs is one of the greatest challenges for both material science and condensed matter physics [FL11, SW15, NFM18]. However, the structure-property relationship of MGs is often challenging to characterize directly due to the complexity of the physical properties [CIS17, BKG20]. EBs describe the local roughness of the energy landscape by comparing the average energy difference around an atom's local neighbors. They are influential in MG dynamics and their physical properties [BB11, KCZ22], for example, the degree of ductility during fracture [TLM21]. Therefore, EBs can act as an important intermediary step when predicting the physical properties with the atomic structures as inputs [DS01, YSW12, WDM20, TLM21]. ML methods have been applied to investigate the relationship between the atomic structures and physical properties in MG [BKG20]. For EBs in particular, [WDM20] explored using XGBoost to classify nodes with the highest 5 percent activation energy. Our work furthers the investigation of [WDM20] by leveraging the natural graph structure using GNNs to perform a regression for EBs and generating insightful explanations.

4.2.1 GNN Explanation

Model explainability is crucial for complex modern ML models. Specifically for explaining GNNs, a wide range of explanation methods are proposed to select the most influential edges, nodes, features, and even subgraphs [YBY19, LCX20, YYW21, ZLS22, LLW22], for the prediction of one node in the sense that the most message passing happens. GNNExplainer is the pioneering work that achieves this goal by learning edge masks to maximize the mutual information between perturbed output and the original model output [YBY19]. These works including GNNExplainer focus on explaining classification problems, whereas we focus on explaining node regression. To the best of our knowledge, the only work that targets GNN explanation for regression tasks is [ZCM23], but it is very different from ours because it focuses on graph-level regression, and it does not consider any invariance explanation nor

any application to material science.

4.3 Problem Setup and Preliminaries

4.3.1 EB Prediction with GNNs

The problem of predicting EBs of MGs can be formalized as a node regression problem on graphs. Under this formulation, atoms become nodes in a graph, and edges are constructed between nearby nodes. The MG data thus becomes a graph with n nodes and m edges. We represent the graph structure with G, which indicates all the edges and is normally represented in the form of an adjacency matrix. The node features are the atom types, which we represent with $\boldsymbol{Z} = \{\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^n\}$. The edge features are the displacement vectors constructed from 3D node coordinates, which we represent with $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^m \mid \boldsymbol{x}^i \in \mathcal{R}^3\}$. The regression task is to predict the EB label $y \in \mathcal{R}$ of each node with the graph structure and features as inputs, i.e., a model that maximizes $P(y|\, G, \boldsymbol{Z}, \boldsymbol{X})$. We further break down the prediction process into two steps. The first step encodes node and edge features to embeddings $\boldsymbol{H}$. The second step predicts y with G and $\boldsymbol{H}$ as inputs. The objective to maximize becomes the following,

$$P(y|\, G, \boldsymbol{Z}, \boldsymbol{X}) = \int_{\boldsymbol{H}} P(y|\, G, \boldsymbol{H}) P(\boldsymbol{H}|\, G, \boldsymbol{Z}, \boldsymbol{X}) d\boldsymbol{H} \tag{4.1}$$

We solve this problem with the state-of-the-art graph ML models - GNNs.

4.3.2 Orthogonality and Invariance

EB is invariant to Euclidean transformations of the atomic graph structure, for example, rotations, reflections, and translations, because it is the average energy needed for a node to hop between its current and nearby energy subbasins. Given that the graph is described with displacement vectors of relative positions, translations will be canceled, and

the invariance to Euclidean transformations can be reduced to the invariance to orthogonal transformations [HH13], which is defined as the following,

Definition 4.3.1 (Orthogonal Transformation)**.** A linear transformation $T : \mathbb{R}^d \to \mathbb{R}^d$ is called an *orthogonal transformation* if it preserves the inner produce $\langle \cdot, \cdot \rangle$ on $\mathcal{R}^d$, i.e., $\forall \boldsymbol{x}^1, \boldsymbol{x}^2 \in \mathbb{R}^d$, $\langle T(\boldsymbol{x}^1), T(\boldsymbol{x}^2) \rangle = \langle \boldsymbol{x}^1, \boldsymbol{x}^2 \rangle$. Then, the matrix form of T has $|\det(T)| = 1$. The *orthogonal group* in dimension d is the group of all such orthogonal transformations on $\mathbb{R}^d$ and is denoted as $\mathrm{O}(d)$.

We also state a well-known lemma in group theory [HH13] and a theorem by Euler [Sla99] for decomposing the orthogonal group and rotations respectively. They will be useful for modeling invariance.

Lemma 4.3.2 (O(3) Decomposition)**.** *The orthogonal group* $\mathrm{O}(3)$ *can be decomposed into rotations and non-rotations. The rotations also form a group denoted as* $\mathrm{SO}(3)$*, and it contains all transformations* R *whose matrix forms have* $\det(R) = 1$*. The non-rotations contain all the reflections and roto-reflections (also called improper rotation)* $\tilde{R}$*, whose matrix form have* $\det(\tilde{R}) = -1$*. Non-rotations can be denoted as* $P \cdot \mathrm{SO}(3)$*, with* P *being any reflection transformation through the origin.*

Theorem 4.3.3. *(Euler) Define the rotations around the three coordinate axes* x_1, x_2*, and* x_3 *in* $\mathbb{R}^3$ *by*

$$O_{x_1}(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

$$O_{x_2}(\beta) = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

$$O_{x_3}(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Then any rotation $R \in \mathrm{SO}(3)$ *can be written as* $R_{\alpha,\beta,\gamma} = O_{x_1}(\alpha)O_{x_2}(\beta)O_{x_3}(\gamma)$ *for some angles* $[\alpha, \beta, \gamma] \in [-\pi, \pi]^3$. *These angles are called the Euler angles.*

Then we formally introduce the invariant/equivariant transformation.

Definition 4.3.4 (Invariant/Equivariant Transformation)**.** Given a group K acts on $\mathcal{R}^d$. A transformation $T : \mathcal{R}^d \to \mathcal{R}^d$ is invariant to K if $T(\boldsymbol{x}) = T(k \cdot \boldsymbol{x})$ and equivariant to K if $k \cdot T(\boldsymbol{x}) = T(k \cdot \boldsymbol{x})$ for all $k \in K$ and for all $\boldsymbol{x} \in \mathcal{R}^d$.

4.3.3 GNNExplainer

As a representative GNN explanation method, GNNExplainer seeks to explain GNN classifications by selecting an important edge-induced subgraph G_S that minimizes the entropy $H(\cdot)$ of the label Y. Since G_S is discrete, GNNExplainer learns a continuous distribution $\mathcal{G}$ over G_S that gives the minimal expected entropy, where $\mathcal{G}$ can be implemented as a learnable edge mask $\boldsymbol{M} \in \mathcal{R}^{|G|}$ applied on edges of G after a sigmoid function σ. Mathematically, the optimization objective is

$$\min_{\mathcal{G}} \mathbb{E}_{G_S \sim \mathcal{G}} H(Y|G = G_S) = \min_{\boldsymbol{M}} H(Y|G = \sigma(\boldsymbol{M}) \odot G) \tag{4.2}$$

4.4 SymGNN: Symmetrized GNNs

In this section, we present SymGNN for solving the EB prediction problem we formalized in Section 4.3.1. We first introduce the theory behind the core symmetrization module for capturing O(3) invariance in Section 4.4.1, then the full SymGNN model in Section 4.4.2, and finally how we apply explanation algorithms to SymGNN to reveal the connection between

the atomic structures and EBs in Section 4.4.5.

4.4.1 Theory of Symmetrization Over O(3)

Although EB is invariant to Euclidean transformations of the atomic graph structure, most GNNs are not designed to automatically capture such invariance. There are existing GNNs specialized for molecular graphs that can handle such invariance, but they either utilize scalarization that cannot handle higher-order information, or cannot scale up to graphs with thousands of nodes like MGs. We thus propose a symmetrization module that can better capture invariance and efficiently scale up. This section presents the theory behind the symmetrization.

For the node regression problem formalized in Section 4.3.1, $\boldsymbol{X}$ only represents one set of displacement vectors under one particular coordinate system. To achieve O(3)-invariant (and thus E(3)-invariant as explained in Section 4.3.2) predictions, we propose a symmetrization over all orthogonal transformations of $\boldsymbol{X}$, denoted as $\mathcal{X} = \{T(\boldsymbol{X}) \,|\, \forall T \in \mathrm{O}(3)\}$. Under symmetrization, we reformulate the feature encoding step in Equation 4.1, i.e.,$P(\boldsymbol{H} \,|\, G, \boldsymbol{Z}, \mathcal{X})$, as a probability integrated over $\mathcal{X}$, i.e.,

$$P(\boldsymbol{H} \,|\, G, \boldsymbol{Z}, \mathcal{X}) = \int_{T \in \mathrm{O}(3)} P(\boldsymbol{H} \,|\, G, \boldsymbol{Z}, T(\boldsymbol{X}))P(T)\, dT \tag{4.3}$$

Notice that a truly O(3)-invariant model will give the same result for $P(\boldsymbol{H} \,|\, G, \boldsymbol{Z}, \boldsymbol{X})$ and $P(\boldsymbol{H} \,|\, G, \boldsymbol{Z}, \mathcal{X})$. In the new formulation, when maximizing $P(\boldsymbol{H} \,|\, G, \boldsymbol{Z}, \mathcal{X})$, the model will learn the desired invariance by foreseeing and aggregating different transformed graphs. To model such an integral, we first define the distribution of T on O(3) through the following two lemmas.

Lemma 4.4.1. *Any non-rotation* $\tilde{R} \in P{\cdot}\mathrm{SO}(3)$ *can be written as* $\tilde{R}_{\alpha,\beta,\gamma} = -O_{x_1}(\alpha)O_{x_2}(\beta)O_{x_3}(\gamma)$ *for some parameters* $[\alpha, \beta, \gamma] \in [-\pi, \pi]^3$.

Proof. Please refer to Appendix A.3.1. □

Intuitively, Theorem 4.3.3 says that any rotation in 3D can be decomposed into a combination of rotations that rotate only around the x_1-axis, x_2-axis, and x_3-axis and parameterized with the Euler angels. Similarly, Lemma 4.4.1 says a similar decomposition and parameterization can be achieved for non-rotations as well. Bring these two results together gives the following lemma for decomposing any orthogonal transformation $T \in \mathrm{O}(3)$.

Lemma 4.4.2. *Any orthogonal transformation* $T \in \mathrm{O}(3)$ *can be written as* $T_{\lambda,\alpha,\beta,\gamma} = (-1)^{\lambda} O_{x_1}(\alpha) O_{x_2}(\beta) O_{x_3}(\gamma)$ *for some parameters* $\lambda \in \{0, 1\}$ *and* $[\alpha, \beta, \gamma] \in [-\pi, \pi]^3$.

Proof. Follow from Lemma 4.3.2, Theorem 4.3.3, and Lemma 4.4.1. □

Lemma 4.4.2 allows the integral in Equation 4.3 to be reduced into an integral over λ and $[\alpha, \beta, \gamma]$ in Equation 4.4, which is the objective our GNN will model.

$$P(\boldsymbol{H} \,|\, G, \boldsymbol{Z}, \mathcal{X}) = \int_{\lambda,\alpha,\beta,\gamma} P(\boldsymbol{H} \,|\, G, \boldsymbol{Z}, T_{\lambda,\alpha,\beta,\gamma}(\boldsymbol{X})) P(T_{\lambda,\alpha,\beta,\gamma})\, d\lambda d\alpha d\beta d\gamma$$

4.4.2 Symmetrized GNN

We now present the full SymGNN model with an illustration shown in Figure 4.3. SymGNN consists of two sub-modules. The first is the symmetrization module mentioned above for producing O(3)-invariant embeddings $\boldsymbol{H}$, which we indicate with $\boldsymbol{H} = Sym(G, \boldsymbol{Z}, \boldsymbol{X})$. The second is a prediction module that takes the symmetrized $\boldsymbol{H}$ and G to perform message passing with attention and then node regression.

The *Sym* module produces embeddings following the objective in Equation 4.4 with a learnable encoder *Enc*, i.e.,

$$\boldsymbol{H} = Sym(G, \boldsymbol{Z}, \boldsymbol{X}) = \int_{\lambda,\alpha,\beta,\gamma} Enc(G, \boldsymbol{Z}, T_{\lambda,\alpha,\beta,\gamma}(\boldsymbol{X})) P(T_{\lambda,\alpha,\beta,\gamma}) d\lambda d\alpha d\beta d\gamma \qquad (4.4)$$

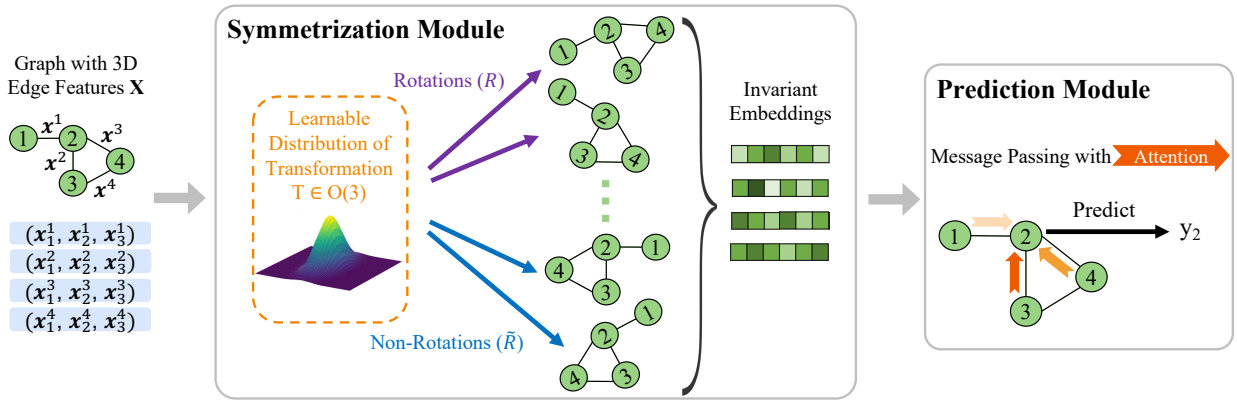


Figure 4.3: **Illustration of the SymGNN framework.** Given an input graph with node features being atom types and edge feature the relative distance, the symmetrization module of SymGNN aggregates encoding results on various orthogonally transformed graphs sampled from a learnable distribution to achieve O(3)-invariant in expectation. The invariant embeddings are then passed to message-passing layers with attention to aggregate information and predict label.

However, one challenge is that there are infinitely many $T \in \mathrm{O}(3)$, which makes the integral intractable. To model such an integral, we generate transformations $T_1, \ldots, T_k$ from $\mathrm{O}(3)$ by sampling λ and $[\alpha, \beta, \gamma]$ to approximate the Sym in Equation 4.4, which gives the $Sym_{T_1,\ldots,T_k}$ we use in practice.

$$Sym_{T_1,\ldots,T_k}(G, \boldsymbol{Z}, \boldsymbol{X}) = \frac{1}{k}\sum_{i=1}^{k} Enc(G, \boldsymbol{Z}, T_i(\boldsymbol{X})) \tag{4.5}$$

We show that $Sym_{T_1,\ldots,T_k}$ is $\mathrm{O}(3)$-invariant in expectation under assumptions of uniform distributions.

Theorem 4.4.3. *Assume $T_1, \ldots, T_k$ are random transformations that follow a uniform distribution over all $T \in \mathrm{O}(3)$. Then, Sym is $\mathrm{O}(3)$-invariant in expectation in the sense that $\mathbb{E}_{T_1,\ldots,T_k}[Sym_{T_1,\ldots,T_k}(G, \boldsymbol{Z}, \boldsymbol{X})] = \mathbb{E}_{T_1,\ldots,T_k}[Sym_{T_1,\ldots,T_k}(G, \boldsymbol{Z}, T_0\boldsymbol{X})]$ for any $T_0 \in \mathrm{O}(3)$.*

Proof. Please refer to Appendix A.3.2. □

Sym can learn from a variety of orthogonal transformations and achieve invariance. In practice, we fix λ to be Bern(0.5) to balance rotations and non-rotations uniformly, but we parametrize α, β, γ with learnable von Mises (Tikhonov) distributions [MZ75] instead of uniform. Learnable distributions help Sym more efficiently sample orthogonal transformation that benefits the prediction. The von Mises parameterization indeed leads to better empirical performance than uniform, since these distributions closely approximate the wrapped normal distribution on $[-\pi, \pi]$.

The second prediction module takes the invariant embeddings $\boldsymbol{H}$ produced by Sym to perform message passing and predict y. Given the complexity of the prediction problem and to enhance model expressiveness, we also compute attention of edge features and add skip connections during message passing. Specifically, we build up on the Edge Graph Attention Network (EGAT) [KLJ21] model to add edge features to the attention calculation in addition to the regular GAT.

Specifically, after the message from each node is computed, we first calculate an attention score a_{ij} over the edge between nodes i and j. Then the representation of node i in the $l+1$-th layer (e_i^{l+1}) is constructed as the attention-weighted average of the neighbor representations from the l-th layer. We show the formula in the following, where σ represents the non-linear activation function and $\mathcal{N}(i)$ represents the set of neighbors of node i.

$$a_{ij}^l = \frac{\exp x_{ij}^l}{\sum_{k \in \mathcal{N}(i)} \exp x_{ik}^l}, \quad e_i^{l+1} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} a_{ij} e_j^l\Big)$$

4.4.3 Computation Time Analysis

We provide a theoretical analysis of the time complexity of SymGNN against three other baselines including SchNet, DimeNet, and Equiformer. For SymGNN, the time complexity is $O(kn + nd^2)$, where k denotes the number of sampled orthogonal transformation, n denotes the number of nodes, and d denotes the average number of neighbors a node has. The first kn

Table 4.2: Theoretical time complexity of SymGNN and baselines.

	SymGNN	SchNet	Equiformer	DimeNet
Complexity	$O(kn + nd^2)$	$O(nd)$	$\Omega(nd^2)$	$O(n^4)$

term comes from the symmetrization of k orthogonal transformations, the second nd^2 term comes from the attention operation among a node's neighbors. Add these two terms together we essentially get $O(nd^2)$ (empirically we set $k = 6$). In comparison, SchNet is faster as it is $O(nd)$. But for DimeNet, since it considers pairwise edge interaction, the time complexity grows at least as $O(m^2) = O(n^4)$, which makes it prohibitively slow for our graphs with thousands of nodes. For Equiformer, its exact time complexity is unknown to us. It is a transformer-based method where they change attention to equivariant attention and linear layer to equivariant linear layer using complex tensor operations, which implies that its big-O time complexity is lower bounded by $O(nd^2)$. Empirically, we found that training Equiformer is excessively slow for our large-scale graphs. A list of analyzed time complexity can be found in Table 4.2.

4.4.4 Expressiveness Analysis

In this section, we show by examples that SymGNN can capture more complex interactions between molecules compared to SchNet-like methods. Consider two three-molecule systems that have the following configurations where the atom type for node 2 is two whereas the atom type for node 1 and node 3 is one (Figure 4.1):

(a) 1: $(-2, 0, 0)$, 2: $(0, 0, 0)$, 3:$(2, 0, 0)$

(b) 1: $(-\sqrt{2}, \sqrt{2}, 0)$, 2: $(0, 0, 0)$, 3:$(\sqrt{2}, \sqrt{2}, 0)$

Notice that the edge distance between node 2 to each of node 1 and node 3 are two in both of these configurations, so SchNet cannot distinguish these two configurations based on the embedding of node 2, as it only takes into account the distance information. However, we

shall see that SymGNN can differentiate these configurations based on node 2's embedding as it considers also higher-order information. To ease the computation, we consider a minimal setting of SymGNN where the encoder is the identity function, and each time two orthogonal transformations will be applied to the graph and then aggregated. Finally, node 2's embedding is calculated by simply summing up the message passed between node 2 and node 1 and between node 2 and node 3. Suppose the two orthogonal transformations sampled are a counterclockwise rotation in xy-plane by 45 degrees and a reflection around y-axis. It can be calculated that SymGNN will give the embedding $(0, 0, 0)$ for node two in the first configuration, but the second configuration gives $(2, 2+2\sqrt{2}, 0)$. Similar example can be given to show that SymGNN can also detect configurations that have equivalent angle structure, and thus we know SymGNN truly considers higher-level information compared to those invariant methods that are based on scalerization.

4.4.5 Explanations for Structure-EB Relationship

ML models have emerged as powerful tools in scientific research, and their utility can extend beyond mere predictions to explanations. This explanatory aspect is crucial because it aligns with the fundamental objective of ML for science: identifying patterns that can elude human analysis and understanding the underlying mechanisms that govern phenomena.

To make the best use of the SymGNN model and truly bolster the scientific research of MGs, we generate explanations to better reveal the structure-EB relationship. We choose GNNExplainer as a starting point for selecting a subgraph G_S with important edges. Since GNNExplainer was developed for classification problems, the cross-entropy-based objective does not apply to the regression problem of EB prediction. Therefore, we still learn an edge mask $\boldsymbol{M}$ on all edges, but modify the objective in Equation 4.2 by replacing entropy with

mean squared error (MSE) as below, with f representing the SymGNN model.

$$\min_{\mathcal{G}} \mathbb{E}_{G_S \sim \mathcal{G}} \text{MSE}(f(G_S)) = \min_{\boldsymbol{M}} \text{MSE}(f(\sigma(\boldsymbol{M}) \odot G)) \tag{4.6}$$

This regression explainer considers all edges involved in the prediction of EB for one node and assigns a score to each edge. These scores represent the importance of their corresponding edges for making the prediction. In Section 4.6, we demonstrate that the important edges identified by our explainer match the MRO insights mentioned in previous material research and possess unique topological properties.

4.5 Prediction Evaluation

We conduct experiments by first constructing an MG dataset with energy barriers simulated by molecular dynamics. Then we apply SymGNN to this dataset and compare its performance with other baseline models. We also perform ablation studies of the symmetrization module to show its effectiveness.

4.5.1 Dataset

The proposed Cu64Zr36 dataset. We employ molecular dynamics to simulate the behavior of a representative Cu64Zr36 MG subjected to shear deformation. The simulated MG system comprises 8000 atoms, generated through the conventional melting-quenching procedure. To evaluate the influence of system size, we also simulate small systems with 3000 atoms. To ensure the statistical robustness of our findings, 9 independent metallic glass samples are generated. To obtain the energy barriers of atoms, we employ the activation-relaxation technique nouveau (ARTn) [BM96, CLM09] to calculate the energy barriers. The simulated results are used to construct a dataset consisting of nine graphs. Among them, six graphs are used for training, one graph is for validation, and two graphs are for testing. Each

training/validation graph has 8,000 nodes and roughly 260,000 edges, and each test graph has 3,000 nodes and roughly 100,000 edges.

A Detailed Dataset Construction Process The simulated MG system comprises 8000 atoms, generated through the conventional melting-quenching procedure with varied cooling rates spanning from 10^{14} to 10^{10} K/s. To evaluate the influence of system size, we also simulate small system (i.e., 3000 atoms). To initiate the simulation, the sample is initially melted at 2000K under zero pressure for 1ns, facilitating the erasure of its initial configuration memory. Temperature and pressure control are maintained through the isothermal-isobaric (NPT) ensemble, employing a Nosé-Hoover thermostat [Nos84, Hoo85]. Subsequently, the liquified state is rapidly quenched to 1K, with cooling rates ranging from 10^{14} to 10^{10} K/s. The resulting glassy structure is further relaxed to its local energy minimum through energy minimization, utilizing the conjugate gradient algorithm. The interatomic interactions within the system are described using the embedded-atom method (EAM) potential [MKO09]. To ensure the statistical robustness of our findings, 9 independent metallic glass samples are generated for each cooling rate. A timestep of 1fs is adopted for all simulations, and the entire set of simulations is carried out using the LAMMPS package [Pli95]. To obtain the energy barriers of atoms, we employ the activation-relaxation technique nouveau (ARTn) [BM96, CLM09] to calculate the energy barriers within MGs. Specifically, starting from a local energy minimum in the landscape, initial perturbations are introduced to a chosen atom and its nearest neighbors. This perturbation allows exploration along a direction of negative curvature, increasing the likelihood of locating a saddle point in the energy landscape. The Lanczos algorithm [BM96] is then applied to guide the system to the saddle point by following the direction of negative curvature. A force tolerance of $0.05eV/\text{Å}^{-2}$ is chosen to ensure convergence of the saddle points. In accordance with previous investigations [FIE14, FIE17, XFL18], 20 searches for saddle points are conducted for each atom. Consequently, the ARTn exploration focuses on determining the average energy barrier

associated with atoms. This parameter is recognized as a key factor influencing the propensity for plastic rearrangement in disordered materials [TML20, TLM21]. The simulated raw dataset initially only contains nodes (atoms) along with their types and 3D coordinates. We construct edges between two nodes if their Euclidean distance is smaller than a threshold, which is chosen to be $5\text{Å} = 10^{-10}m$.

CuZr-Based MGs as Representative Examples We choose the CuZr-based MGs as representative examples in our experiments. First, Cu-Zr-based metallic glass is one of the most widely investigated MGs due to its outstanding mechanical properties and good glass-forming ability [CM11]. Many well-known studies on MGs, such as those focusing on mechanical properties and ductility [LLL12, PGW10], are centered on Cu-Zr. Additionally, Cu-Zr has been used as a standard MG example in ML research to study β processes [WDM20] and perform hierarchical structure analysis [HNH16]. Cu64Zr36 is known as the best glass former in this class of MGs and is commonly used as the archetype model in MD simulations [WDM20]. While other MGs are not included in this study, some common dynamic behaviors (e.g., relaxation, dynamical heterogeneity, shear band formation) are believed to be controlled by energy barriers, with structure-property relationships transferable between different MGs.

Cu-Zr MGs from [WDM20] We also tested our method on other metallic glass dataset. We adopted the dataset proposed in one of the previous work [WDM20], which includes two additional Cu-Zr type metallic glasses Cu80Zr80 and Cu50Zr50. For each type of material the dataset contains two graphs each with 5000 nodes and around 650000 edges. We picked one as training graph and the other one for testing.

4.5.2 Experiment Settings

Baselines: We evaluate our model against a variety of other ML models including Graph Convolutional Network (GCN) [KW17] withe edge features, E(n) Equivariant GNN (EGNN)

[SHW22] that are designed to handle equivariant features, a non-graph based multi-layer perceptron (MLP) model, and various invariant baselines that are proposed to handle molecular data including SchNet [SKS17], MGCN [LLW19], DimeNet [GGG20], Torch-MD Net [TD22], Equiformer [LS22], and FAENet [DSH23],. Furthermore, we perform two ablation studies named SymGNN w/o symmetrization where we remove the symmtrization layer and Data Augmentation where we only aggregated the embedding from three fixed orthogonal transformation instead of a learned one. In addition we have include a simple baseline in which we use the absolute length of edge instead of its 3D coordinates as an input edge feature to achieve invariance. We also compared to MD based local sampling approximation that is widely used by material scientists [KWY17, SDS98].

Evaluation: The predicted energy barriers are evaluated by the Pearson product-moment correlation coefficient against the true values following from previous work in material science literature [BKG20]. We run each experiment 4 times with different random initializations. On our Cu64Zr36 dataset, we use the validation set to determine the best model and compute the score with the best model on the test set. For the other Cu-Zr MGs dataset, we compute the test accuracy on the final epoch since there is no validation set.

Implementation: For our proposed Cu64Zr36 dataset, we train 4-layer GNNs for 20,000 epochs using an Amsgrad optimizer [RKK19] with a learning rate of 0.0001. We adopt an early stopping scheme if the model's prediction score on the validation set did not improve for 1000 epochs. For the other Cu-Zr MGs dataset, we train each model to a fixed number of epochs as there is no validation set. For those models that have smaller scale and faster convergence, i.e., MLP, GCN, EGNN, EGAT, SchNet, and MGCN, we train to 5000 epochs. For SymGNN, we trained the model to 10000 epochs for better convergence. In all the datasets for SymGNN, the distribution over the angles α, β, and γ is parameterized by the von Mises (Tikhonov) distribution.

Table 4.3: **Testing scores comparison of SymGNN, the molecular dynamics (MD) method, and other ML methods.** Test results are with the best model on the validation set. Our SymGNN significantly outperforms the MD method and achieves the best among all the ML methods.

	Methods	Cu64Zr36	Cu80Zr20	Cu50Zr50
MD	Local Sampling [SDS98]	0.3614	–	–
Non-Invariant ML	MLP	0.0575 ± 0.0127	0.0727 ± 0.0154	-0.0652 ± 0.0099
	GCN with Edge Features	0.5123 ± 0.0507	0.2478 ± 0.0051	0.1395 ± 0.0068
Invariant ML	E(n) Equivariant GNN	0.2588 ± 0.0077	0.1382 ± 0.0113	0.1381 ± 0.0098
	EGAT (Edge Length as 1D Feature)	0.7264 ± 0.0063	0.5489 ± 0.0218	0.1571 ± 0.0095
	SchNet	0.7588 ± 0.0088	0.2505 ± 0.0128	0.1808 ± 0.0106
	MGCN	0.7352 ± 0.0066	0.1793 ± 0.0133	0.1596 ± 0.0033
	FAENet	0.6603 ± 0.0218	0.2947 ± 0.0171	0.2214 ± 0.0160
Ours	SymGNN	**0.7859 ± 0.0056**	**0.6084 ± 0.0167**	**0.5862 ± 0.0277**
	SymGNN w/o symmetrization	0.2669 ± 0.0371	0.2283 ± 0.0256	0.1135 ± 0.0129
	Data Augmentation	0.6614 ± 0.0285	0.3304 ± 0.0201	0.2135 ± 0.0337

4.5.3 Prediction Results

We report the results of SymGNN and other baselines in Table 4.3. It can be seen from the table that SymGNN outperforms the baselines by a large amount and exhibits a much stronger generalization power. When we remove the symmetrization module, (i.e. SymGNN w/o symmetrization), the ablated model cannot generalize well, and a similar performance drops is observed when we only aggregates embedding from three fixed orthogonal transformations. This demonstrates the effectiveness of the symmetrization module. Also we observe that models capable of handling invariance can lead to much better result compared to the ones that cannot, which again highlights the importance of symmetrization module in achieving good prediction performance. In addition, we also ran Equiformer, Torch-MD Net, and DimeNet as our baselines. However, we noticed that the training time for these methods are prohibitively long (i.e longer than 2 days) on our dataset.

Table 4.4: Training time comparison for one epoch between SymGNN and baselines.

Method	SymGNN	SchNet	Equiformer	DimeNet
Time	3 secs	1 sec	82 mins	-

Table 4.5: Inference time comparison on an MG with 3,000 atoms between SymGNN and baselines.

	SymGNN	ARTn	MD local sampling
Time	0.26 seconds	41 days	150 mins

4.5.4 Computation Time Comparison

We notice that SymGNN reaches high performance without dramatically increase both the training cost or the inference cost. Table 4.4 provides an empirical time comparison for the time needed to train the model for one epoch. We observe that DimeNet would run indefinitely for our larger graph, and the time taken by Equiformer is also prohibitively long. Table 4.5 shows an inference time comparison. Compared to traditional MD simulation, our ML-based approach needs much fewer computation resources and is much more efficient. For precise MD simulation with ARTn, the calculation of the energy barrier for each atom takes around 20 minutes in a supercomputer with 16 parallel threads. Therefore, for a MG system that has the size of our test graph, i.e., 3,000 atoms, the computation will take $\frac{20\times3000}{60\times24} \approx 41$ days. Even for the much faster and less inaccurate local sampling method, the inference time for this MG system can take 150 minutes. In contrast, SymGNN's inference time on the test graph is almost negligible.

4.5.5 Abalations and Further Comparisons

Abalations on Dataset Splits To check our method's robustness under different dataset configurations, we performed an ablation study with two more random dataset splits. These

Table 4.6: Performance comparison of the original and new dataset splits between SymGNN and SchNet.

	Model	Train	Test
Original	SymGNN	0.8368	**0.7859**
	SchNet	0.7858	0.7588
New Split 1	SymGNN	0.8600	**0.7613**
	SchNet	0.7778	0.7583
New Split 2	SymGNN	0.8362	**0.7645**
	SchNet	0.7070	0.6948

splits use different sets of randomly sampled training graphs, one randomly sampled graph for validation, and rest for testing. The result can be found on Table 4.6. We see that SymGNN consistently outperforms SchNet in all the dataset splits we considered.

Ablation on Number of Orthogonal Transformations In our experiments, we found that the model performance is relatively stable with respect to the number of transformations. We performed an ablation study with different number of aggregated orthogonal transformations. We presented experiments results with 2, 4, 6, and 12 transformations. Although using 12 transformations led to out-of-memory (OOM) issues, we found that both 2 and 4 transformations yielded effective results, with the performance using 2 transformations even being slightly better than our reported results. We hypothesize that as long as we are sampling various orthogonal transformations from a reasonable distribution, the framework can benefit from the symmetrization module and achieve good results. The number of transformations mostly influences the convergence time rather than the performance. For example, using 2 transformations required 3000 epochs for convergence, while 4 transformations required 2300 epochs. This observation aligns with our expectations, as fewer transformations necessitate more iterations for the model to capture the necessary information from the data. The results can be found in Table 4.7.

Table 4.7: SymGNN performance ablated on different number of orthogonal transformations.

Number of Transformations	0	2	4	6	12
Train	0.8736	0.8302	0.8072	0.8368	OOM
Test	0.2669	0.7901	0.7778	0.7858	OOM

Comparison with Data Augmentation Empirically, we find that our symmetrization module can learn a condensed subspace of the orthogonal transformations corresponding to the task, which allows more effective aggregations. In this section, we provide an analysis with the subspace learned by SymGNN. We report the mean and concentration for each distribution that controlled one Euler angle after the training. For rotations, we have

$$(\mu_i, \kappa_i) = (-0.3414, 0.2985), (0.7023, 0.9146), (-1.5622, 1.3543)$$

and

$$(\mu_i, \kappa_i) = (-0.4102, 3.0350), (-0.4946, 3.0645), (-0.7043, 0.1533)$$

for the rest. We perform an estimation of how much volume we need in order to capture 80 percent of the whole probability density. We notice that Von Mises distribution with a bigger κ is approximately a Gaussian distribution with variance $\frac{1}{\kappa}$, and a Von Mises distribution with a smaller κ is approximately a uniform distribution. Therefore, we approximate the estimation using a similar density-volume estimation with four Gaussian random variables $\mathcal{N}(-1.5622, \frac{1}{1.3543}), \mathcal{N}(-0.4102, \frac{1}{3}), \mathcal{N}(-0.4946, \frac{1}{3}), \mathcal{N}(0.7023, \frac{1}{0.9146})$, and two uniform distributions between $[-\pi, \pi]$. We found that for these four Gaussian distributions, intervals of length $2.21, 3.37, 3.37, 4.01$ around the mean can approximately yield 94.5 percent of density. Therefore, in total it will yield $0.945^4 \approx 0.8$ of density. By picking all the uniform distribution, we know that at least 80 percent of the density can be included by less than 7 percent of the density. This analysis implies the effectiveness of the learning. We should note that this estimation is rough and an overestimation, since in reality the distributions that are approximated as uniform are also more centered.

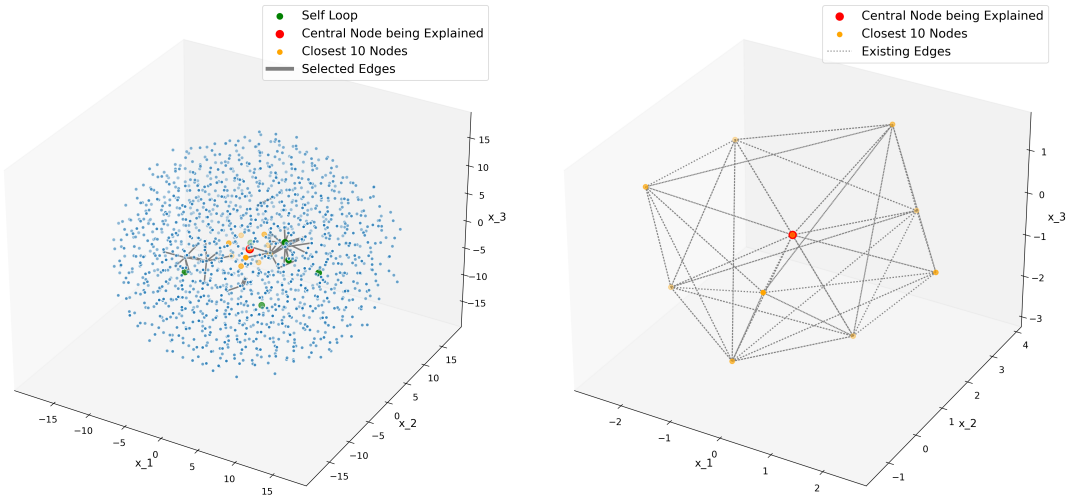


Figure 4.4: Global and local explanation visualizations for SymGNN on MG node 1501.

4.6 Explanation Evaluation and Analysis

We produce explanations using the method in Section 4.4.5. We first provide visualizations to qualitatively show our explanation, and then we do a quantitative evaluation by connecting of our explanation to the MRO hypothesis and the topological data analysis (TDA) to reveal more insights.

4.6.1 Explanation Visualization and MRO

We visualize our explanation for a randomly sampled node. We provide both the global and the local version. All atoms are plotted in their actual 3D coordinates. The explanation is presented in Figure 4.4. For the global version, we visualize the top 50 important edges. For the local version, we zoom in to the top 10 closet nodes. From the visualizations, we see that edges close to the central node or far away from the central node may both be selected. Also, as we can see from the local version of the explanation, few edges within the top 10 closet edges is selected by our explanation.

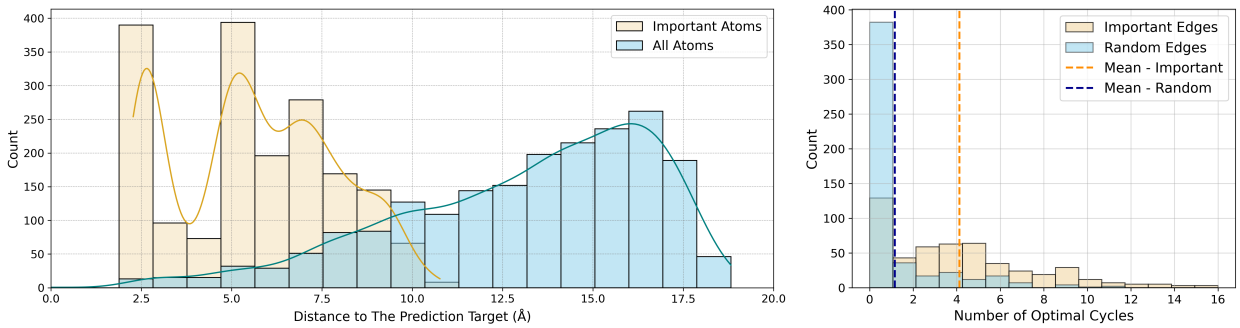


Figure 4.5: **Analysis of explanations for SymGNN prediction.** (a) Distribution of distance to the prediction target (central node) of important atoms identified by our explanation vs. all atoms. (b) Distribution of the number of cycles involved in important edges identified in our explanation vs. randomly selected edges.

Material scientists proposed an MRO hypothesis, which basically says the atoms that lie in a range of medium distance (5 - 10 Å) from the central atom play a more important role in determining its MG properties [MSW09, SLA06, NCG21, EDR23]. We aggregate our selected edges to nodes (atoms) and plot the atom importance against their distance to the central node in Figure 4.5 (a). We see there are two modes of the important atoms, one for the closest ones and one for the medium-range ones, which matches the MRO hypothesis.

4.6.2 Edge Importance Explanation and TDA

We perform TDA to further understand our edge importance explanations and see if the results recover meaningful topological structures.

Persistent Homology (PH) PH is a widely used TDA method, a field of study that applies concepts from algebraic topology to data analysis [Bar94, ZC04, EH08]. PH examines how topological features such as connected components, holes, or voids, emerge and disappear as one moves through different scales in a dataset. The persistence of certain topological features across scales can reveal important insights about the underlying structure of the data, making PH a powerful tool for material science. This method has also been applied to metallic glass to uncover important topological structures [SBB20]. In PH, the concepts

Table 4.8: **SymGNN explanation and optimal cycles.** Average number of optimal cycles participated by edges with high/medium/low importance according to the SymGNN explanation and randomly selected edges.

Edge Importance	High	Medium	Low	Random
Avg # Optimal Cycles	4.130	1.202	0.874	1.148

of "birth" and "death" are the essential quantities we would like to study, which visually represent the lifespan of topological features in a dataset. "Birth" refers to the scale at which a feature, like a connected component or a hole, first appears during the filtration process, while "death" denotes the scale at which this feature disappears or merges.

Explanation Results In our case, we apply PH to study the emergence and death of 1D hole as we increase the radius of a ball surrounding each atom. We perform the inverse analysis to pair the hole with a representative optimal cycle [Oba18]. In this way, each edge in the graph can be associated with a sequence of births and deaths of the cycles that it has participated in. We perform statistical analysis to see if there is significant difference between selected edges by our explanation and other edges. We plot and compare the distribution of the number of optimal cycles involved in the highest importance edges selected by our explanation versus randomly selected edges over multiple central nodes that are being explained in Figure 4.5 (b). We found that on average the importance edges participates in much more cycles compared to others, and there is a clear trend in the decrease in cycle number as the importance of edges decrease. The mean of the number of cycles involved in by edges in the four different group can be found in Table 4.8.

4.7 Discussion

In this paper, we study the connection between the local atomic structures of MGs and their EBs of the energy landscape. We formalize this problem as node regression on

graphs and propose SymGNN to solve the problem by effectively capturing the invariance of orthogonal transformations of the graph. We compare SymGNN with several baseline models and demonstrate that SymGNN performs the best. In addition, we extend the GNNExplainer to regression tasks and generate explanations. We further investigate the explanations with MRO and PH. We show a strong correlation between the importance of edge and the number of optimal cycles they involved in. Our work enables effective prediction and interpretation of MG EBs, bolstering material science research.

CHAPTER 5

Motif Mining via Clustering Representations of Graphs

5.1 Introduction

Graph-structured data is ubiquitous in scientific research and real-world applications. Especially in biochemistry, small chemical compounds and proteins can both be modeled via graphs with atoms and helix/sheet as nodes, respectively. Graph motifs, defined as frequent and significant subgraph patterns [MSI02], are the building blocks of graph data. They usually represent important graph characteristics and help uncover the properties of complex graphs. For example, important motifs can determine molecule properties. Like hydroxide (–OH), which is a chemical functional group, usually implies higher water solubility for chemical compounds, and Zif268, which is a protein structure, can mediate protein-protein interactions in sequence-specific DNA-binding proteins [PPG01]. Graph motif mining is a classic data mining problem with many important applications like quantum chemistry calculation and drug discovery [REW19]. When a set of motifs are mined from a graph dataset, a bag-of-motifs vector representation, which we call a *motif descriptor*, can be generated as an interpretable feature to describe the graph property [DKW05]. Thus, a model utilizing such motif descriptor is capable to make interpretable predictions for downstream graph tasks, e.g. which functional group in a molecule plays a decisive role in toxicity classification.

However, mining motifs from large graph datasets remains a challenging question and requires a proper definition of motifs. Traditional motif mining methods define motifs as discrete subgraph structures and use subgraph isomorphism test to perform exact pattern

match [MSI02, KIM04, CHL06, Wer06]. This definition can sometimes be too restrictive to capture the semantic meanings of motifs and ignore the cases where structurally different motifs have similar properties. For example, sulfate ($-SO_4^{-2}$) and chloride (Cl^-) are two very different subgraphs when measured with structure similarity. However, their chemical properties are actually very similar in many cases as they correspond to two strong acids, i.e. sulfuric acid (H_2SO_4) and hydrochloric acid (HCl) respectively. In view of this, a desirable definition of motifs should be able to capture the semantic similarity between these two functional groups and group them to the same motif. We thus propose to define motifs using continuous embedding vectors rather than discrete structure patterns. As is illustrated in Figure 5.1, such a definition is more expressive and flexible to enable structurally different subgraphs that share similar semantic properties, e.g. $-CF_3$ and $-CCl_3$, to be grouped into the same motif.

Our novel reformulation of the motif definition converts the NP-hard subgraph isomorphism test problem into a feasible clustering-based representation learning problem. To learn such motif embedding, it's required to properly encode different subgraphs. Recently, GNNs have shown expressive power for learning representations of graph data [KW16, HYL17, VCC17, XHL18]. The problem of GNNs is that they are black-box models lacking interpretability. Combining the less-interpretable GNNs with motif mining brings the best of two worlds together.

Leveraging the reformulated graph motifs and expressive GNN encoder, we propose Motif mIning via Clustering RepresentatiOns of Graphs (MICRO-Graph). MICRO-Graph sets up a probabilistic model on graph data. We treat each graph as a set of nodes, represented by continuous GNN embedding vectors that are consistent with the graph topology. Each motif is represented by a vector distribution, e.g. a multivariate Gaussian distribution. For each node, we first sample its hidden motif, then sample a node vector from the corresponding motif distribution. For each given graph $\mathcal{G}$, a motif descriptor, which describes $\mathcal{G}$ as a bag of motifs, can be inferred by the MICRO-Graph via first encoding $\mathcal{G}$ and then computing the

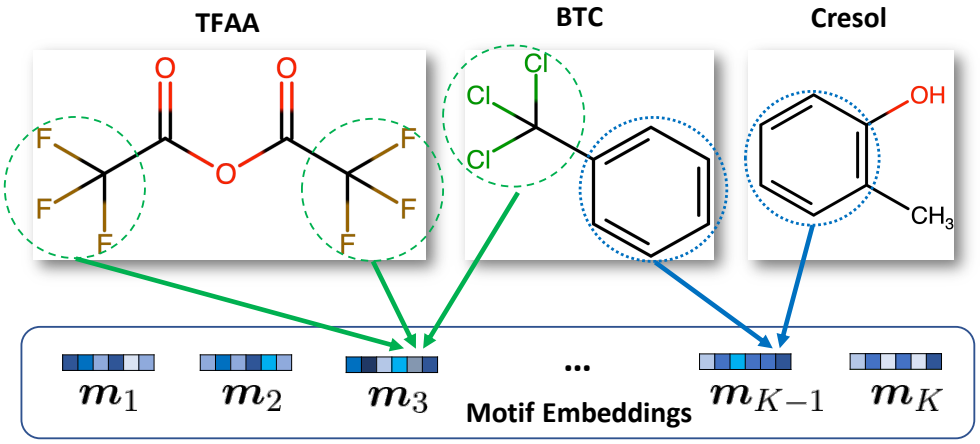


Figure 5.1: **Difference between the traditional motif definition and our motif definition.** Given three chemical compounds, the traditional definition based on discrete structures can only mine the benzene rings in blue circle. In contrast, our definition using continuous embedding vectors can mine the benzene ring as one motif and also the semantically similar $-CF_3$ and $-CCl_3$ as another motif.

dot product between node embeddings and motif embeddings. Also, as the whole framework is differentiable with respect to both the generative model parameters and GNN encoder parameters, they thus can be learned jointly with gradient updates.

We show that MICRO-Graph can mine the ground truth motifs from synthetic graphs and interpretable motifs from chemical compound graphs. The mined motifs from the HIV chemical dataset [HFZ20] correspond to meaningful functional groups including benzene rings, amine, carboxylic acid, etc. Meanwhile, the interpretable motif descriptors generated by MICRO-Graph serve as discriminative graph features and outperform graph features extracted by other unsupervised learning baselines on downstream supervised tasks.

5.2 Related Work

Clustering-Based Representation Learning. Clustering algorithms are classical unsupervised learning techniques and they have been used for representation learning recently

[CBJ18, YCA20, CMM20, LZX20]. The clustering-based representation learning methods usually iterate between two steps to jointly learn data representations and cluster assignments of the representations. Specifically, the first step fixes the data representations and computes the cluster assignment of each data representation by a clustering algorithm, e.g. K-Means in [CBJ18]. The second step fixes the cluster assignments from the first step and uses them as labels to update data representations. Following this iterative paradigm, [CBJ18] shows that a simple K-means clustering can help to learn meaningful image representations. However, such a paradigm has a degenerate solution where all clusters collapse into a single one, so the sEM algorithm [MNV20, GDV19] was proposed as a better optimization paradigm. [YCA20, CMM20] shows that the sEM with the equal-cluster-size constraint can learn better representations and scale on large datasets. In our case, learning motif embedding vectors can be interpreted as clustering node embeddings, where the motif embeddings correspond to the cluster centers.

Interpretable Graph Machine Learning Interpretability is an important research topic for modern machine learning models including the GNNs on graph data [DM21, YBY19, LCX20, VT20, HYT20, ZLS22]. The goal for interpreting a graph model is formalized as identifying the import subgraph for the model to make its prediction. From this perspective, the motif-based graph machine learning models [DKW05, WWK08] are thus naturally interpretable, including the motif descriptor generated by MICRO-Graph. Our case of motif mining via GNNs thus adds interpretability to the powerful but less interpretable GNNs.

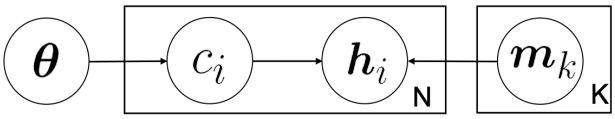


Figure 5.2: Probabilistic graphical model of the graph generating process in MICRO-Graph.

5.3 Micro-Graph: Motif Mining via Clustering Representations of Graphs

5.3.1 Probabilistic Modeling

Our goal is to mine graph motifs through representation learning. Unlike the traditional motif mining methods which define motifs by discrete subgraph patterns, we represent motifs as multivariant Gaussian distributions, with the motif embedding vectors $\boldsymbol{M} = \{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K | \boldsymbol{m}_k \in \mathcal{R}^d\}$ as means. We then set up a probabilistic model for generating graphs from K motif clusters. If we could learn a set of motifs to reconstruct each graph in the dataset, then these motifs could capture the important properties of these graphs.

Model setup. Let $\{\mathrm{c}_1, \ldots, \mathrm{c}_N\}$ be N latent index random variables corresponding to K motifs, and each c_i follows a categorical distribution $Cat(\boldsymbol{\theta})$ such that $P(\mathrm{c}_i = k) = \boldsymbol{\theta}_k$ for $k \in [1, K]$. Given a graph $\mathcal{G}$ with the number of nodes $|\mathcal{V}| = N$, we generate all the nodes in graph $\mathcal{G}$ from motifs. Specifically, for each node $v_i \in \mathcal{V}$, we first sample its motif index c_i. Then, conditioned on c_i, we sample a vector $\boldsymbol{h}_i$ which represents the information of node v_i from the multivariate Gaussian as in Equation 5.1.

$$\boldsymbol{h}_i | \mathrm{c}_i = k \sim \mathcal{N}(\boldsymbol{m}_k, \sigma^2 \boldsymbol{I}) \tag{5.1}$$

The $\boldsymbol{h}_i \in \mathcal{R}^d$ is a contextualized node embedding encoded with an L-layer GNN f_ϕ. It contains both feature and structural information corresponding to an L-hop subgraph. Considering the challenges of generating the discrete graph structure, we propose to generate $\boldsymbol{h}_i$ instead

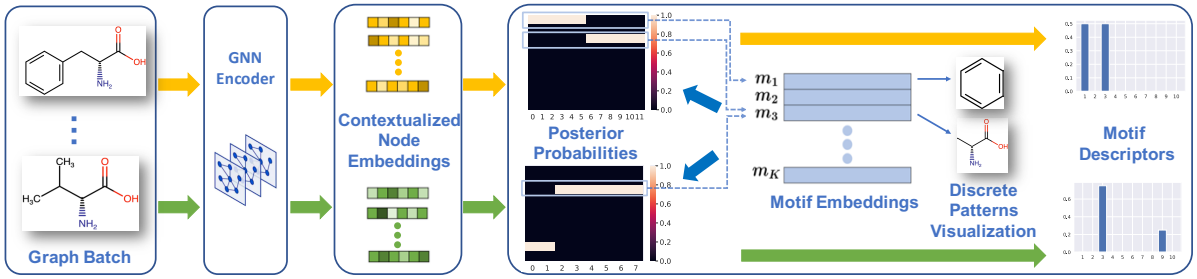


Figure 5.3: **The** MICRO-Graph **framework.** MICRO-Graph infers motif descriptors and visualizes mined motifs via two example chemical graphs follow the yellow and green flows respectively. For each graph, it is first encoded by GNN to get node embeddings $\boldsymbol{h}_i$. Then the posterior probabilities (soft assignments) $\boldsymbol{P}$ are computed (Equation 5.4). Averaging $\boldsymbol{P}$ column-wise gives the motif descriptor $\boldsymbol{z}$ (Equation 5.5). Meanwhile, we group nodes and visualize their induced subgraph patterns by turning soft assignments to hard assignments (Equation 5.6 & 5.7).

of the discrete v_i. $\boldsymbol{h}_i$ approximates the structural role of v_i, and we also add regularization to encourage $\boldsymbol{h}_i$ encodes more structure information as elaborated later in Section 5.3.4. The overall log-likelihood of the probabilistic model is shown in Equation 5.2.

$$\ell(\boldsymbol{M}, \boldsymbol{\theta}, \phi) = \log \prod_{i=1}^{N} (\sum_{k=1}^{K} P(\mathrm{c}_i = k, \boldsymbol{h}_i)) \tag{5.2}$$

$$\propto \sum_{i=1}^{N} \log \sum_{k=1}^{K} \exp(\frac{\boldsymbol{h}_i^T \boldsymbol{m}_k}{\sigma^2}) \boldsymbol{\theta}_k \tag{5.3}$$

We show the graphical model of the generating process in Figure 5.2. An analogy can be drawn between our model and the PLSA topic model with nodes corresponding to words and motifs to topics.

5.3.2 Motif Descriptor Inference

The motifs $\boldsymbol{M}$ and GNN $f_\phi(\cdot)$ can be used to infer the semantic motif descriptors $\boldsymbol{z} \in \mathcal{R}^K$ as K-dimensional feature vectors of both seen and new graphs. We show an example of the inference process in Figure 5.3 and illustrate the detailed inference steps below.

Compute soft node-to-motif assignments. For a given graph $\mathcal{G}$, we first perform GNN encoding to get all the contextualized node embeddings $\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N\} = f_\phi(\mathcal{G})$. Then we calculate the posterior probability of latent variable $\mathrm{c}_i = k$ conditioned on $\boldsymbol{h}_i$ and denote all such probabilities using a matrix $\boldsymbol{P} \in \mathcal{R}_+^{K \times N}$ as in Equation 5.4

$$\boldsymbol{P}_{k,i} = P(\mathrm{c}_i = k|\boldsymbol{h}_i) = \frac{\exp(\boldsymbol{m}_k^\top \boldsymbol{h}_i/\sigma^2)\boldsymbol{\theta}_k}{\sum_{k'} \exp(\boldsymbol{m}_{k'}^\top \boldsymbol{h}_i/\sigma^2)\boldsymbol{\theta}_{k'}} \tag{5.4}$$

Thus, each column $\boldsymbol{P}_{:,i}$ can be interpreted as a soft assignment for clustering node embedding $\boldsymbol{h}_i$ to motif k, and the average of over all N nodes gives the motif descriptor $\boldsymbol{z}$ as in Equation 5.5

$$\boldsymbol{z} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{P}_{:,i} \tag{5.5}$$

The descriptor $\boldsymbol{z}$ specifies $\mathcal{G}$ as a normalized bag of motifs, and it can be used as the graph feature for downstream supervised tasks. We show that the descriptor $\boldsymbol{z}$ inferred by our MICRO-Graph learned through unsupervised learning could achieve competitive results to SOTA supervised GNNs.

5.3.3 Visualize and Interpret Mined Motifs

Besides the descriptors, the learned motifs $\boldsymbol{M}$ can also be visualized and interpreted as discrete patterns by grouping nodes assigned to the same motif. We show an example of the visualization process in Figure 5.3 and illustrate the detailed steps below.

Node selection for motif k. To get the discrete pattern corresponding to a continuous motif embedding, we first get the most likely motif cluster that each node is assigned to. We turn the soft assignment $\boldsymbol{P}_{:,i}$ to a hard assignment $\hat{\boldsymbol{P}}_{:,i}$ via the indicator function $\boldsymbol{I}(\cdot)$ as in Equation 5.6.

$$\hat{\boldsymbol{P}}_{k,i} = \boldsymbol{I}(k = \arg\max_{k'} \boldsymbol{P}_{k',i}) \tag{5.6}$$

Induced subgraph generation Then we group nodes assigned to motif k and visualize

Algorithm 5 MICRO-Graph Learning Loop

Input: Graph dataset, GNN encoder $f_\phi(\cdot)$, random initialized $\boldsymbol{M} \in \mathcal{R}^{K\times d}$ and $\boldsymbol{\theta} \in \mathcal{R}^K$, hyperparameter λ and σ.
for A graph batch $\{\mathcal{G}_1, \ldots, \mathcal{G}_B\}$ **do**
  Encode all $\mathcal{G}_b$ with $f_\phi(\cdot)$ to get node embeddings $\boldsymbol{H} \in \mathcal{R}^{B\times N,d}$.
  $\boldsymbol{P} = (\log \boldsymbol{\theta}_k)^\top \boldsymbol{M}\boldsymbol{H}^\top/\sigma^2$ // proportional to log joint prob
  $\boldsymbol{Q}^* = \text{Sinkhorn}(\boldsymbol{P})$ // sEM E-step, no gradient
  $\mathcal{L}_{OT} = (\boldsymbol{Q}^* * * \log \boldsymbol{P}).mean()$ // sEM M-step, ** means element-wise product
  $\mathcal{L}_{cut} = 0$
  for $\mathcal{G}_b \in \{\mathcal{G}_1, \ldots, \mathcal{G}_B\}$ **do**
    Select $\boldsymbol{P}^b$ from $\boldsymbol{P}$ correspond to $\mathcal{G}_b$
    $\boldsymbol{D}^b = Diag(\sum_i \boldsymbol{A}^b_{i,:})$ with $\boldsymbol{A}^b$ corresponds to $\mathcal{G}_b$
    $\mathcal{L}_{cut} += Tr(\boldsymbol{P}^b\boldsymbol{A}^b\boldsymbol{P}^{b\top})/Tr(\boldsymbol{P}^b\boldsymbol{D}^b\boldsymbol{P}^{b\top})$ // Equation 5.10
  end for
  $\mathcal{L} = \alpha\mathcal{L}_{OT} + (1-\alpha)\mathcal{L}_{cut}/B$
  Back propagate gradient from $\mathcal{L}$ to update $\boldsymbol{M}$, $\boldsymbol{\theta}$, and ϕ.
end for

their induced subgraph as a discrete correspondence of $\boldsymbol{m}_k$. In practice, there can be multiple subgraph patterns corresponding to the same $\boldsymbol{m}_k$ from different graphs. Like in Figure 5.3, both chemical compounds contain the amino acid ($-NH_2COOH$) motif. For such cases, we get the subgraph-level representation $\boldsymbol{s}_k$ by averaging node embeddings as in Equation 5.7.

$$\boldsymbol{s}_k = \frac{\sum_i \hat{\boldsymbol{P}}_{k,i}\boldsymbol{h}_i}{\sum_i \hat{\boldsymbol{P}}_{k,i}} \tag{5.7}$$

Subgraph selection via dot product. In practice, there will be multiple subgraphs from different graphs corresponding to each motif k. We thus pick the subgraph with the largest $\boldsymbol{m}_k^\top \boldsymbol{s}_k$ to visualize. As we will see, the discrete patterns corresponding to mined motifs often uncover the characteristics of the graph $\mathcal{G}$ and match the real motifs from domain knowledge.

5.3.4 MICRO-Graph **Parameter Learning**

For MICRO-Graph, parameters that we need to learn include both the clustering parameters $\boldsymbol{M}$ and $\boldsymbol{\theta}$ in the probabilistic model and the GNN parameters ϕ for representation

learning. A straightforward choice is to optimize these parameters regarding the log-likelihood function in Equation 5.2. However, several previous works have demonstrated that learning both the clustering parameters and encoder parameters can lead to a degenerate solution where all clusters collapse to one [YCA20, GDV19].

Objective prevents cluster collapse. We consider another objective based on entropic optimal transport (OT), which is slightly different from the log-likelihood, but shares the same global optimum and help prevent cluster collapse as shown in [MNV20]. We show the objective in Equation 5.8 and explain it below.

$$\mathcal{L}_{OT}(\boldsymbol{M}, \boldsymbol{\theta}, \phi) = \max_{\boldsymbol{Q} \in U(r,c)} \sum_{i=1}^{N} \sum_{k=1}^{K} \boldsymbol{Q}_{k,i} \log P(\mathrm{c}_i = k, \boldsymbol{h}_i) + \frac{1}{\lambda} H(\boldsymbol{Q}) \tag{5.8}$$

$U(\boldsymbol{r}, \boldsymbol{c}) = \{\boldsymbol{Q} \in \mathcal{R}_+^{K \times N} | \boldsymbol{Q}\mathbf{1_N} = \boldsymbol{r}, \boldsymbol{Q}^\top \mathbf{1_K} = \boldsymbol{c}\}$ in $\mathcal{L}_{OT}$ is the transport polytope of two probability vectors $\boldsymbol{r} = \frac{\mathbf{1_K}}{K}$ and $\boldsymbol{c} = \frac{\mathbf{1_N}}{N}$ with $\mathbf{1_K}(\mathbf{1_N})$ stands for the $K(N)$-dimensional all-one vector. $\boldsymbol{Q}$ is a $K \times N$ matrix in $U(\boldsymbol{r}, \boldsymbol{c})$ where the summation over columns and rows equals $\boldsymbol{r}$ and $\boldsymbol{c}$ respectively. $\boldsymbol{Q}$ can be interpreted as the joint distribution of two random variables where their marginal distributions are $\boldsymbol{r}$ and $\boldsymbol{c}$. Also, $H(\boldsymbol{Q}) = -\sum_{k,i} \boldsymbol{Q}_{k,i} \log \boldsymbol{Q}_{k,i}$ is the entropy of $\boldsymbol{Q}$, with a hyperparameter factor $\frac{1}{\lambda}$. $\mathcal{L}_{OT}$ is actually equivalent to the evidence lower bound of the log-likelihood, with a restricted searching space for $\boldsymbol{Q} \in U(\boldsymbol{r}, \boldsymbol{c})$ and an extra entropy term controlling the smoothness of $\boldsymbol{Q}$. These two restrictions work together to make sure the parameters for each cluster are optimized in balance.

OT objective optimization. We use the Sinkhorn expectation-maximization (sEM) algorithm [MNV20] to optimize $\mathcal{L}_{OT}$, because the Sinkhorn algorithm can solve the maximization problem over $\boldsymbol{Q} \in U(\boldsymbol{r}, \boldsymbol{c})$ in $\mathcal{L}_{OT}$ very fast [Cut13] and has been shown to enjoy good convergence [MNV20]. Specifically, the E-step of sEM solves maximization problem over $\boldsymbol{Q} \in U(\boldsymbol{r}, \boldsymbol{c})$ and finds the optimal $\boldsymbol{Q}^*$. Then the M-step of sEM as takes $\boldsymbol{Q}^*$ as the reference distribution like the standard EM [DLR77] and optimizes $\mathcal{L}_{OT}$ over $\boldsymbol{M}$, $\boldsymbol{\theta}$ and ϕ.

After dropping the terms independent of $\boldsymbol{M}$, $\boldsymbol{\theta}$ and ϕ, the M-step reduces to Equation 5.9

$$\mathcal{L}_{OT}(\boldsymbol{M}, \boldsymbol{\theta}, \phi) \propto \sum_{i=1}^{N} \sum_{k=1}^{K} \boldsymbol{Q}^{*}_{k,i} \log \boldsymbol{P}_{k,i} \tag{5.9}$$

Enhance structure coherence. In the probabilistic model, we used $\boldsymbol{h}_i$ to approximate the structural role of v_i. To further improve this approximation and increase the structural coherence of learned motifs, we add an extra term $\mathcal{L}_{cut}$ to M-step objective in Equation 5.9. The $\mathcal{L}_{cut}$ is defined in terms of the posterior probability matrix $\boldsymbol{P}$ in Equation 5.4, graph adjacency matrix $\boldsymbol{A}$, and graph degree matrix $\boldsymbol{D}$ as in Equation 5.10.

$$\mathcal{L}_{cut}(\boldsymbol{M}, \boldsymbol{\theta}, \phi) \ = \frac{Tr(\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^{\top})}{Tr(\boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{\top})} \tag{5.10}$$

The cut loss $\mathcal{L}_{cut}$ is deduced from the K-way normalized min-cut problem. Maximizing it regarding a given cluster assignment is meant to increase the number of intra-cluster edges and decrease inter-cluster edges, which helps clustering nodes based on their structure roles as a complement to the node embedding $\boldsymbol{h}$.

The intuition behind the cut loss is the following. Graph motif mining in general is considered a hard frequent pattern mining problem, say harder than mining set data, because of the topological structure between graph nodes. The graph structure is the main distinguishing factor between graph and non-graph data. It describes dependencies between nodes and contains critical information for learning motifs. According to the definition of motifs, nodes forming a frequent and significant structure should be considered as from one motif. For example, the benzene ring as an important and commonly seen functional group in chemical compounds is a motif we want to learn. It requires the method to capture the hexagon structure and assign all those six nodes to the same motif. The $\mathcal{L}_{OT}$ loss is designed with no special structure information considered. Although GNN message-passing has been show to be able to capture some graph structural information in the node embeddings [CCV20]. However, most GNNs used in practice lack the ability to fully capturing the

structural information through message passing [XHL18, DBY19, GJJ20, BFZ21].

The $\mathcal{L}_{cut}$ is based on the classic spectral clustering algorithm on the graph structure. For nodes clustered into the same spectral cluster, we increase their probability of being assigned to the same motif. Another interpretation is that we regularize the assignment result of our probabilistic model, so it is not too different from the spectral clustering result. Note the consistency with spectral clustering is a regularization rather than strictly forced. We don't directly use the spectral clustering result as our assignment because it is solely based on the graph structure, which ignores the rich node features and won't give us the ideal motifs with semantic meanings. Spectral clustering is non-differentiable and has $\mathrm{O}(N^3)$ time complexity. An in-depth discussion of this approximation can be found in [BGA20].

The overall objective. $\mathcal{L}$ in Equation 5.11 combines the $\mathcal{L}_{OT}$ and $\mathcal{L}_{cut}$ together with a weight parameter α. $\mathcal{L}$ is differentiable and is optimized regarding the $\boldsymbol{M}$, $\boldsymbol{\theta}$, and ϕ via gradients.

$$\mathcal{L}(\boldsymbol{M}, \boldsymbol{\theta}, \phi) = \alpha \mathcal{L}_{OT}(\boldsymbol{M}, \boldsymbol{\theta}, \phi) + (1 - \alpha)\mathcal{L}_{cut}(\boldsymbol{M}, \boldsymbol{\theta}, \phi) \tag{5.11}$$

We also summarize the motif mining procedure and show the learning loop of MICRO-Graph in Algorithm 2. We assume the graph dataset has been padded with N nodes for each graph, and we use SINKHORN to denote the fast iterative algorithm in [Cut13] and $**$ to denote element-wise matrix multiplication.

5.4 Evaluation

In this section, we evaluate both the mined motifs and motif descriptors by MICRO-Graph on graph datasets from different domains. We first do a qualitative analysis of the mined motifs by visualization and interpretation, where we compare mined motifs to ground truth motifs from domain knowledge for chemical datasets. Then we do a quantitative analysis of

motif descriptors by treating them as graph features and evaluating them on downstream supervised tasks.

5.4.1 Datasets

We consider three types of datasets including synthetic graphs with known ground truth motifs, chemical compounds, and proteins. We introduce each dataset below.

Synthetic Dataset. Evaluating mined motifs is non-trivial for real datasets because the ground truth relies on domain knowledge and is sometimes even unknown. We thus follow [LCX20] to generate synthetic MOTIF-BA graphs with known ground truth motifs. Each MOTIF-BA graph is created by first generating a 10-node base Barabási–Albert (BA) graph, and then attaching synthetic motifs selected from five pre-defined motifs to randomly selected nodes in the base graph. In our experiment, the MOTIF-BA dataset contains all 2-motif-combination and 3-motif-combination graphs, each with 30 different instances, which leads to 20 different labels y and 600 graphs in total. We assign all nodes in all graphs a 10-dimensional all-one vector as the feature.

Chemical Compounds. We consider seven chemical compound datasets: BACE, BBBP, CLINTOX, HIV, SIDER, TOX21, and TOXCAST [WRF18, HFZ20]. Graphs in these datasets are small molecules where nodes are atoms and edges are chemical bonds. The tasks for these datasets are graph classification with chemical properties as labels, and we follow [HFZ20] to use ROC-AUC score as the evaluation metric.

We provide a synopsis of each downstream task dataset taken verbatim from [HLG20] below:

- **BBBP:** Blood-brain barrier penetration (membrane permeability).
- **Tox21:** Toxicity data on 12 biological targets, including nuclear receptors and stress response pathways.
- **ToxCast:** Toxicology measurements based on over 600 in vitro high-throughput

screenings.

- **SIDER:** Database of marketed drugs and adverse drug reactions (ADR), grouped into 27 system organ classes.
- **ClinTox:** Qualitative data classifying drugs approved by the FDA and those that have failed clinical trials for toxicity reasons.
- **MUV** Subset of PubChem BioAssay by applying a refined nearest neighbor analysis, designed for validation of virtual screening techniques.
- **HIV:** Experimentally measured abilities to inhibit HIV replication.
- **BACE:** Qualitative binding results for a set of inhibitors of human β-secretase 1.

Proteins. We consider the PROTEIN, ENZYME, and DD datasets containing proteins [MKB20]. Different from the chemical compounds, graphs in these datasets are macromolecules where nodes are secondary structure elements like helixes, and edges represent the spatial relationship between nodes. The task of ENZYME is to classify each graph into 6 different enzyme classes. The task of PROTEIN and DD is a binary classification that predicts whether a graph is an enzyme. We follow [HK20] to use the average classification accuracy as the evaluation metric. For further information on the protein datasets, please refer to [MKB20].

5.4.2 Model Configuration and Implementation

We use the Graph Isomorphism Network (GIN) as the GNN encoder in MICRO-Graph [XHL18]. We follow the same model hyperparameters from previous works [HLG20, SHV20], which are different from datasets to datasets, where we majorly follow the same setting as previous works for fair comparison [HLG20, YCS20, SHV20, HK20]. The GIN is trained with the Adam optimizer [KB17]. We run all experiments on a machine with 80 Intel(R) Xeon(R) E5-2698 v4 @ 2.20GHz CPUs, and a single NVIDIA V100 GPU with 16GB RAM. Our implementations are based on Python 3.8.10, PyTorch 1.10.0, and PyTorch-Geometric 1.7.1 [FL19].

Chemical Compounds Following [HLG20, YCS20], we use five convolutional layers with 300 hidden dimension (embedding dimension) for the GNN. We pre-train our model using Adam optimizer for 100 epochs, with batch size (number of graphs per batch) 512, and we fine-tune the pre-trained model for 100 epochs with batch size 32.

Proteins For proteins, we adopt the encoder and SVM evaluation implementation from [YCS20], and thus follow all their default hyperparameters for a fair comparison. The result we show in table 5.2 is based on GIN with 3 convolutional layers, 32 hidden dimension, and 128 batch size. However, in [SHV20, HK20], the reported results are based on a grid search over hyperparameters over number of convolutional layers from {2, 4, 8, 12}, number of training epochs from {10, 20, 40, 100}, batch size from {32, 64, 128, 256} as in [HK20].

5.4.3 Qualitative Evaluation

As we discussed in Section 5.3.2, motif embedding $\boldsymbol{m}_k$ can be interpreted by visualizing subgraph patterns with $\boldsymbol{s}$ close to $\boldsymbol{m}_k$. We thus evaluate the mined motifs by visualization and compare them to the ground truth motifs justified by the domain knowledge on real datasets.

Mined motifs on HIV **matches real functional groups.** We then visualize motifs we mined with MICRO-Graph $K = 100$ on the HIV dataset. We show ten example motifs ordered in increasing graph size in Figure 5.4. Even though our motifs are mined without any human annotation or prior knowledge, we find that the learned chemical compound motifs match the real-world functional groups. Motif 1 matches the propane [1]. Motif 2 matches the Amine [2]. Motif 3 matches the Carboxylic acid[3]. Motif 6 matches the benzene ring [4], and

[1] `https://en.wikipedia.org/wiki/Propane`

[2] `https://en.wikipedia.org/wiki/Amine`

[3] `https://en.wikipedia.org/wiki/Carboxylic_acid`

[4] `https://en.wikipedia.org/wiki/Benzene`

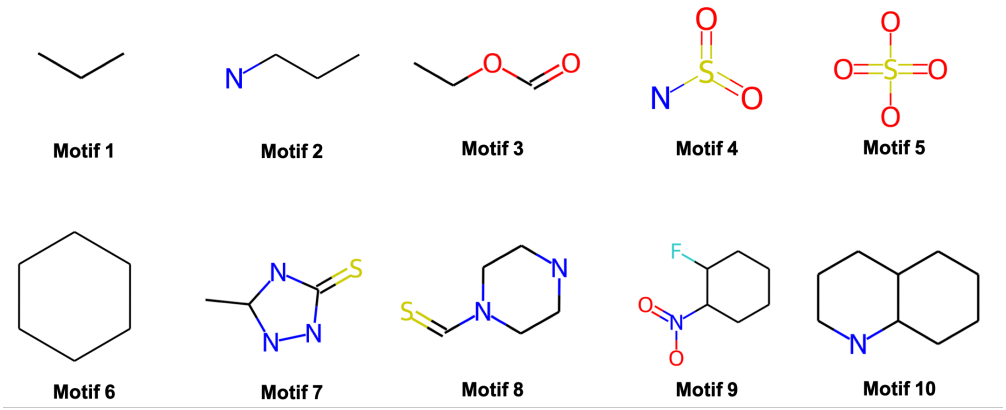


Figure 5.4: **Frequent motifs mined from the** HIV **chemical compound dataset.** Motifs are ordered with their sizes in increasing order. The mined motifs match real-world functional groups.

etc. This shows that our motif learning can really capture the semantic property of molecule subgraphs, which benefits model interpretability.

Mined motifs on HIV **are diverse.** Also, we show that the learned motif can capture different levels of properties, though we didn't add explicit hierarchical modeling. We see that the simple motifs tend to model some frequently occurring building blocks such as small propane with three carbon atoms in Motif 1, benzene ring with 6 carbon atoms in Motif 3, and the double carbon ring with 10 atoms in Motif 10. Such hierarchy shows the diversity of our mined motif, which makes the corresponding motif descriptors powerful graph features as shown in Section 5.4.4 via supervised downstream tasks.

Mined motifs on HIV **allow small variants while keeping the core part.** We visualize the top 3 subgraph patterns corresponding to Motif 7, 8, and 9 from HIV in Figure 5.5. We observe that these patterns allow small variants but keep the core part unchanged.

Mined Motifs on DD are visualized in Figure 5.6

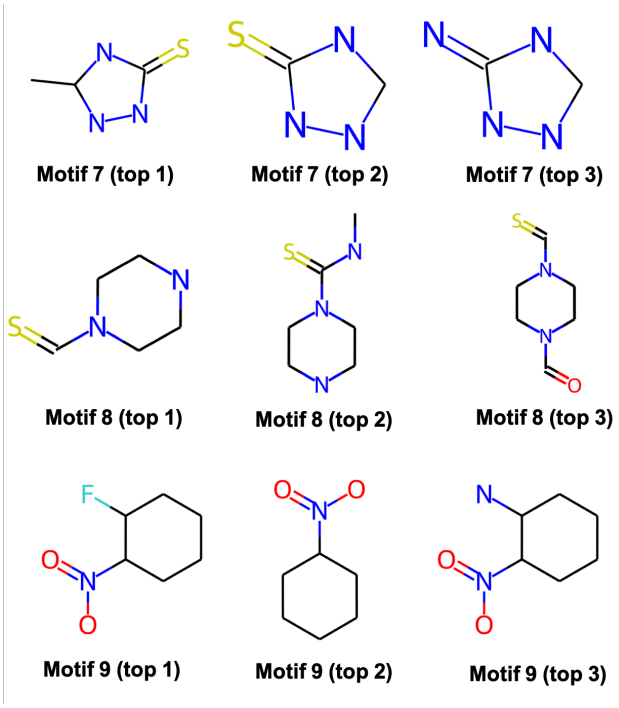


Figure 5.5: **Visualization of the top 3 most similar subgraphs corresponding to motifs.** Showing examples for Motifs 7, 8, and 9 in Figure 5.4. the embedding definition of motifs allows small structure variants while keeping the core part unchanged.

5.4.4 Quantitative Evaluation

The motif descriptors generated by MICRO-Graph can be used as graph-level features for downstream supervised graph prediction tasks. The task of each dataset is described in Section 5.4.1. We evaluate the motif descriptors by comparing them to graph-level features generated by other unsupervised learning methods on these tasks. We follow the standard evaluation paradigm used in the literature for these chemical compound and protein datasets. For each chemical compound dataset, we generate the graph-level features and evaluate them by training a one-layer NN on them. For protein datasets, we also generate the graph-level features but plug them into an SVM classifier following [SHV20]. For all datasets, we follow the standard train-valid-test split for each dataset and report the result on the test set for the best model selected on the validation set.

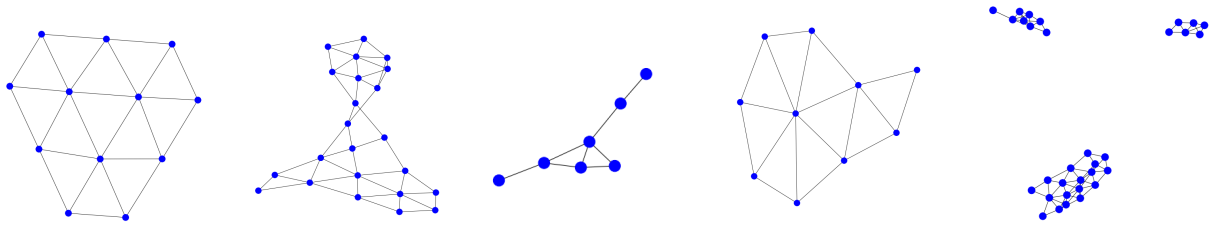


Figure 5.6: **Frequent motifs mined from the** DD **protein dataset.** Five frequent motifs learned by GStarXrepresented by their closest subgraphs.

Baselines. Following the previous literature, we use baselines [SHV20, HDW20, RBX20, YCS20] for the chemical compound datasets, and baselines [SHV20, HK20, YCS20] for protein datasets. We summarize these baselines below.

- InfoGraph [SHV20] maximizes the mutual information between the representations of the whole graphs and the representations of its substructures.
- GPT-GNN [HDW20] predicts masked edges and masked node attributes. The edge prediction makes node representations to be close when there are edges between them. The attribute prediction captures how node attributes are distributed over all graphs.
- GROVER [RBX20] first uses professional software, e.g. RDKit[Lan10], to extract motifs from the data and create a motif descriptor for each graph. Then it learns graph representations by predicting the motif descriptor.
- GraphCL [YCS20] performs contrastive learning and construct views with four types of augmentations methods including node dropping, edge perturbation, attribute masking, and subgraph sampling. In our experiments, we adopt the default setting, i.e., randomly choose two out of four methods to construct views.
- MVGRL [HK20] performs contrastive multi-view learning to learn graph level representations.

Evaluation Results The evaluation result of chemical compounds is illustrated in Table 5.1. We see that MICRO-Graph performs the best on 5/7 datasets and is the highest on

Table 5.1: **Performance comparison between** MICRO-Graph **and baselines on chemical compound datasets.** ROC-AUC of evaluating motif descriptors generated by MICRO-Graph and graph-level features generated by other unsupervised baselines on chemical compound property prediction datasets. We report the mean and std of 5 experiments. Highest results are in bold.

Methods	bace	bbbp	clintox	hiv	sider	tox21	toxcast	Average
InfoGraph	66.06 ± 0.82	75.34 ± 0.51	**75.71 ± 0.53**	61.45 ± 0.74	54.70 ± 0.24	63.95 ± 0.24	52.69 ± 0.07	64.27
GPT-GNN	66.43 ± 0.66	71.58 ± 0.54	66.78 ± 0.58	67.08 ± 0.36	54.67 ± 0.16	68.20 ± 0.14	57.06 ± 0.13	64.53
GROVER	65.67 ± 0.38	78.47 ± 0.36	53.19 ± 0.68	69.03 ± 0.23	54.94 ± 0.12	67.63 ± 0.13	57.28 ± 0.05	63.74
GraphCL	60.93 ± 1.72	76.91 ± 0.85	73.79 ± 2.13	70.52 ± 1.22	55.01 ± 1.43	**72.14 ± 0.78**	58.51 ± 0.39	66.83
MICRO-Graph	**70.83 ± 2.06**	**82.97 ± 1.53**	73.49 ± 2.16	**73.34 ± 1.27**	**57.32 ± 0.62**	71.82 ± 0.82	**59.46 ± 0.25**	**69.73**

Table 5.2: **Performance comparison between** MICRO-Graph **and baselines on protein datasets.** Accuracy of evaluating motif descriptors generated by MICRO-Graph and graph-level features generated by other unsupervised baselines on protein classification datasets. We report the mean and std of 5 experiments. Results with * are from the previous papers under the same experiment setting. Highest results are in bold.

Methods	ENZYMES	PROTEINS	DD	Average
InfoGraph	51.4 ± 0.8	74.4 ± 0.3*	72.9 ± 1.8*	66.2
GraphCL	55.5 ± 1.8	74.4 ± 0.5*	78.6 ± 0.4*	69.5
MVGRL	43.7 ± 4.6	72.2 ± 0.8	73.2 ± 0.6	63.0
MICRO-Graph	**58.1 ± 1.3**	**75.6 ± 0.5**	**79.5 ± 0.7**	**71.07**

average. The evaluation result of proteins is illustrated in Table 5.2. MICRO-Graph shows the best performance over the baselines on all three datasets. Both tables show the motif descriptors generated by MICRO-Graph have good discriminative power, which shows the motifs capture rich information about the graph.

5.4.5 Ablation Study and Analysis

We conduct ablation studies to analyze the influence of the cut loss $\mathcal{L}_{cut}$ and the number of motifs K in our model. The experiment setting follows Section 5.4.4.

Dropping $\mathcal{L}_{cut}$. The $\mathcal{L}_{cut}$ is a critical part of our learning objective to enhance motif structure coherence. We conduct our qualitative experiments on the chemical datasets without

Table 5.3: MICRO-Graph **ablation on $\mathcal{L}_{cut}$.** Average performance of evaluating motif descriptors generated by MICRO-Graph on chemical compound property prediction datasets and protein classification datasets with and without $\mathcal{L}_{cut}$. Experiment settings follow Section 5.4.4.

Methods	Compound Average	Protein Average
MICRO-Graph	69.73	71.1
MICRO-Graph w/o $\mathcal{L}_{cut}$	61.33	64.3

$\mathcal{L}_{cut}$ and keep other settings the same. The result is shown in Table 5.3, we see that removing $\mathcal{L}_{cut}$ causes significant performance drop. This study shows that the graph structure is significant in learning good graph representations. GNNs, although have been shown with the ability to capture structural information [CCV20], still have room for improvement. Using more expressive GNN architectures may alleviate such problems, which we leave as future work.

Number of Motifs. The number of motifs, K, is an important hyperparameter of MICRO-Graph. We conduct our qualitative experiments on the chemical datasets with 6 different K values, i.e. 10, 20, 50, 100, 200, and 300, and report the average ROC-AUC score for each K in Figure 5.7. We observe that as K gets larger from 10, the average performance increases until $K > 100$. Also, MICRO-Graph consistently outperforms other baseline methods as long as $K \geq 50$. We found the performance of an intermediate value 100 gives the best result and report it in Table 5.1.

5.5 Discussion

The existing graph motif mining methods are based on subgraph isomorphism tests and cannot mine motifs based on their semantic meanings. We thus propose MICRO-Graph to mined graph motifs as embedding vectors corresponding to semantically similar subgraph patterns. We show that MICRO-Graph can mine the ground truth motifs from synthetic

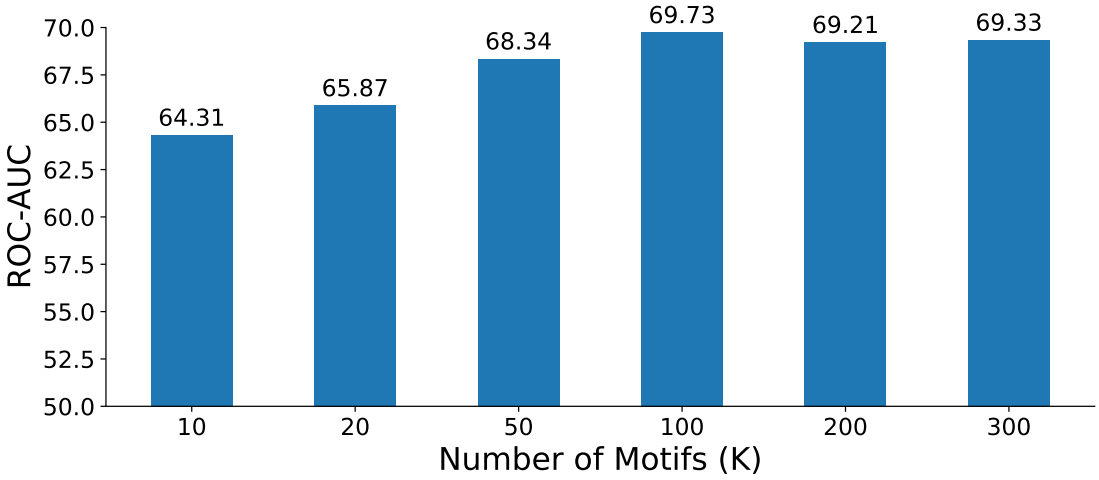


Figure 5.7: MICRO-Graph **Ablation on** K**.** Average performance of MICRO-Graph on the chemical datasets with different K. Experiment settings follow Section 5.4.4.

graphs, and motifs mined from chemical compound graphs match real-world functional groups. Evaluation of the motif descriptors generated by MICRO-Graph shows that they capture useful information of the graph and thus are good features for downstream supervised tasks. One limitation of our framework is the relationship between motifs is not explicitly modeled. We assume including a combination of motifs or hierarchy of motifs will further improve the mined motif quality and generate even more expressive motif descriptors, which we left as future work.

CHAPTER 6

Automated Molecular Concept Generation and Labeling

6.1 Introduction

AI has been a driving force behind several groundbreaking scientific discoveries, particularly in the domain of molecular science. A prime example is the utilization of deep learning by the MIT Jameel Clinic, leading to the identification of halicin – the first antibiotic discovered in three decades that is effective against a broad spectrum of 35 bacteria [SYS20]. In molecular science, deep learning models like GNNs have shown promising capabilities for learning complex atomic structures and making accurate predictions of molecular properties [WRF18]. However, a major challenge with such deep-learning-based models like GNNs is their "black boxes" nature and lack of explainability [YYG22]. Despite their high predictive performance, black-box models fail to provide insights into the underlying reasoning behind their predictions, making it difficult for scientists to interpret and intervene in the model's decision-making process, which hinders scientific understanding and limits the potential for knowledge discovery.

In contrast, concept-based models (CMs) [LNH09, KNT20, YKA20, WDG23] have emerged as a promising explainable AI (XAI) solution for scientific problems, whose explainability can provide valuable insights and increase the chance of new scientific discoveries. Unlike other black-box models, CMs first predict labels of human-interpretable concepts from the data and then use these concept labels as features to predict the actual task label. For example, in computer vision, CMs are applied for predicting the species of birds from their

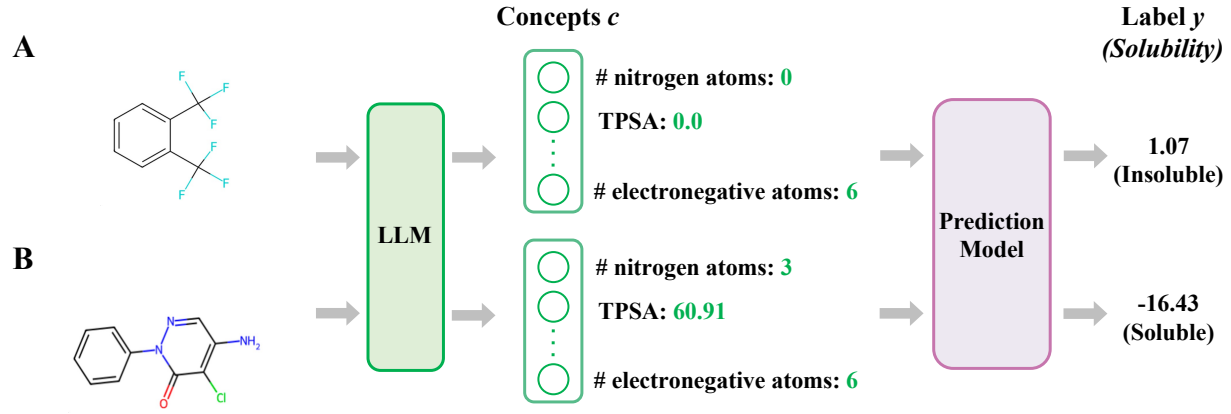


Figure 6.1: **The prediction process of molecule properties is greatly illuminated with** AutoMolCo**.** First, concepts are generated and labeled with an LLM. Afterward, a simple prediction model (e.g., linear regression) can be fitted to achieve explainable predictions. LLM-relevant pieces are highlighted in green, and specifically, green arrows indicate prompting.

images by identifying the relevant concepts like "wing color", which greatly increases the prediction explainability [KNT20]. In molecular science, CMs can be especially beneficial as they can break down the prediction of complex molecular properties into more interpretable molecular concepts like functional groups and molecular descriptors. For instance, Figure 6.1 demonstrates the prediction process of a CM for molecules A and B. The CM first elucidates the relevant molecular descriptors as concepts, such as the *# of nitrogen atoms* and the *topological polar surface area (TPSA)*, which are further employed to predict the molecule's solubility. This process not only yields a prediction but also provides a rationale that can be scrutinized and built upon. Like molecule B is predicted to be soluble based on its high TPSA, researchers can explore similar molecules with slight variations in these key concepts to hypothesize and test solubility-enhancing modifications. In contrast, black-box models, while potentially accurate, offer no such explanatory power.

Despite being a promising XAI approach, CMs have not been widely applied to molecular science problems, primarily due to their limitations in concept generation and labeling. Extant CM approaches either require predefined concepts and manual labels generated by domain experts [KNT20] or can only utilize simple and qualitative concepts that are insufficient

for molecular problems [ODN23]. Collecting such concepts and their labels is feasible and adequate for some computer vision tasks [KNT20, ODN23]. However, the desired concepts for molecules are sophisticated and nuanced, which require a depth of chemical knowledge and accurate quantitative labels. For example, TPSA in Figure 6.1 is a molecule descriptor relevant to the solubility prediction. It provides insight into a molecule's absorption and permeability characteristics, representing a quantitative concept pivotal for understanding the pharmacokinetics of potential drug compounds. Identifying and quantifying such a concept often demands domain knowledge and computational methods that extant CM approaches cannot achieve, thereby highlighting the challenge in effective CMs for molecular sciences.

In response to the effectiveness of CMs and the challenges of applying them to molecular science, we propose Automated Molecular Concept (AutoMolCo) generation and labeling. AutoMolCo leverages Large Language Models (LLMs) to generate molecular concepts that are predictive for the task and label these concepts for each molecule instance. AutoMolCo also repeats these procedures through iterative interactions with LLMs to refine concepts, enabling simple linear models on the refined concepts to outperform GNNs and LLM in-context learning (ICL) on several molecular benchmarks. The whole AutoMolCo framework is automated and does not require human knowledge inputs in either concept generation, labeling, or refinement, thus surpassing the limitations of extant CMs.

The motivation of AutoMolCo is founded on the idea that LLMs can be treated as extensive and integrated knowledge bases [PRL19, ALC22], and their effectiveness for solving molecular science problems has been benchmarked in [GGN23] through ICL. In our work, we exploit the potential of LLMs for XAI through CMs. For concept generation, we prompt LLMs with the task description and ask LLMs to suggest relevant concepts. For concept labeling, since it is particularly challenging, we explore three different approaches: direct LLM prompting, function code generation, and external tool calling. The concept generation and labeling process form the foundation of our framework. Afterward, we fit simple prediction models on the concepts to achieve an explainable CM. Moreover, we also do an iterative

refinement of the concepts by running feature selection algorithms of the prediction model and prompting LLMs again with the feature selection results. This allows LLMs to generate new concepts to replace the less useful ones from the previous iteration, which ensures the CM remains up-to-date with the most relevant concepts thus boosting its performance.

In this work, we first show that AutoMolCo can produce meaningful concepts and accurate labels, which lead to CMs with simple prediction models to achieve surprisingly good performance for molecular science problems. Then we perform a systematic study via experiments on MoleculeNet [WRF18] and High-Throughput Experimentation (HTE) [AEL18, RWB16] datasets to show the strengths and weaknesses of AutoMolCoand answer five research questions. In summary, our contribution includes:

- **Automated framework:** We propose AutoMolCo, which leverages LLMs for automated concept generation and labeling, eliminating the need for human domain knowledge and labor-intensive data collection, thereby streamlining the development of CMs.
- **Accuracy and explainability:** AutoMolCo produces meaningful molecular concepts that, when combined with simple prediction models for CMs, can achieve superior or comparable accuracy to powerful black-box models while providing greater explainability.
- **LLM-driven XAI for science:** Our work highlights the potential of LLMs in addressing complex problems in molecular science, introduces a novel perspective on CMs with LLMs, and paves the way for future research to further exploit the capabilities of LLMs in molecular science and beyond.

6.2 Related Work

Concept-based Models. A popular example of CMs is the Concept Bottleneck Model (CBM) [KNT20]. CBMs make predictions through an intermediate layer of human-specified concepts, such as "wing color" in specious classification from bird images. While CBMs offer more transparent predictions, they are often constrained by the predefined concepts

and label demands. Several follow-up CBMs with predefined concepts targeting different tasks [DLR18, YWG18, BHJ19, LFS19, CBR20]. One particular follow-up that is more relevant to this work, label-free CBM [ODN23], tries to bypass the need for predefined concepts and labels by generation and similarity matching. Unlike the other CBMs, this model identifies concepts using GPT-3 for concept set creation and CLIP-Dissect for matching text concepts with images. However, label-free CBM only focuses on vision tasks, and the generated concepts are often simple and qualitative, e.g., "yellow" is a concept, with similarity as the label, e.g., how similar a "lemon" image is to the text "yellow". Such concepts and labels are insufficient for molecules, where a depth of chemical knowledge is required for meaningful concepts and accurate qualitative labels.

Concept Learning in Graphs/Molecule. Several lines of research study concept learning from graph data, particularly for molecules. One line identifies graph motifs as concepts through counting or sampling [MSI02, Wer06] and builds GNNs on top of them [ZHS20, YG22]. However, motif identification cannot be comprehensive as it is NP-complete. Another line tries to use concept-based explanations for GNNs with human-in-the-loop, e.g., GCExplainer [MKS21]. Subsequent works have refined this idea with k-means clustering and similarity scoring algorithms to neuron-level grouping within activation layers [MBK22, XBG23]. These methods exemplify the attempt to extract and interpret salient features in graph data, yet they often face challenges in fully capturing the nuanced complexity of molecular structures.

LLMs for Molecular Science. GPT4Graph [GDL23] prompts LLMs to explain the format or to summarize a raw molecule graph input, where the graph is represented by the Graph Modelling Language (GML) [Him97] or Graph Markup Language (GraphML) [BEL13]. Graph-ToolFormer [Zha23] lets LLMs generate API calls to use external graph reasoning tools, which can be applied to molecule function reasoning problems. [GGN23] studies solving molecular problems with LLMs ICL. We show our AutoMolCo outperforms ICL and enjoys better explainability. Some other survey papers discussing the potential of LLMs for molecular

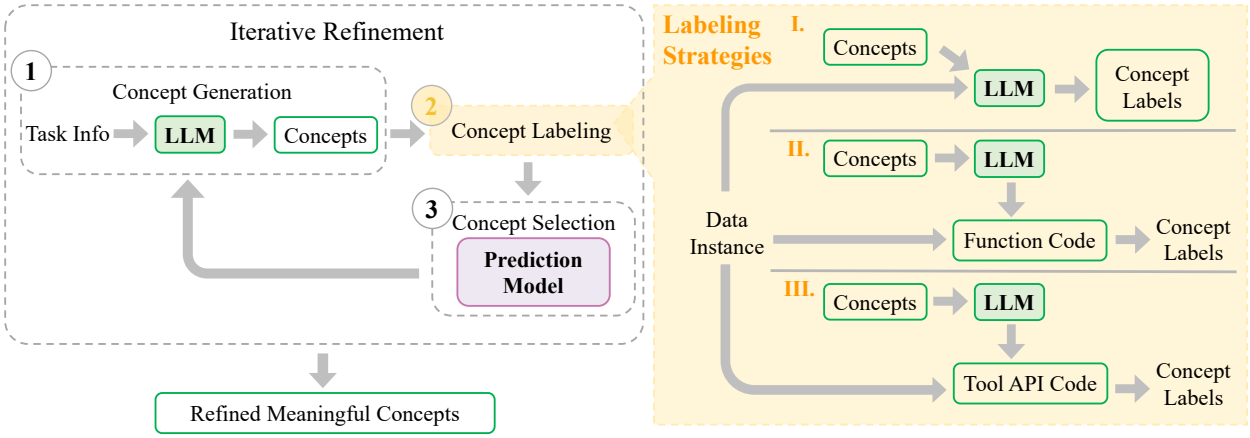


Figure 6.2: **The** AutoMolCo **framework.** Step 1: concept generation. Step 2: concept labeling with three different strategies. Step 3: fitting a prediction model and perform concept selection. These three steps are repeated for multiple iterations to achieve a refined list of meaningful concepts, where the selected concepts in each iteration are feedback to the LLM through prompting. LLM outputs are highlighted as green boxes.

science include: [ZWH23] from a scientific research perspective and [JLH23] from an LLM for graph perspective, and [YBC24] for fine-tuning LLMs.

6.3 AutoMolCo: Automated Molecular Concept Generation and Labeling

In this section, we describe AutoMolCo for concept generation, labeling, and refinement, where the concepts are used to build an explainable CM. Figure 6.2 depicts the three major steps of AutoMolCo: 1) concept generation, 2) concept labeling, and 3) CM fitting and concept selection.

6.3.1 Step 1: Concept Generation

Given a particular task on molecules, e.g., predicting the hydration-free energy of small molecules in water, the first step is to prompt LLMs to propose a diverse list of concepts that are potentially relevant to the task. This step is analogous to a brainstorming process.

Concepts range from counting-based ones, like # *nitrogen atoms*, to more complicated ones that require precise calculation, like TPSA. Without LLMs, coming up with meaningful concepts requires domain experts. The underlying intuition for concept generation is founded on the idea that LLMs can be treated as extensive and integrated knowledge bases. Their capacity to comprehend and output meaningful concepts is pivotal in this phase, yielding a wide spectrum of potentially relevant concepts for our analysis. The prompt for this step is shown in Figure 6.3 Step 1. The LLM-suggested concepts might be less relevant initially, but they will be refined later.

6.3.2 Step 2: Concept Labeling

Following the concept generation step, we then label the generated concepts for each data instance. Compared to human labeling, which requires domain knowledge and can be labor-intensive. Labeling with LLMs is streamlined to a process of interaction with a single LLM interface, which can be easily scaled and minimizes human error. This automation with LLMs is crucial for efficiently processing large volumes of data encountered in molecular studies. In this step, we consider three different labeling strategies to enhance labeling quality.

Labeling Strategy 1: Direct LLM prompting We prompt LLMs directly to assign each data instance numerical or categorical labels for the generated concepts from Step 1. Similar to concept generation, this strategy builds up on the idea that LLMs can be treated as integrated knowledge bases for retrieving useful information. For each data instance, we provide LLMs with the molecule names or SMILES strings. The prompt is shown in Figure 6.3 Step 2.

Labeling Strategy 2: Function code generation with LLMs Since LLMs are particularly skilled in code generation, we explore a second approach for concept labeling: generate functions in Python code for computing the concept labels. The function generation approach

has two advantages. Firstly, it greatly reduces the need for repeated LLM API calls. Only a single API call is required for each concept to obtain the function code, as opposed to making a separate call for each data instance in the direct label prompting case. Secondly, the generated functions can utilize preprocessed dataset features as function arguments, such as atom types in terms of node features and molecule structures in terms of adjacency matrix. These features provide more direct information beyond molecule names or SMILES strings. Leveraging these features, the LLM-generated functions can offer more nuanced and accurate concept labels, enhancing the effectiveness of AutoMolCo. The prompt is shown in Figure 6.4.

Labeling Strategy 3: External tool calling with LLMs We also utilize LLMs to call external tools like RDKit [Lan10] for labeling, combining the generation ability of LLMs with the reliability of specialized tools. This strategy enjoys the same efficiency advantage as the function generation approach, meaning it requires only a single API call of the LLM per concept to get the API code for calling the tool. Moreover, the use of labeling tools ensures that labels for all the tool-calculable concepts are accurate and reliable. One disadvantage of this strategy is that not all generated concepts are calculable by the external tool, in which case we can only turn to the first two strategies. The prompt is shown in Figure 6.5.

There are also some issues we overcame for each of these three labeling strategies. Below we discuss the challenges and our solutions in detail

Challenges and solutions for direct LLM prompting. For labeling with direct LLM prompting, we encountered two key issues: missing labels and unit inconsistency.

One issue we found for concept labeling with direct LLM prompting is that it is challenging to have LLMs generate some concept labels for certain molecules. For instance, LLMs identified *acid dissociation constant (pKa)* as a crucial concept for predicting water solubility. However, pKa is a quantity only apply to acids, and thus the model will output "Unknown" for the label. For the ESOL dataset, this results in a 13.03% missing rate for this concept. The

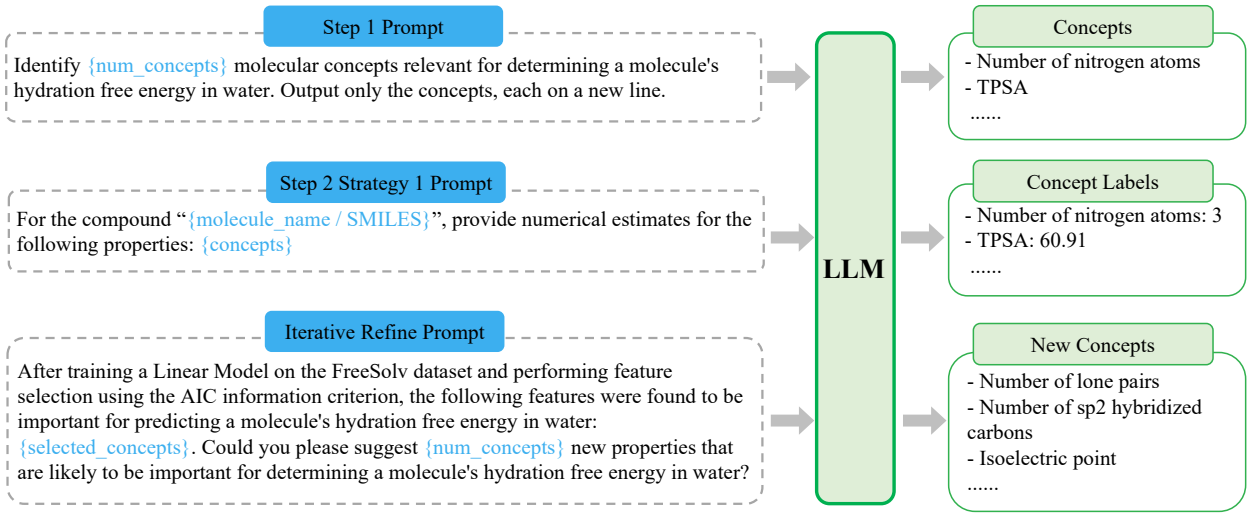


Figure 6.3: Prompts for concept generation and labeling on FreeSolv. Hyperparameters, molecule instance information, and re-used LLM responses from a previous step are in blue.

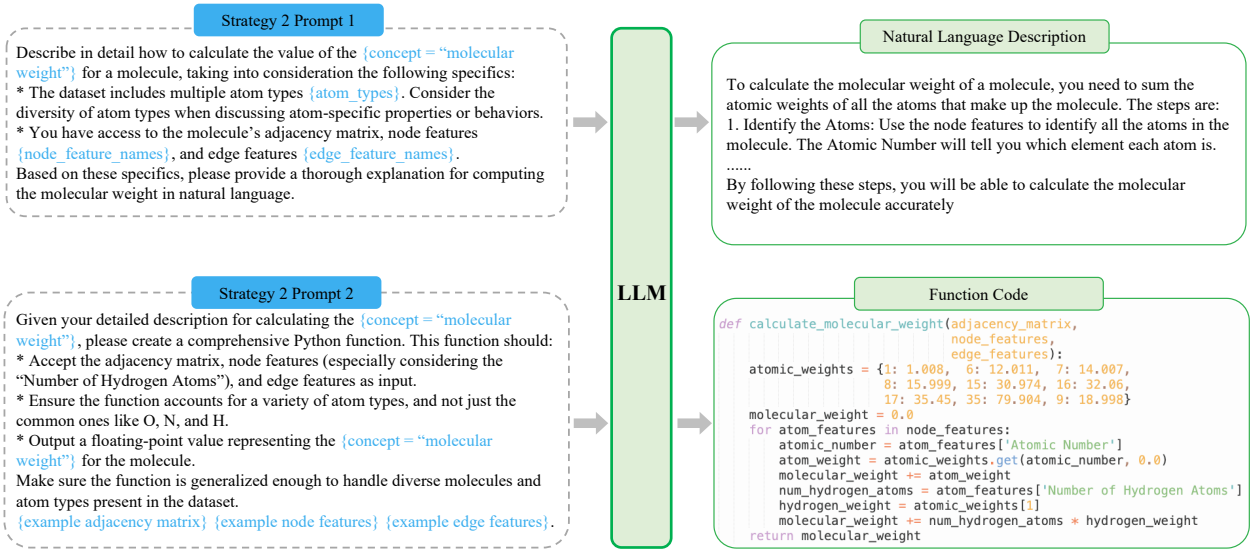


Figure 6.4: Prompts for generating concept labeling functions in Python code on FreeSolv.

missing-label issue underscores AutoMolCo's limitation in recognizing concept applicability across molecules. To mitigate this, we apply various imputation methods, including mean value imputation and domain-knowledge-driven imputation. For the latter, we set missing pKa labels to 100—significantly above water's pKa of 14—to denote weak or non-acidity, which enhances the CM's performance.

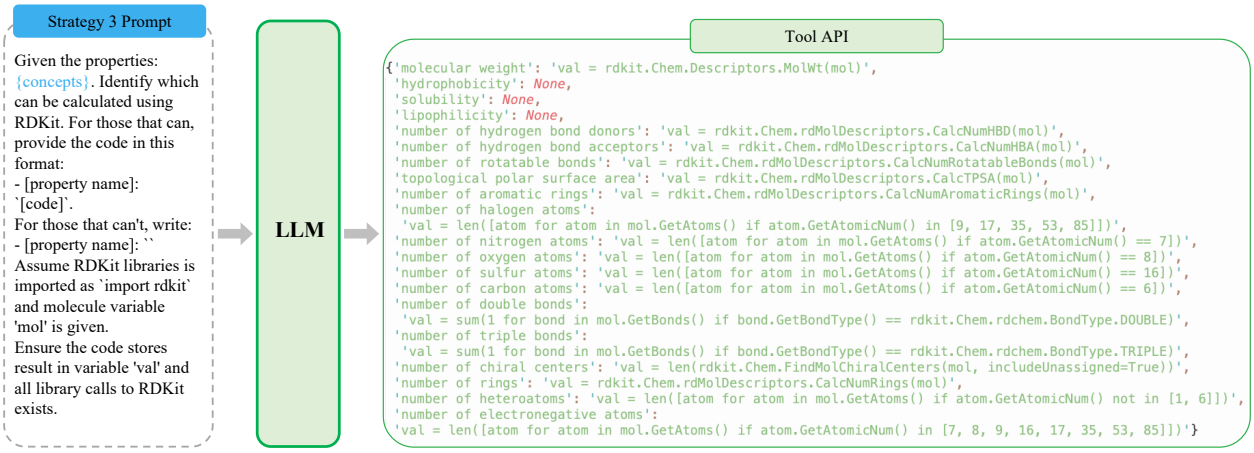


Figure 6.5: Prompts for calling the external tool RDKit to label concepts on FreeSolv.

We observe that for concepts with multiple possible units, labels generated by LLMs for these concepts may exhibit inconsistent units across different molecules. Take our experiments on the ESOL dataset as an example. The LLM suggest *melting point* as the relevant concept for predicting water solubility. When we call LLM API to generate values for *melting point*, it randomly chooses from Celsius ($°C$), Fahrenheit (F), or Kelvin (K) as *melting point*'s unit, which resulted in inconsistent scales for *melting point*'s labels. Similar issues also show up in other concepts, e.g. *molecular Volume* and *molecular surface area*. The inconsistency is due to the randomness in LLM context generation as we plug in different data instances into the prompt template. We first tried to fix the problem by specifying with LLM what unit should be used for each concept in the prompt, it was not every effective. Therefore, we added an intermediate step between Step 1 and Step 2, which lets LLM generate a concept-to-unit dictionary for each concept it proposed in Step 1. Then, we merged the resulting dictionary into Step 2's prompt, so LLM can generate labels with consistent units based on the unit specified in the dictionary.

Challenges and solutions for function code generation with LLMs. When generating labeling functions in Python code, we find it is non-trivial to prompt LLMs for executable functions with no errors. We made two efforts to increase the likelihood of producing

executable functions with LLMs. We first perform prompt engineering to clearly specify atom types, adjacency matrices, and node and edge features, which enhances the function quality. Through careful prompt engineering, most generated functions for simpler concepts become executable. However, functions for labeling complex concepts like "number of rings" are still unlikely to be error-free due to their intricate nature. We thus adopt a chain of thought (CoT) approach to generate functions. For the CoT prompt, we first ask the LLM to describe the function in natural language, which can best leverage the LLM's strength in generating natural language. Then, the CoT prompt asks the LLM to turn the natural language description of the function into Python code, which we found increases the likelihood of generating accurate and executable functions. An example of the CoT function-generation prompt is shown in Figure 6.4.

Challenges and solutions for external tool calling with LLMs. Given there are external tools for molecular science with API access, we prompt LLMs to generate code snippets for calling the tool API. We observe that LLMs are adept at obtaining callable APIs for a majority of our generated concepts from step 1, which we successfully employed to calculate the concept labels for each molecule in our dataset. The example prompts and generated API calls can be found in Figure 6.5. The drawback of this strategy is that the external tool cannot cover all the concepts generated by the LLM, especially for those measured concepts like melting point. For these cases, we turn to the first two strategies for labeling.

6.3.3 Step 3: CM Fitting and Concept Selection.

After getting the generated concepts and their labels, we utilize them to fit prediction models for the molecular task. Since the concept labels can be treated as tabular data, any model from the off-the-shelf ones in Scikit-learn [PVG11] to sophisticated deep learning models can be applied. However, we found that explainable models like linear models and decision

trees or simple two-layer MLPs are often sufficient for achieving competitive performance. We attribute the credit to the high-quality concepts and their labels. We will discuss our model choice and perform a systematic study of different prediction models in Section 6.4. While fitting the model, we also run feature selection methods like Akaike Information Criterion (AIC) [Aka73, Aka74] and Recursive Feature Elimination (RFE) [GWB02] to determine the useful concepts. Feature selection not only boosts the model performance but also leads to automated iterative refinement for identifying a list of the most useful concepts.

6.3.4 Iterative Concepts Refinement

After all three steps. We do an iterative refinement of the generated concepts by prompting LLMs again with the empirical performance of our prediction model and the concept selection results from Step 3. We include such information in an updated prompt to make LLMs generate new concepts to replace the less useful ones from the previous iteration. Using the empirical results as feedback, we ensure that our CM remains adaptable and up-to-date with the most relevant molecular concepts. Through this iterative refinement process, we guarantee that the model performance improves over iterations and prune the irrelevant concepts generated in previous iterations. The prompt for this step is shown in Figure 6.3.

6.4 Evaluation

6.4.1 Experiment Settings

Datasets We include four datasets from MoleculeNet [WRF18]. Two regression datasets, e.g., FreeSolv and ESOL, and two classification datasets, e.g., BBBP and BACE. FreeSolv provides hydration-free energy (which induces solubility) for 642 molecules, while ESOL contains water solubility data for 1128 organic small molecules. BBBP contains 2,039 molecules and assesses compounds' blood-brain barrier penetration. BACE has 1,513 molecules and

predicts β-secretase 1 inhibitors, relevant for Alzheimer's research. For all datasets, we employed the same scaffold splits as the Open Graph Benchmark (OGB) [HFZ20]. We also include two HTE datasets, e.g., Buchwald-Hartwig (BH) [AEL18] and Suzuki-Miyaura (SM) [RWB16] for chemical reaction yield prediction. BH provides the yields of the Buchwald-Hartwig reaction for 3957 molecules, and SM provides the yield of the Suzuki reaction for 5650 molecules. We employed the same data splits as used in [GGN23].

Metrics We follow the standard evaluation metrics for these datasets. For FreeSolv and ESOL, results are measured with Root Mean Square Error (RMSE). For BBBP and BACE, we mainly evaluate these datasets using AUC-ROC and report results in our main Table 6.1. Since [GGN23] evaluates BBBP and BACE with accuracy, we also report a comparison in accuracy in Table 6.7. For BH and SM, we evaluate with accuracy.

Baselines Our baselines include GNNs, LLM ICL, and GNN + CBM. Specifically, we use the GIN and GCN. For LLM ICL, we reference the findings from [GGN23] and use their prompts. For GNN + CBM, we use GIN and use GPT-3.5 Turbo for generating concept labels for CBM. For HTE datasets, we only consider LLM ICL baselines as graphs are not provided for the test set.

Models We employ GPT-3.5 Turbo as our primary LLMs for generating concepts and directly labeling. Additionally, we utilize GPT-4 for labeling strategy 2: function code generation, and strategy 3: external tool calling. For strategy 3, the LLM will create code snippets for invoking RDKit [Lan10]. We don't use GPT-4 for direct labeling due to the high cost of per-instance labeling. After collecting the concept labels, we explore four types of prediction models to cover a broad spectrum of tasks and performance levels. As a basic setting, we use linear models like linear regression and logistic regression, we also consider more advanced models including decision trees and 2-layer MLPs. We use off-the-shelf prediction models from sklearn [PVG11]. We do ablation on LLMs with Claude-2. Since we call LLMs through their APIs and the prediction models are light and off-the-shelf, there is no specially requirements, like GPUs, for our framework.

Concept Selection We employ AIC [Aka73, Aka74] for regression and RFE [GWB02] for classification. These selection methods are specifically applied to linear models. For multi-iteration performance with decision trees and MLPs, we use the selection results from the linear models.

6.4.2 AutoMolCo-induced CM Performance

In Table 6.1, we compare the performance of the AutoMolCo-induced CM to baselines. Compared to GNNs, our CM achieves better results on MoleculeNet regression tasks and HTE tasks and competitive results on MoleculeNet classification tasks. In comparison to the results presented by ICL, our models have demonstrated a substantial performance advantage on all tasks. Our best-performing model is the culmination of multiple iterations of refinement and a combination of labeling strategies. Specifically, the results presented in Table 6.1 are achieved using the following approaches: **1.** A combination of all three labeling strategies for concept labeling **2.** The optimal CMs from linear models, decision trees, and MLPs **3.** Concepts refinement over three iterations, as discussed in Section 6.4.6. An in-depth exposition of these techniques is discussed in detail in the RQs below through experiments on MoleculeNet datasets.

6.4.3 RQ1: Can AutoMolCo Generate Meaningful Molecular Concepts?

The effectiveness of the CM is built on meaningful concepts, which used to rely on domain experts [KNT20]. In this RQ1, we examine the concepts generated by AutoMolCo in each refinement iteration and consult with domain experts for their rationality. Figure 6.6 shows the concepts selected by a linear regression model for predicting solubility (FreeSolv), evolving from an initial broad set to a more focused and chemically relevant set. According to our consultation with domain experts, the eliminated concepts such as the counts of specific atoms and rotatable bonds, while informative, may not have significantly contributed to the model's

Table 6.1: **Performance comparison of the** AutoMolCo**-induced CM with baselines.** MoleculeNet regression tasks (FreeSolv and ESOL) are measured in RMSE (↓). MoleculeNet classification tasks (BBBP and BACE) are measured in AUC-ROC (↑). HTE datasets (BH and SM) are measured in accuracy (↑). Ours achieve better results on MoleculeNet regression and HTE tasks and competitive results on MoleculeNet classification tasks.

	FreeSolv (↓)	ESOL (↓)	BBBP (↑)	BACE (↑)	BH (↑)	SM (↑)
GIN	2.151	0.998	**69.710**	**73.460**	-	-
GCN	2.186	1.015	67.800	68.930	-	-
GIN + CBM	2.412	1.373	54.500	68.457	-	-
GPT-3.5 Turbo (zero-shot)	5.450	2.039	49.256	48.765	0.320	0.473
GPT-3.5 Turbo (4-shot)	4.852	1.161	51.580	41.871	0.640	0.630
GPT-3.5 Turbo (8-shot)	4.491	1.128	56.632	47.757	0.706	0.693
AutoMolCo-CM (ours)	**2.065**	**0.843**	65.278	70.744	**0.810**	**0.800**

predictive power or may have been correlated with other concepts. The retained concepts, like molecular weight, # hydrogen bond donors, and TPSA, are fundamental properties that influence molecular interaction and behavior in a solvent. For instance, hydrogen bond donors directly relate to a molecule's ability to engage in hydrogen bonding, a key interaction for solubility. The TPSA measures the molecule's surface that can engage in polar interactions, which is crucial for solubility in polar solvents [PL05]. Thus, the final selected concepts are indeed more chemically meaningful for the task and align with domain knowledge.

6.4.4 RQ2: Can AutoMolCo Assign Molecules Reasonable Concept Labels Using Each Strategy?

Accurate concept labels are another critical component of CM performance. In this RQ2, we evaluate AutoMolCo labeling results. we collect ground truth labels for concepts where labels are available, either through calculation (e.g., for molecular weights), or manual lookup (e.g., for melting points). We evaluate labels produced by our direct prompting strategy and function generation labeling strategy using the Pearson correlation coefficient (r) with the ground truth, due to the scale-invariant nature of the metric. The external tool calling

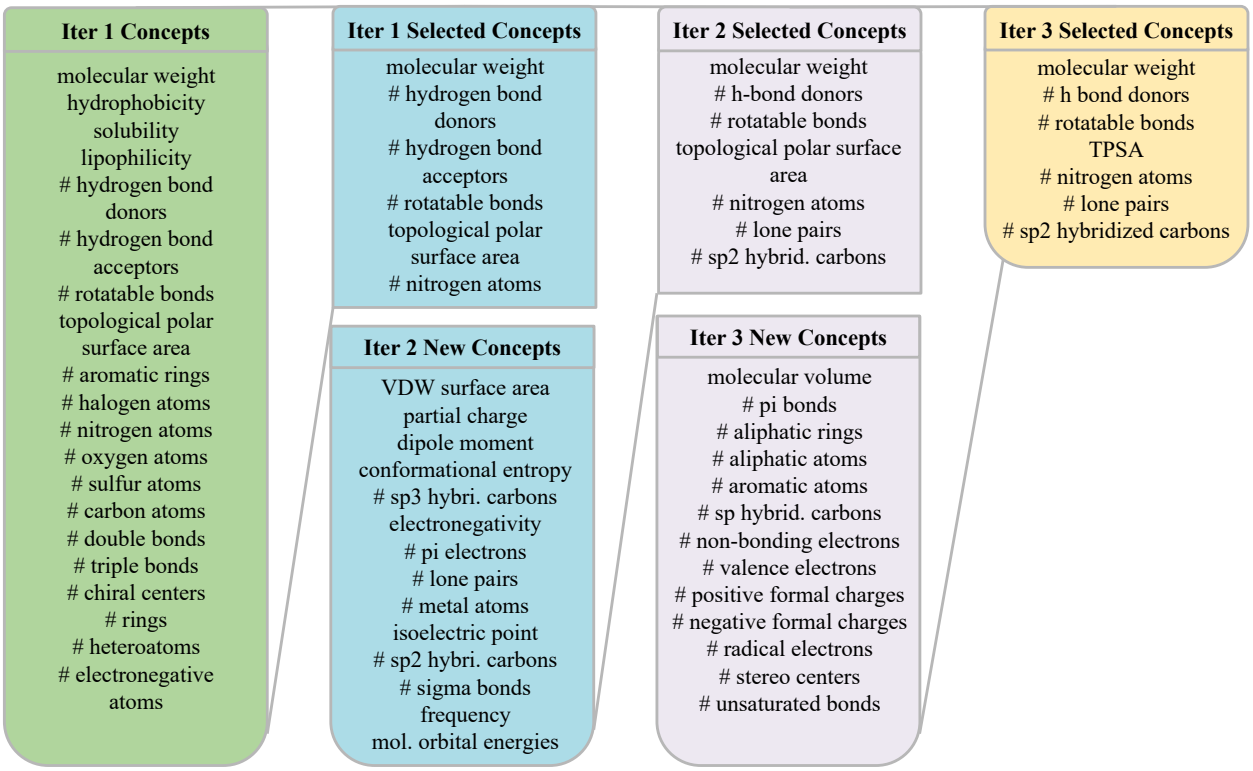


Figure 6.6: RQ1: Concepts selected by AutoMolCo in three refinement iterations on FreeSolv.

strategy is excluded from this evaluation as tools will always provide correct labels, and the downside of this strategy is that not all the concepts are tool-calculable (e.g., melting points). Results in Table 6.2 show strong correlations can be achieved on most datasets with AutoMolCo labeling. Nonetheless, variations in correlation underscore the potential for method improvement. In addition to this benchmarking effort, we also discuss several challenges we encountered and overcame in the labeling step, including imputation for missing values, dictionary for unit inconsistency, and Chain-of-Thoughts (CoT) prompts for syntax errors in function code.

Table 6.2: RQ2: Percentage of concepts with high correlations (r score ≥ 0.7) with the ground-truth.

Labeling Strategy	LLM	Molecule Format	FreeSolv	ESOL	BBBP	BACE
Str-1 Direct Prompt	GPT-3.5	Name	0.82	0.63	0.06	-
Str-1 Direct Prompt	GPT-3.5	SMILES	0.82	0.75	0.69	0.22
Str-2 Function	GPT-4	-	1.00	0.79	0.69	0.67

Table 6.3: RQ3: AutoMolCo-induced CM performance with different labeling strategies and prediction models.

Labeling Strategy	Prediction Model	FreeSolv(↓)	ESOL (↓)	BBBP (↑)	BACE (↑)
Str-1 Direct Prompt	Linear/Logistic	2.685	1.250	52.836	56.894
Str-1 Direct Prompt	Decision Tree	2.791	1.272	56.887	**68.632**
Str-1 Direct Prompt	MLP	2.338	1.194	51.794	60.059
Str-2 Function	Linear/Logistic	3.284	1.254	55.671	56.624
Str-2 Function	Decision Tree	2.569	1.238	54.167	55.573
Str-2 Function	MLP	2.805	1.034	**58.738**	56.894
Str-3 Tool	Linear/Logistic	3.142	1.011	57.350	63.154
Str-3 Tool	Decision Tree	3.750	1.027	55.903	65.658
Str-3 Tool	MLP	**1.981**	**0.911**	58.449	60.772

6.4.5 RQ3: Can AutoMolCo Produced Concepts and Labels Be Utilized To Build an Effective CM?

In RQ1, we have verified that the generated concepts are meaningful according to domain experts. In RQ2, we have shown that concept labels are relatively accurately assigned after properly handling potential issues like missing labels and unit inconsistency. In this RQ3, we compare the performance of the AutoMolCo-induced CMs when different predictions models and labeling strategies are adopted. Results in Table 6.3 show that AutoMolCo can give reasonable performance even with the most basic direct prompting labeling strategy and the simplest linear model. The good performance of different prediction models demonstrates the quality of the concepts and the effectiveness of AutoMolCo.

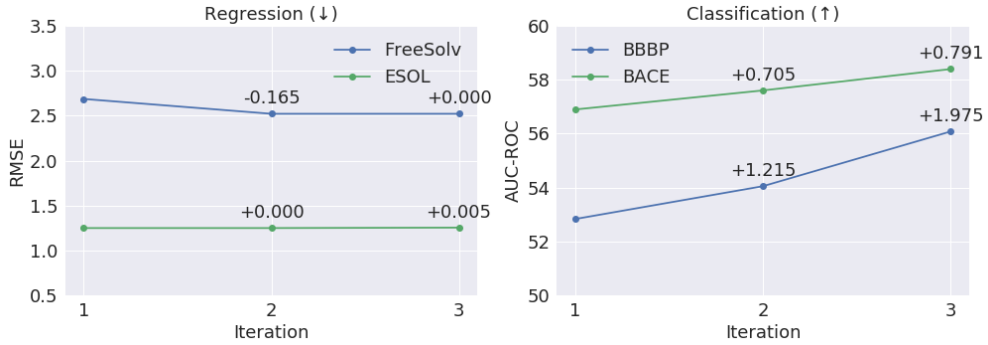


Figure 6.7: RQ4: Iterative refinement improves CM performance for classification tasks.

6.4.6 RQ4: Does Iterative Refinement Boost The Performance of AutoMolCo-induced CM?

As one of the most important designs of our AutoMolCo framework, concept refinement helps to identify meaningful important concepts through iterative interactions with LLMs. The concept relevance has been shown to improve in RQ1, but that does not necessarily mean CM performance will also improve. In this RQ4, we run AutoMolCo with three iterative concept refinements on the MoleculeNet datasets with linear prediction models. We show the results in Figure 6.7, and we observe that the CM prediction performance indeed improved through concept refinement, especially for classification tasks. The improvement for regression tasks is marginal, partially because the performance is already good for regression.

6.4.7 RQ5: Does The AutoMolCo-induced CM Facilitate Explainable Molecular Science?

One of the key advantages of CMs over black-box models is their explainability. In this section, we evaluate this aspect of AutoMolCo-induced CMs through three experiments on all three types of prediction models: coefficient interpretation of linear models, split interpretation of decision trees, and concept label intervention of MLPs.

Coefficient Interpretation of Linear Models Using linear model in the AutoMolCo-induced CM offers excellent explainability through direct interpretation of the model coefficients. We plot the coefficients of the linear model on FreeSolv for predicting hydration free energy in Figure 6.9 (a), highlighting three significant concepts: *# hydrogen bond donors*, *TPSA*, and *# rotatable bonds*. According to domain experts, the # hydrogen bond donors relates to a molecule's ability in hydrogen bonding, reflecting its potential to interact with solvents and other molecules. Therefore, its increment typically leads to a more favorable (more negative) hydration free energy [CP15]. TPSA quantifies the surface area of a molecule that can engage in polar interactions, providing insights into a molecule's permeability characteristics. Thus, higher TPSA also leads to more favorable (more negative) hydration free energy [PL05]. Conversely, the # rotatable bonds positively correlated with hydration free energy. More rotatable bonds increase molecular flexibility, allowing the molecule to adopt conformations that enhance interactions with water molecules. This increased flexibility can lead to less favorable hydration free energy (less negative), as it reduces the stability of the solvation shell around the molecule [GC08]. Our linear model interpretation aligns with domain knowledge without requiring any human knowledge input into the model.

We show the linear model on the BBBP dataset. We focused on the top positive coefficient lipophilicity (logP) and the top negative coefficient hydrogen bond acceptors shown in Figure 6.8. Notably, logP has a coefficient of 1.97, and hydrogen bond acceptors has a coefficient of -4.36. These findings align with domain knowledge, as higher logP enhances a molecule's ability to cross lipid-rich biological membranes. Conversely, a lower number of hydrogen bond acceptors generally enhances a molecule's permeability through the BBB. These findings validate the CM's alignment with established biochemical principles, demonstrating its potential utility in predictive modeling for molecular properties.

Splits Interpretation of Decision Trees Complementing to the coefficients of the linear model, decision tree enhances the understanding of model's decision process. In Figure 6.9

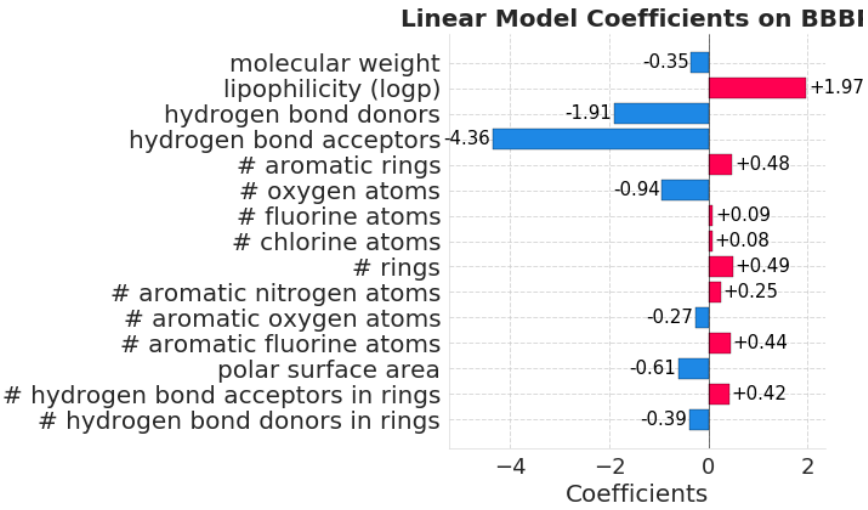


Figure 6.8: RQ5: Coefficients of the logistic regression model on BBBP with concepts refined by AutoMolCo after three iterations.

(b), we show the 3-layer decision tree for BBBP dataset. In the first two layers, the model uses TPSA to categorize the molecules into four categories, where molecules with TPSA less than 26.99 are likely to penetrate the BBB while molecules with TPSA greater than 176.22 are rarely penetrative. The decision tree further differentiates molecules with TPSA between 26.99 and 107.1 by whether or not it contains a hydrogen bond in its ring structure, where molecules without this property are more likely to penetrate the BBB. On the other hand, the model splits molecules with TPSA between 107.1 and 176.22 using the number of carbon atoms, illustrating that molecules containing more than 23 carbon atoms are very likely to be penetrative. Figure A.5 shows the impurity details of the decision tree.

Concept Label Intervention Besides analyzing interpretable prediction models like linear models and decision trees, we also conduct a case study of concept label interventions with MLPs. Our goal is to identify molecules with similar concept labels except for the one we intend to intervene on (e.g., similar molecular weights, # aromatic rings, etc., except logP) but different task labels (e.g., soluble vs. insoluble). Two examples we identify from the ESOL dataset are: *Diphenylamine (N(c1ccccc1)c2ccccc2)* and *RTI*

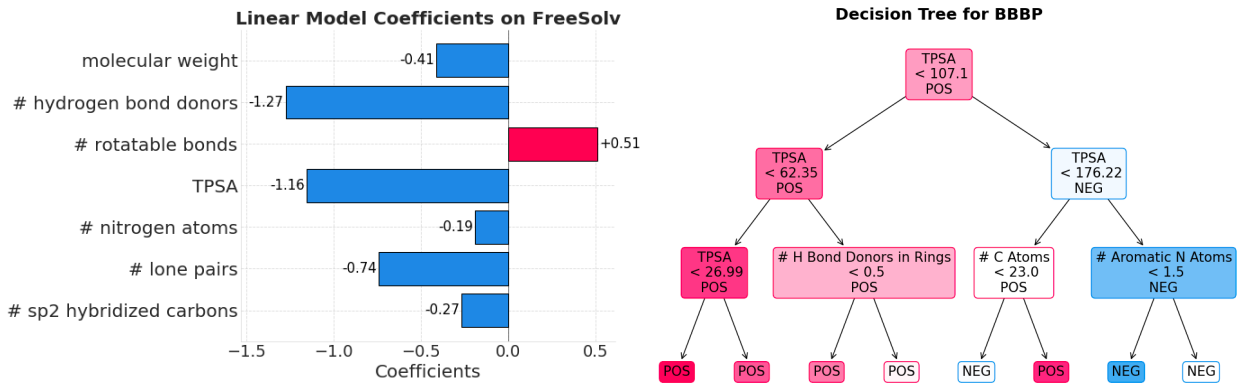


Figure 6.9: **RQ5: Coefficients of the linear regression model and the decision tree on FreeSolv.** Both with concepts refined by AutoMolCo after three iterations

17 (CCN2c1ccccc1N(C)C(=S)c3cccnc23). After three iterations of refinement these two molecules have the same labels for three out of the four remaining concepts, except for their logP labels, which differ by 0.275 (standardized to have mean 0 and standard deviation 1). Diphenylamine is predicted to be insoluble (-3.648), whereas RTI 17 is predicted to be soluble (-4.079), based on a conventional solubility threshold of -4 [SKE19]. These predictions proved to be quite accurate, with the ground truth solubility of these two molecules being -3.857 and -4.227, respectively. By intervening on diphenylamine's logP value (0.209) to match RTI 17's logP value (0.484) through interpolation, we observe a linear change in solubility. This study highlights the significant impact of logP on solubility predictions, which is consistent with expert conclusions [LLD97, Avd12], providing insights beyond traditional black-box models. We show the intervention plot in 6.10.

6.4.8 Ablation Studies

We conduct ablation studies of AutoMolCoto test its performance with different LLMs, molecule input formats, and combinations of labeling strategies. We found that AutoMolCo can perform consistently with different LLMs and is robust to molecule input formats. Also, properly combining the labeling strategies can enhance model performance.

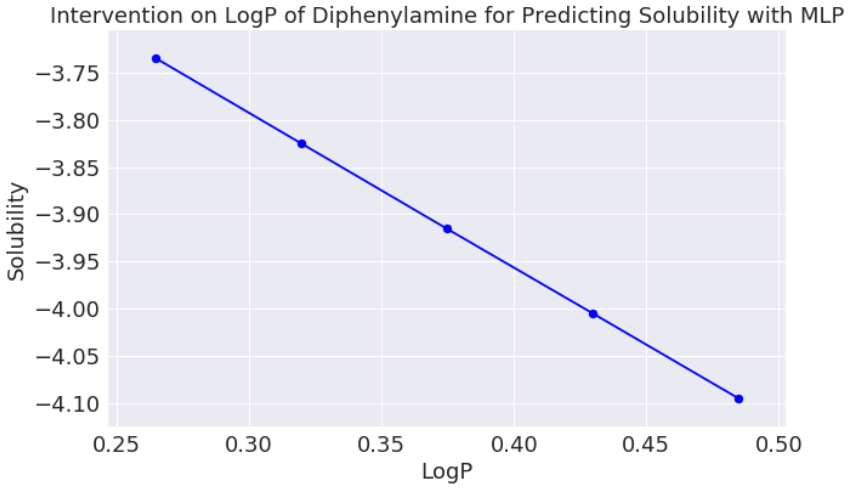


Figure 6.10: Intervention on logP of diphenylamine for predicting solubility with MLP

Table 6.4: AutoMolCoablation on LLMs (GPT-3.5 vs. Claude-2).

LLM	FreeSolv(↓)	ESOL (↓)	BBBP (↑)	BACE (↑)
GPT-3.5	2.685	1.250	52.84	56.89
Claude-2	2.804	1.327	52.78	56.11

Different LLMs We study the performance of AutoMolCo with different LLMs. Table 6.4 compares the performance of GPT-3.5 Turbo and Claude-2 using the direct LLM prompting labeling strategy with linear prediction models. While both GPT-3.5 Turbo and Claude-2 exhibit slightly inferior performance compared to GNNs across four datasets, they maintain competitive results, emphasizing simplicity and interpretability. Specifically, Claude-2 underperforms GPT-3.5 Turbo after first iteration, potentially due to its less consistent and accurate response. This inconsistency, partly attributed to more frequent issues with missing values and unit inconsistencies observed in Claude-2, suggests GPT-3.5 Turbo's superior ability to generate reliable ground truth knowledge. Additionally, GPT-3.5 Turbo's better prompt comprehension and domain knowledge in chemistry might contribute to its enhanced performance in predicting target concepts.

Table 6.5: AutoMolCoablation on input formats (SMILES strings vs. molecule names).

	FreeSolv (↓)		ESOL (↓)		BBBP (↑)		BACE (↑)
Input Format	SMILES	Names	SMILES	Names	SMILES	Names	SMILES
GPT-3.5 iter 1	2.854	2.685	1.401	1.250	53.88	52.84	56.89
GPT-3.5 iter 2	2.662	2.520	1.262	1.250	56.08	54.05	57.60
GPT-3.5 iter 3	2.763	2.520	1.262	1.255	60.41	56.08	58.38

Direct LLM prompting with molecule names vs. with SMILES strings Building on insights from [GGN23] regarding LLMs' challenges with long molecular representations, we examined LLM's capability in labeling concepts using SMILES strings and molecule names across our datasets. Our findings indicate that LLMs perform reasonably well in identifying basic concepts like molecular weight and atom counts using either molecule names or SMILES strings, with a strong correlation to ground truth labels ($r > 0.9$). However, LLMs struggle with complex concepts requiring detailed structural knowledge, such as the *number of chiral centers*. Moreover, our analysis reveals a notable decline in LLM's performance with molecule names in larger datasets like BBBP, suggesting LLM's familiarity with common molecular names improves its performance on smaller datasets, but this advantage diminishes with less familiar names in larger datasets. In contrast, the structural specificity of SMILES strings maintains more consistent performance across dataset sizes, highlighting their utility in representing unique molecular concepts. Furthermore, we compared the performance difference between the two representations over 3 iterations. As demonstrated in Table 6.5, the model performance using SMILES strings matched the model performance using molecule names on most datasets, but a notable improvement in performance with SMILES strings is observed on the BBBP dataset.

We also present a comparative analysis of the quality of concept labels generated using molecular names vs. SMILES strings. The comparison is visualized through a series of heatmaps, as illustrated in Figure 6.11.

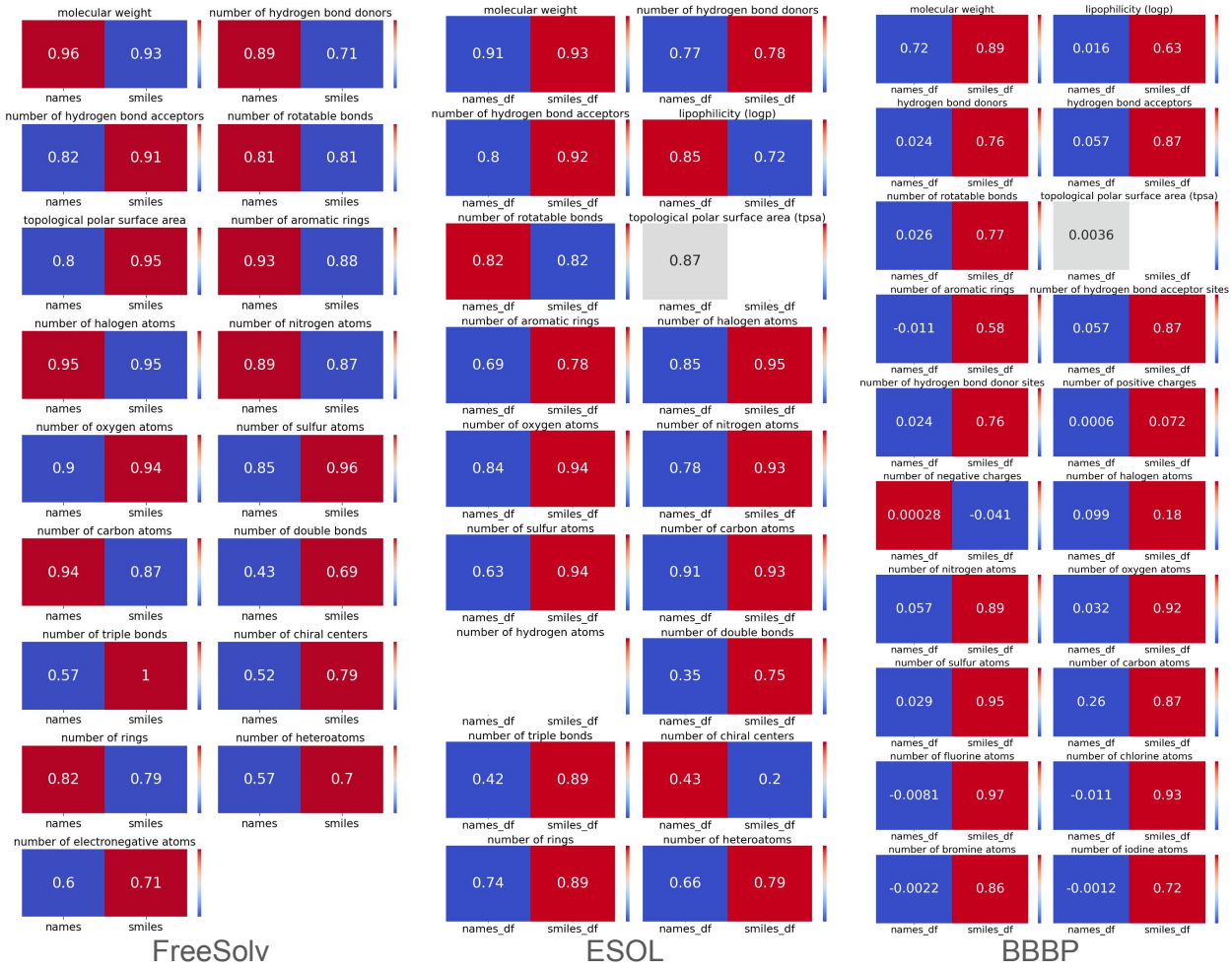


Figure 6.11: **Correlation (r score) between the ground truth labels and concept labels generated using molecule names or SMILES strings.** Red indicates a higher correlation.

Combine different labeling strategies The AutoMolCo framework includes three labeling strategies and allows easy extension to new ones. We consider combinations of the labeling strategies and study their impact on model performance, where we adopt a simple priority heuristic where strategy 3 > strategy 2 = strategy 1. Specifically, whenever the external tool is available for a suggested concept, we get the accurate concept labels by calling it. Otherwise, two concept labels are derived from both direct prompting and function code generation, and they are both considered in step 3 for the selection. This combined-strategy

Table 6.6: AutoMolCowith combined labeling strategies.

Labeling Strategy	Model	FreeSolv(↓)	ESOL (↓)	BBBP (↑)	BACE (↑)
Direct Prompt	Linear/Logistic	2.685	1.250	52.836	56.894
Direct Prompt	Tree Model	2.791	1.272	56.887	**68.632**
Direct Prompt	MLP	2.338	1.194	51.794	60.059
Direct Prompt + Function	Linear/Logistic	2.697	1.254	56.134	56.712
Direct Prompt + Function	Tree Model	2.540	1.364	55.150	59.998
Direct Prompt + Function	MLP	2.211	0.971	57.697	65.797
Direct Prompt + Function + Tool	Linear/Logistic	3.002	1.136	55.845	64.032
Direct Prompt + Function + Tool	Tree Model	3.752	1.107	56.250	65.658
Direct Prompt + Function + Tool	MLP	**2.122**	**0.791**	**58.391**	62.624

Table 6.7: **Performance comparison of the best** AutoMolCo**-induced CM vs. LLM ICL in accuracy on BBBP and BACE** (↑). (GPT results are taken from [GGN23]).

	BBBP (↑)	BACE (↑)
GPT-4 (zero-shot)	0.476	0.499
GPT-4 (Scaffold, k= 8)	0.614	0.679
GPT-3.5 (Scaffold, k= 8)	0.463	0.496
AutoMolCo	**0.657**	**0.704**

labeling turns out to outperform most of the standalone strategies as shown in Table 6.6. These findings demonstrate that the three labeling strategies have their own strengths and weaknesses for different concepts, and they can complement each other to maximize the model performance. We leave the exploration of new strategies and more sophisticated strategy combinations as future work.

Accuracy comparison with LLM ICL on BBBP and BACE In our main experiments, we follow the standard and widely-used evaluation metrics for all datasets. For classification tasks on BBBP and BACE, we mainly evaluate these datasets using the AUC-ROC metric and report results in our main Table 6.1. Since [GGN23] provides the ICL prompts but evaluates BBBP and BACE with accuracy, we also report a comparison in accuracy in Table 6.7 for a fair comparison.

Table 6.8: **Performance comparison of the** AutoMolCo**-induced CM with different prediction models vs. LLM ICL in accuracy on BH and SM** (↑). (GPT results are taken from [GGN23]).

	BH (↑)	SM (↑)
GPT-4 (random, k = 8)	0.800	0.764
GPT-3.5 + logistic (200 samples)	0.800	0.770
GPT-3.5 + MLP (200 samples)	0.800	0.780
GPT-3.5 + logistic (500 samples)	0.790	0.780
GPT-3.5 + MLP (500 samples)	**0.810**	**0.800**

More results on Buchwald-Hartwig and Suzuki-Miyaura The GPT ICL performance from [GGN23] are measured on 100 data samples. To compare AutoMolCo's performance with their numbers, for each dataset we picked the best model performance from the logistic regression models or MLP models trained on either 200 or 500 sampled training data. The performance details are presented in Table 6.8 with best performance reported in Table 6.1.

6.5 Discussion

We propose the AutoMolCo framework that automates the generation and labeling of molecular concepts, overcoming challenges of existing CMs and enhancing explainability through iterative refinement of useful concepts. We demonstrate that, for molecular property prediction tasks, simple linear models on our generated concepts can perform competitively or even better than GNNs and LLM ICL. Our work paves the way for future research to further exploit the capabilities of LLMs for XAI in molecular science and beyond.

The framework also has limitations that can inspire future work. One limitation is that LLMs can sometimes hallucinate, generating inaccurate concepts and labels. This affects the accuracy and reliability of the outputs. Using more powerful LLMs could improve this issue. Another limitation is that the evaluation of the generated concepts and labels often requires validation by human experts, introducing subjectivity and dependency on domain knowledge.

Developing automated evaluation methods is another potential direction for improvement.

CHAPTER 7

Conclusion

The pursuit of XAI for graph data is a critical endeavor that has far-reaching implications for the responsible and ethical deployment of AI systems across numerous domains. Throughout this thesis, I have explored three complementary perspectives – model-oriented, user-oriented, and data-oriented – to advance the explainability of AI models on graph data.

The model-oriented approaches apply post hoc techniques to explain the decision-making processes of black-box GNNs. These methods demonstrate how AI models reason about and infer on graph data. They pave the way for a deeper understanding of AI models and show the potential to debug and enhance model design. Complementing the model-oriented perspective, the user-oriented techniques developed in this thesis have focused on providing intuitive visualizations and natural language explanations tailored to human understanding. By bridging the gap between complex AI models and end-users, who likely have no technical background, these approaches foster user trust and satisfaction in the deployed AI systems for graph-based applications. Furthermore, the data-oriented perspective covers both post hoc explanations and inherently explainable models. These approaches leverage the inherent structure and properties of graph data to enhance explainability, with the help of mask learning, motif embedding learning, and LLM-based concept learning. The insight extracted through explanations from scientific data like material atomic graphs and molecule graphs can not only match domain knowledge but potentially lead to new scientific discoveries. The synergistic combination of these three perspectives has yielded a comprehensive framework for explainable AI for graph data, addressing the critical challenges of interpretability,

accountability, and alignment.

As AI continues to evolve and make new breakthroughs, the demand for explainable and trustworthy AI systems will intensify. The methods and insights derived from this research lay a foundation for future endeavors in XAI for graphs, fostering a deeper understanding of AI decision-making processes and promoting the responsible deployment of these powerful models in graph-based applications that shape our world.

APPENDIX A

Appendices

A.1 Chapter 2 Appendices

A.1.1 The Myerson Value, C-Shapley Value, and L-hop Graph Cutoff

A.1.1.1 The Myerson value

In the study of cooperative games, [Mye77] proposed to characterize the cooperation possibilities between players using a graph structure $\mathcal{G}$, which leads to the communication structure introduced in Section 2.2.2 and the Myerson value as a solution for this special type of games $(N, v, \mathcal{G})$. The Myerson value is closely related to the Shapley value. In fact, it is the Shapley value on a transformed game where players are partitioned by the graph. We now formally introduce the partition and the transformed game.

Definition A.1.1 (**Partition**)**.** Given a set of players N and a graph $\mathcal{G}$. For any coalition $S \subseteq N$, the partition of S is denoted by $S/\mathcal{G}$ and defined by

$$S/\mathcal{G} = \{\{i | i \text{ and } j \text{ are connected in S by } \mathcal{G}\} | j \in S\}$$

and a member of the set $S/\mathcal{G}$ is called a component of S.

Definition A.1.2 (**Transformed Game**)**.** Given a game $(N, v, \mathcal{G})$, we can transform it to a

new game $v/\mathcal{G}$ such that for all $S \subseteq N$

$$(v/\mathcal{G})(S) = \sum_{T \in S/\mathcal{G}} v(T)$$

Intuitively, given a coalition S, the transformed game treats each connected component of S as independent, evaluates them separately, and sums their payoff as the payoff of S.

The Shapley value has an axiomatic characterization that uniquely determines it. Likewise, the Myerson value was proposed to be a unique solution that satisfies the *component efficiency* and the *fairness* property defined below.

Property A.1.3 (**Component Efficiency**)**.** For a game $(N, v, \mathcal{G})$ and any connected component $S \in N/\mathcal{G}$, a solution is component efficient if

$$\sum_{i \in S} \phi_i(N, v, \mathcal{G}) = v(S)$$

Property A.1.4 (**Fairness**)**.** For a game $(N, v, \mathcal{G})$ and any edge (i, j) in $\mathcal{G}$, let $\tilde{\mathcal{G}}$ be $\mathcal{G}$ with the edge (i, j) removed, a solution is fair if

$$\phi_i(N, v, \mathcal{G}) - \phi_i(N, v, \tilde{\mathcal{G}}) = \phi_j(N, v, \mathcal{G}) - \phi_j(N, v, \tilde{\mathcal{G}})$$

The component efficiency property is an extension of the regular efficiency property to games with a communication structure. It requires efficiency to hold for each disconnected piece because these pieces are assumed as independent from each other. The fairness property states that if breaking an edge (i, j) changes the value of player i, then the value of player j should be changed by the same amount.

Theorem A.1.5 (**Myerson Value**)**.** *There exists a unique solution ϕ of game $(N, v, \mathcal{G})$ satisfying component efficiency and fairness. With $\tilde{\phi}$ represents the Shapley value, the solution*

is given by the formula

$$\phi(N, v, \mathcal{G}) = \tilde{\phi}(N, (v/\mathcal{G}))$$

For games with a communication structure, the Myerson value is a better choice than the Shapley value as it uses the graph structure. However, it also suffers from some criticisms. For example, the fairness assumption may not be realistic. When an existing edge is broken, the value changes for players on the two edge ends can be asymmetric. Intuitively, if the edge connects a popular hub player i to a leaf player j, then the change of i can be less significant than j since j becomes isolated when (i, j) is removed. This is also the case when the game value is used for model explanation. For example in Figure 2.1 (b), when the edge ("good", "quite") is broken, the value of "quite" should change a lot. It used to contribute positively together with "good", and thus gets some payoff allocation, but it now becomes an isolated node, which is neutral by itself. On the other hand, the word "good" can still contribute positively by itself and interact with other nodes through its other edges, and thus its value shouldn't change too much. Because of such criticisms, we choose to use the HN value as our scoring function, which characterizes the value by associated consistency rather than fairness.

A.1.1.2 The C-Shapley value

The Myerson value was also mentioned in [CSW19] for the model explanation on text, where the C-Shapley value was proposed as an approximation of the Shapley value, and it was claimed to be equal to the Myerson value. We have discussed why Shapley value and Myerson are not-ideal choices for explaining graph data in Section 2.3.1 and Appendix A.1.1.1. These are partially the reason why our HN-value-based method is better than the C-Shapley value. However, the major reason why we don't do a direct comparison to the C-Shapley value as a baseline is that its formula only works for line graphs like sequence data, and not even all nodes in line graphs. In contrast, our target task is general graph prediction for graphs with possibly complicated topological structures.

We now clarify a mistake of the C-Shapley value formula and explain why it won't work for general graphs. The notations are following the [CSW19], where d is the number of players corresponding to n in our notation, and $[d]$ corresponding to N.

The formula for the C-Shapley value is given in Equation 6 in Definition 2 in [CSW19], and it is stated for "a graph G" without mentioning any assumptions of the graph. However, from the proof of this formula in Appendix B.2 of [CSW19], the line graph assumption can be seen in two places. The first place is Equation 20, where the set $\mathcal{C}$ is explicitly defined only for subsequences. The second place is Equation 22, the first line converts $\sum_{A:U_S(A)=U}$ to $\sum_{i=0}^{d-|U|-2}$, which is implicitly saying $V_S(A)$ can be picked from all d but $|U|-2$ nodes. However, this conversion is only possible when there are exactly 2 edges between U and $[d]\backslash U$, i.e. the middle part of a line graph. If there are l edges between U and $[d]\backslash U$, then the summation should go up to $d-|U|-l$. When $l=0$, i.e. U equals $[d]$ or a connected component of $[d]$, no partition is needed and the coefficient simply evaluates to 1. By correcting all these cases, the final formula for the C-Shapley value coefficients of marginal contributions thus becomes

$$\sum_{i=0}^{d-|U|-l} \frac{1}{\binom{d-1}{i+|U|-1}} \binom{d-|U|-l}{i} \tag{A.1}$$

$$= \frac{d}{(|U|+l)\binom{|U|+l-1}{|U|-1}} \tag{A.2}$$

$$= \frac{dl}{(|U|+l)(|U|+l-1)\cdots|U|} \tag{A.3}$$

for $l>0$, and 1 for $l=0$.

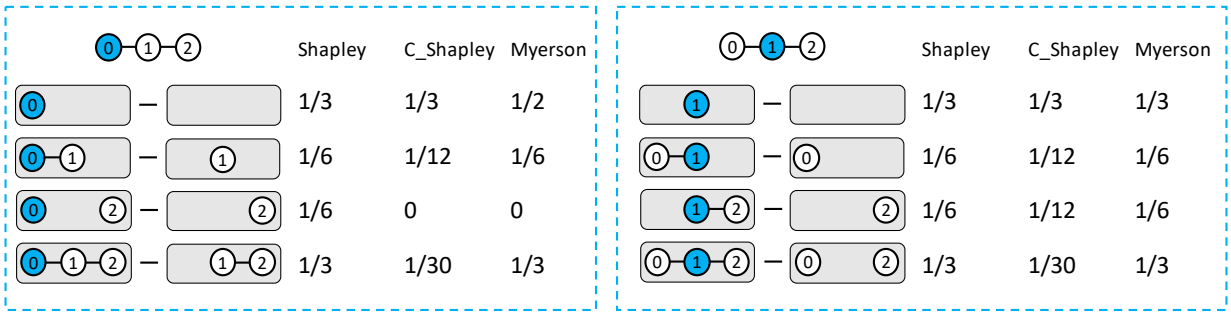


Figure A.1: **Marginal contribution coefficients comparison.** A toy 3-node graph example for comparing the marginal contribution coefficients between the Shapley, the C-Shapley, and the Myerson value. **(a)** Value computation for node 0 (left). **(b)** Value computation for node 1 (right).

The correct formula for the C-Shapley value of general graphs will be

$$\phi_X(i) = \begin{cases} \sum_{U \in \mathcal{C}} \frac{l}{(|U|+l)\cdots|U|} m_X(U,i) & \text{if } l > 0 \\ \frac{1}{d} & \text{if } l = 0 \end{cases} \tag{A.4}$$

with l represents the edges between U and $[d]\backslash U$ and $\mathcal{C}$ represents all connected subgraphs in $[d]$ containing i.

To verify this formula with the 3-node toy graph in Figure A.1. When computing the value of node 0 (left), the three connected components containing 0 are $\mathcal{C} = \{\{0\}, \{0,1\}, \{0,1,2\}\}$. Since 0 is an end node and has no leaf nodes to its left, l for these three components will be 1, 1, and 0 respectively. According to our new formula in Equation A.4, the coefficients will be $\frac{1}{2}$, $\frac{1}{6}$, and $\frac{1}{3}$ respectively, with the disconnected $\{0,2\}$ case removed. This matches the original idea of Myerson value, where the $\{0,2\} - \{2\}$ case is reduced to the $\{0\} - \emptyset$ case, which turns the Shapley coefficients from $[\frac{1}{3}, \frac{1}{6}, \frac{1}{6}, \frac{1}{3}]$ to $[\frac{1}{3} + \frac{1}{6}, \frac{1}{6}, \frac{1}{6} - \frac{1}{6}, \frac{1}{3}]$, which is $[\frac{1}{2}, \frac{1}{6}, 0, \frac{1}{3}]$. However, the original C-Shapley formula from Equation 6 in the [CSW19] evaluates to $[\frac{1}{3}, \frac{1}{12}, 0, \frac{1}{30}]$, which doesn't match the Myerson value and not even sum up to 1. Another example of

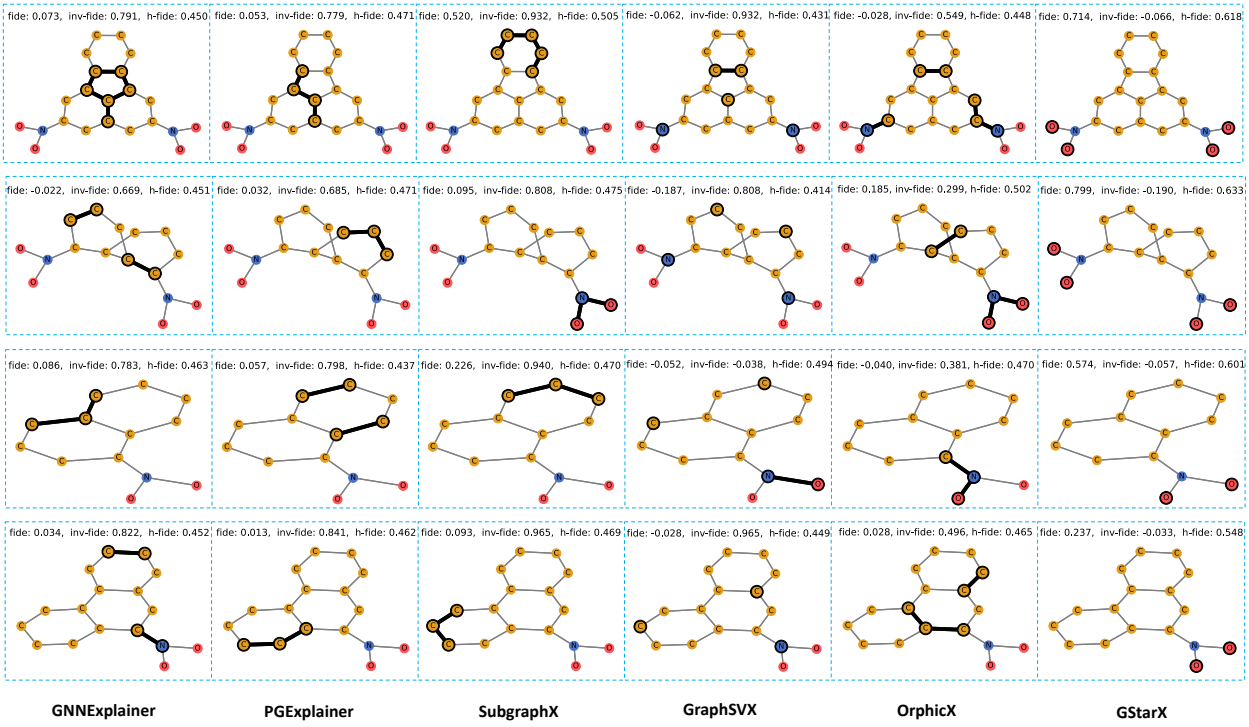


Figure A.2: **More explanations on mutagenic molecules from** MUTAG**.** Carbon atoms (C) are in yellow, nitrogen atoms (N) are in blue, and oxygen atoms (O) are in red. We use dark outlines to indicate the selected subgraph explanation and report the Fidelity (fide), Inv-Fidelity (inv-fide), and H-Fidelity (h-fide) of each explanation. GStarX gives a significant better explanation than other methods in terms of these metrics.

computing the value of node 1 is shown in Figure A.1 right.

The C-Shapley, even with the correct formula, eventually boils down to an approximation of the Shapley value or the Myerson value, which as we discussed are less ideal than the HN value. Also, the correct formula in Equation A.4 requires generating all possible subgraphs U containing the node i and specify the edges between U and $[d]\backslash U$. This makes the computation very complicated, we thus skip the comparison to the C-Shapley value.

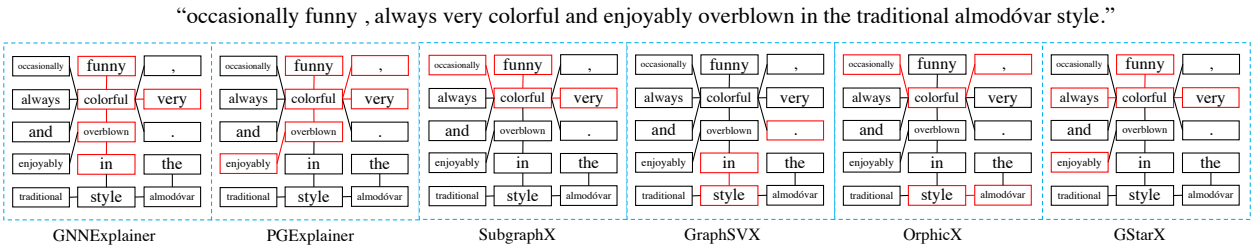


Figure A.3: **More explanations on sentences from** GraphSST2. The sentence is predicted to be positive sentiment. Red outlines indicate the selected nodes/edges as the explanation. GStarX identifies the sentiment words more accurately compared to baselines.

A.1.1.3 Use The Graph Structure via an L-hop Cutoff

Although the Shapley value itself is not structure-aware, there are existing Shapley-value-based GNN explanation methods besides the Myerson value and C-Shapley value that use an L-hop cutoff to help approximate the Shapley value [YYW21, DM21]. Technically, this operation uses the graph structure, so we can't strictly refer to these explanation methods as not structure-aware. However, we argue that the L-hop cutoff is a naive way of utilizing the graph structure. It has several concerns, and it is not the same structure-awareness as the HN value.

The L-hop cutoff approximates the Shapley value of node i by considering only the L-hop neighbors of i when explaining an L-layer GNN. The rationale of this operation is that an L-layer GNNs only propagate messages within L-hops so a node more than L-hop away from i has never passed any messages to i which means no interactions are possible. In existing Shapley-value-based GNN explanation methods, this L-hop cutoff operation was meant to reduce the exponentially growing computations of the Shapley value, and the ultimate goal is still to compute the Shapley value. The L-hop cutoff operation has several issues making it a less desirable choice. **1)** Even meant to save computation, there are still many nodes involved in the computation after applying the L-hop cutoff since the number of nodes grows exponentially as L grows. For advanced GNNs, the L can be large. When L is larger than the

diameter of the graph, which is actually the case for many recent deep GNNs, the L-hop cutoff is not effective anymore. **2)** When constructing coalitions of nodes within the local graph of L-hops, the computation still follows the Shapley value formula. This means the useful graph structure information among these nodes is forfeited which causes the structure-awareness concern of Shapley value as we discussed in Section 2.3.1,

A.1.2 More Explanation Visualizations

We visualize more explanations in Figure A.2 and Figure A.3.

A.2 Chapter 3 Appendices

A.2.1 Proof of Proposition 3.4.1

Proof. We prove $Z_{n,d} = o(S_{n,d})$ by definition, where we show $\lim_{n\to\infty} \frac{Z_{n,d}}{S_{n,d}} = 0$. As we can permute the indices of nodes in $\mathcal{G}(n,d)$, without loss of generality, we assume $Z_{n,d}$ is the expected number of paths between nodes indexed 1 and n. Our proof is mainly based on the result in [RK07], which computes the expected number of all 1-n paths, i.e., $Z_{n,d} = (n-2)!d^{n-1}e(1+o(1))$. On the other hand, the number of edge-induced subgraphs considered in [YBY19, LCX20] equals the size of the power set of all edges, i.e., $S_{n,d} = 2^{d\binom{n}{2}}$. We thus have

$$\log Z_{n,d} = \log\left[(n-2)!d^{n-1}e(1+o(1))\right] \tag{A.1}$$

$$< \log\left[\sqrt{2\pi(n-2)}(\frac{n-2}{e})^{(n-2)}e^{\frac{1}{12(n-2)}}d^{n-1}e(1+o(1))\right] \tag{A.2}$$

$$= \frac{1}{2}\log(2\pi(n-2)) + (n-2)\log(\frac{n-2}{e}) + \log\frac{1}{12(n-2)}$$

$$+ (n-1)\log d + 1 + \log(1+o(1)) \tag{A.3}$$

$$= O(\log n) + O(n\log n) + O(\log\frac{1}{n}) + O(n\log d) \tag{A.4}$$

$$+ \log(1+o(1)) \tag{A.5}$$

$$= O(n\log n) + \log(1+o(1)) \tag{A.6}$$

$$\log S_{n,d} = \log 2^{d\binom{n}{2}} = d\binom{n}{2}\log 2 = O(n^2) \tag{A.7}$$

$$\lim_{n\to\infty}\frac{Z_{n,d}}{S_{n,d}} = \lim_{n\to\infty}\exp(\log\frac{Z_{n,d}}{S_{n,d}}) \tag{A.8}$$

$$= \exp(\lim_{n\to\infty}\log\frac{Z_{n,d}}{S_{n,d}}) \tag{A.9}$$

$$= \exp(\lim_{n\to\infty}\log Z_{n,d} - \log S_{n,d}) \tag{A.10}$$

$$= \exp(\lim_{n\to\infty}O(n\log n) + \log(1+o(1)) - O(n^2)) \tag{A.11}$$

$$= 0 \tag{A.12}$$

Step (1) to (2) is Stirling's formula. Step (8) to (9) is because exp is continuous. □

A.2.2 Theorem 3.4.3: A Detailed Version

We now state a more detailed version of Theorem 3.4.3. This theorem gives the exact formula of $\delta_{\mathcal{V}}(n,d,k)$ and $\delta_{\mathrm{E}}(n,d,k)$, which are built upon a Poisson random variable. The argument is adapted from [JL08, PSW96]. Readers can refer to [JL08, PSW96] for the proof.

For $\mu > 0$, let $Po(\mu)$ denote a Poisson distribution with mean μ. Let $\psi_k(dn) = P(Po(dn) \geq$

k) be the tail probability of $Po(dn)$. Let $c_k = \inf_{\mu>0} \mu/\phi_{k-1}(\mu)$. When $dn > c_k$, the equation $\mu/\psi_{k-1}(\mu) = dn$ will have two roots for μ. Let $\mu(dn, k)$ be the larger root. Then we have the following more detailed version of Theorem 3.4.3 with $\delta_{\mathcal{V}}(n, d, k)$ and $\delta_{\mathrm{E}}(n, d, k)$ as functions of $\mu(dn, k)$.

Theorem A.2.1 (Pittel, Spencer and Wormald)**.** *Let $\mathcal{G}(n, d)$ be a random graph with m edges as in Proposition 3.4.1. Let $\mathcal{G}^k(n, d) = (\mathcal{V}^k(n, d), \mathrm{E}^k(n, d))$ be the k-core of $\mathcal{G}(n, d)$. When $dn > c_k$, $\mathcal{G}^k(n, d)$ will be nonempty with high probability (w.h.p.) for large n. Also, $\mathcal{G}^k(n, d)$ will contain $\psi_k(\mu(dn, k))n$ nodes and $[\mu(dn, k)^2/(d^2n(n-1))]m$ edges w.h.p. for large n, i.e., $|\mathcal{V}^k(n, d)|/n \xrightarrow{p} \psi_k(\mu(dn, k))$ and $|\mathrm{E}^k(n, d)|/m \xrightarrow{p} \mu(dn, k)^2/(d^2n(n-1))$ ($\xrightarrow{p}$ stands for convergence in probability).*

A.3 Chapter 4 Appendices

A.3.1 Proof of Lemma 4.4.1

In this section, we prove the non-rotation decomposition Lemma 4.4.1 stated in Section 4.4.1.

Proof. From Lemma 4.3.2, we know any non-rotation $\tilde{R} \in P \cdot \mathrm{SO}(3)$ has $\det \tilde{R} = -1$. By linearity of $\tilde{R}$ we know that

$$\langle -\tilde{R}(\boldsymbol{x}^1), -\tilde{R}(\boldsymbol{x}^2)\rangle = \langle \tilde{R}(-\boldsymbol{x}^1), \tilde{R}(-\boldsymbol{x}^2)\rangle = \langle -\boldsymbol{x}^1, -\boldsymbol{x}^2\rangle = \langle \boldsymbol{x}^1, \boldsymbol{x}^2\rangle$$

which shows hat $-\tilde{R}$ is also orthogonal according to Definition 4.3.1. As we know that $-\tilde{R}$ will have $\det -\tilde{R} = (-1)^3 \cdot -1 = 1$, by Lemma 4.3.2 again we know that $-\tilde{R}$ is a rotation. By Euler Theorem 4.3.3, we know there exists $[\alpha, \beta, \gamma] \in [0, 2\pi]^3$ such that $-\tilde{R} = O_{x_1}(\alpha)O_{x_2}(\beta)O_{x_3}(\gamma)$, which implies that $\tilde{R} = -O_{x_1}(\alpha)O_{x_2}(\beta)O_{x_3}(\gamma)$. □

A.3.2 Proof of Theorem 4.4.3

In this section, we prove the invariance in expectation Theorem 4.4.3 stated in Section 4.4.2.

Theorem A.3.1. *Assume $T_1, \ldots, T_k$ are random transformations that follow a uniform distribution over all $T \in \mathrm{O}(3)$. Then, Sym is $\mathrm{O}(3)$-invariant in expectation in the sense that $\mathbb{E}_{T_1,\ldots,T_k}[Sym_{T_1,\ldots,T_k}(G, \boldsymbol{Z}, \boldsymbol{X})] = \mathbb{E}_{T_1,\ldots,T_k}[Sym_{T_1,\ldots,T_k}(G, \boldsymbol{Z}, T_0\boldsymbol{X})]$ for any $T_0 \in \mathrm{O}(3)$.*

Proof. From Lemma 4.4.2, we know that all $T \in \mathrm{O}(3)$ have the form of $T = (-1)^{\lambda} O_{x_1}(\alpha) O_{x_2}(\beta) O_{x_3}(\gamma)$ for $\lambda \in \{0, 1\}$ and $[\alpha, \beta, \gamma] \in [-\pi, \pi]^3$. A uniform distribution over all $T \in \mathrm{O}(3)$ implies $\lambda \sim \mathrm{Bern}(0.5)$ and $[\alpha, \beta, \gamma] \sim \mathrm{Unif}([-\pi, \pi]^3)$.

Now consider a specific orthogonal transformation $T_0 = (-1)^{\lambda_0} O_{x_1}(\alpha_0) O_{x_2}(\beta_0) O_{x_3}(\gamma_0)$. Then its composition with the uniformed distributed T is

$$T \circ T_0 = (-1)^{\lambda+\lambda_0} O_{x_1}(\alpha + \alpha_0) O_{x_2}(\beta + \beta_0) O_{x_3}(\gamma + \gamma_0)$$

By the Bernoulli assumption, we get $(-1)^{\lambda} \sim (-1)^{\lambda+\lambda_0}$ as they both follow a discrete distribution on $\{-1, 1\}$ with probability 0.5 of each value. Moreover, $\alpha \sim \mathrm{Unif}[-\pi, \pi]$ implies $\alpha + \alpha_0 \sim \mathrm{Unif}[-\pi + \alpha_0, \pi + \alpha_0]$. Since this only shifts the interval that supports the uniform distribution, the joint distribution of $(\cos(\alpha), \sin(\alpha)) \sim (\cos(\alpha + \alpha_0), \sin(\alpha + \alpha_0))$ due to periodicity, which further implies $O_{x_1}(\alpha) \sim O_{x_1}(\alpha + \alpha_0)$. Similarly for $O_{x_2}(\beta)$ and $O_{x_3}(\gamma)$. Given the random variables (matrices) $(-1)^{\lambda}, O_{x_1}(\alpha), O_{x_2}(\beta)$, and $O_{x_3}(\gamma)$ are independent, we concluded that $T \sim T \circ T_0$.

Now consider $Sym_{T_1,\ldots,T_k}(G, \boldsymbol{Z}, \boldsymbol{X}) = \frac{1}{k} \sum_{i=1}^{k} Enc(G, \boldsymbol{Z}, T_i(\boldsymbol{X}))$. For each i, $T_i \sim T_i \circ T_0$ implies $T_i(\boldsymbol{X}) \sim T_i \circ T_0(\boldsymbol{X})$, and thus $Enc(G, \boldsymbol{Z}, T_i(\boldsymbol{X})) \sim Enc(G, \boldsymbol{Z}, T_i \circ T_0(\boldsymbol{X}))$ by transformation of random variables [Bil17]. Therefore, we also get $\mathbb{E}_{T_i}[Enc(G, \boldsymbol{Z}, T_i(\boldsymbol{X}))] = \mathbb{E}_{T_i}[Enc(G, \boldsymbol{Z}, T_i \circ T_0(\boldsymbol{X}))]$. Finally, by linearity of expectation, $\mathbb{E}_{T_1,\ldots,T_k}[Sym_{T_1,\ldots,T_k}(G, \boldsymbol{Z}, \boldsymbol{X})] = \mathbb{E}_{T_1,\ldots,T_k}[Sym_{T_1,\ldots,T_k}(G, \boldsymbol{Z}, T_0\boldsymbol{X})]$. □

A.3.3 More Explanation Visualizations

We provide more visualizations of the explanation generated by SymGNN. We note that none of the edges within the top 10 closest nodes are being selected as important edges by our explanation. Visualizations of more nodes are shown in Figure A.4. In the figures, the left column presents the global version plot whereas the right column presents the local version.

A.4 Chapter 5 Appendices

A.4.1 Derivations of The Log-likelihood and The Posterior Probability

We derive the log-likelihood in Equation 5.2 and and the posterior probability in Equation 5.4. Through our derivation, for simplicity, we assume $\boldsymbol{h}$ and $\boldsymbol{m}$ are both unit vectors in $\mathcal{R}^d$. In practice, $\boldsymbol{h}$ and $\boldsymbol{m}$ don't need to be equal in dimension or unit length. In this case, we apply an extra step to project them into the same space and normalize them to be unit vectors before the calculation. As this transformation only involves learnable free parameters for projection and normalization, we omit it for now.

Recall that the model assumes $P(\boldsymbol{h}_i|c_i = k) \sim \mathcal{N}(\boldsymbol{m}_k, \sigma^2\boldsymbol{I})$. Therefore,

$$
\begin{aligned}
P(\boldsymbol{h}_i|c_i = k) &= \frac{1}{\sqrt{(2\pi)^d|\sigma^2\boldsymbol{I}|}} \exp(-\frac{1}{2}(\boldsymbol{h}_i - \boldsymbol{m}_k)^T(\sigma^2\boldsymbol{I})^{-1}(\boldsymbol{h}_i - \boldsymbol{m}_k)) \\
&= \frac{1}{\sqrt{(2\pi)^d\sigma^d}} \exp(-\frac{1}{2\sigma^2}\|\boldsymbol{h}_i - \boldsymbol{m}_k\|^2) \\
&= \frac{1}{\sqrt{(2\pi)^d\sigma^d}} \exp(-\frac{1}{2\sigma^2}(2 - 2\boldsymbol{h}_i^T\boldsymbol{m}_k)) \\
&= \frac{1}{\sqrt{(2\pi)^d\sigma^d}} \exp(\frac{1}{\sigma^2}(\boldsymbol{h}_i^T\boldsymbol{m}_k - 1)) \\
&= \frac{1}{\sqrt{(2\pi)^d\sigma^d}} \exp(-\frac{1}{\sigma^2}) \exp(\frac{\boldsymbol{h}_i^T\boldsymbol{m}_k}{\sigma^2}) \\
&= g(\sigma) \exp(\frac{\boldsymbol{h}_i^T\boldsymbol{m}_k}{\sigma^2})
\end{aligned}
$$

Here we use $g(\sigma)$ to denote the part that only depends on σ. Then the joint probability is

$$P(\boldsymbol{h}_i, c_i = k) = g(\sigma) \exp(\frac{\boldsymbol{h}_i^T \boldsymbol{m}_k}{\sigma^2}) \boldsymbol{\theta}_k$$

and the posterior probability is

$$\begin{aligned}
\boldsymbol{P}_{k,i} = P(\mathrm{c}_i = k | \boldsymbol{h}_i) &= \frac{P(\boldsymbol{h}_i, c_i = k)}{\sum_{k'} P(\boldsymbol{h}_i, c_i = k')} \\
&= \frac{g(\sigma) \exp(\boldsymbol{h}_i^T \boldsymbol{m}_k / \sigma^2) \boldsymbol{\theta}_k}{\sum_{k'} g(\sigma) \exp(\boldsymbol{h}_i^T \boldsymbol{m}_{k'} / \sigma^2) \boldsymbol{\theta}_{k'}} \\
&= \frac{\exp(\boldsymbol{m}_k^T \boldsymbol{h}_i / \sigma^2) \boldsymbol{\theta}_k}{\sum_{k'} \exp(\boldsymbol{m}_{k'}^T \boldsymbol{h}_i / \sigma^2) \boldsymbol{\theta}_{k'}}
\end{aligned}$$

The log-likelihood thus becomes

$$\begin{aligned}
\ell(\boldsymbol{M}, \boldsymbol{\theta}, \phi) &= \log \prod_{i=1}^{N} (\sum_{k=1}^{K} P(\boldsymbol{h}_i, \mathrm{c}_i = k)) \\
&= \sum_{i=1}^{N} \log \sum_{k=1}^{K} g(\sigma) \exp(\frac{\boldsymbol{h}_i^T \boldsymbol{m}_k}{\sigma^2}) \boldsymbol{\theta}_k \\
&\propto \sum_{i=1}^{N} \log \sum_{k=1}^{K} \exp(\frac{\boldsymbol{h}_i^T \boldsymbol{m}_k}{\sigma^2}) \boldsymbol{\theta}_k
\end{aligned}$$

Then we derive Equation 5.9. Note that $\boldsymbol{Q}^*$ becomes a constant independent from $\boldsymbol{M}$, $\boldsymbol{\theta}$,

and ϕ after we plug it in Equation 5.8.

$$\begin{aligned}
\mathcal{L}_{OT}(\boldsymbol{M}, \boldsymbol{\theta}, \phi) &= \max_{\boldsymbol{Q} \in U(r,c)} \sum_{i=1}^{N} \sum_{k=1}^{K} \boldsymbol{Q}_{k,i} \log \frac{\boldsymbol{P}_{k,i}}{N} + \frac{1}{\lambda} H(\boldsymbol{Q}) \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} \boldsymbol{Q}^*_{k,i} \log \frac{\boldsymbol{P}_{k,i}}{N} + \frac{1}{\lambda} H(\boldsymbol{Q}^*) \\
&\propto \sum_{i=1}^{N} \sum_{k=1}^{K} \boldsymbol{Q}^*_{k,i} \log \frac{\boldsymbol{P}_{k,i}}{N} \\
&\propto \sum_{i=1}^{N} \sum_{k=1}^{K} \boldsymbol{Q}^*_{k,i} \log \boldsymbol{P}_{k,i}
\end{aligned}$$

A.4.2 Derivation of The Log-likelihood Lower Bound with Optimal Transport

The log-likelihood objective is

$$\ell(\boldsymbol{M}, \boldsymbol{\theta}, \phi) = \log \prod_{i=1}^{N} (\sum_{k=1}^{K} P(\boldsymbol{h}_i, \mathrm{c}_i = k))$$

Let $\tilde{Q}(\mathrm{c}_i)$ be a arbitrary reference distribution on c_i, i.e. $\sum_{k=1}^{K} \tilde{Q}(\mathrm{c}_i = k) = 1$. We then use the Jensen's inequality to derive the evidence lower bound, which is commonly used the objective

since the likelihood is hard to optimize directly. Consider 1/N times the log-likelihood,

$$
\begin{aligned}
&\frac{1}{N}\ell(\boldsymbol{M}, \boldsymbol{\theta}, \phi) \\
&= \frac{1}{N}\log\prod_{i=1}^{N}(\sum_{k=1}^{K} P(\boldsymbol{h}_i, \mathrm{c}_i = k)) \\
&= \frac{1}{N}\sum_{i=1}^{N}\log(\sum_{k=1}^{K} P(\boldsymbol{h}_i, \mathrm{c}_i = k)) \\
&= \frac{1}{N}\sum_{i=1}^{N}\log(\sum_{k=1}^{K} \tilde{Q}(\mathrm{c}_i = k)\frac{P(\boldsymbol{h}_i, \mathrm{c}_i = k)}{\tilde{Q}(\mathrm{c}_i = k)}) \\
&\geq \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} \tilde{Q}(\mathrm{c}_i = k)\log\frac{P(\boldsymbol{h}_i, \mathrm{c}_i = k)}{\tilde{Q}(\mathrm{c}_i = k)} \\
&= \sum_{i,k}\frac{\tilde{Q}(\mathrm{c}_i = k)}{N}\log\frac{P(\boldsymbol{h}_i, \mathrm{c}_i = k)}{\tilde{Q}(\mathrm{c}_i = k)} \\
&= \sum_{i,k}\frac{\tilde{Q}(\mathrm{c}_i = k)}{N}\log P(\boldsymbol{h}_i, \mathrm{c}_i = k) - \sum_{i,k}\frac{\tilde{Q}(\mathrm{c}_i = k)}{N}\log\tilde{Q}(\mathrm{c}_i = k) \\
&= \sum_{i,k}\frac{\tilde{Q}(\mathrm{c}_i = k)}{N}\log P(\boldsymbol{h}_i, \mathrm{c}_i = k) - \sum_{i,k}\frac{\tilde{Q}(\mathrm{c}_i = k)}{N}\log\frac{\tilde{Q}(\mathrm{c}_i = k)}{N} \\
&\quad - \log N
\end{aligned}
$$

To proceed, we ignore the constant $\log N$, and let $\boldsymbol{Q}_{k,i} = \frac{\tilde{Q}(\mathrm{c}_i = k)}{N}$. Observe that $\sum_{i=1}^{N}\boldsymbol{Q}_{k,i} = \tilde{Q}(\mathrm{c}_i = k)$ and $\sum_{k=1}^{K}\boldsymbol{Q}_{k,i} = \frac{1}{N}$, which gives $\boldsymbol{Q}_{k,i}$ a probabilistic interpretation as a joint distribution on c_i and $\boldsymbol{h}_i$, where the marginal distribution equals $\tilde{Q}(\mathrm{c}_i)$ and the empirical distribution on $\{\boldsymbol{h}_1, ..., \boldsymbol{h}_N\}$ respectively. We thus get

$$
\begin{aligned}
\frac{1}{N}\ell(\boldsymbol{M}, \boldsymbol{\theta}, \phi) &\propto \sum_{i,k}\boldsymbol{Q}_{k,i}\log P(\boldsymbol{h}_i, \mathrm{c}_i = k) - \sum_{i,k}\boldsymbol{Q}_{k,i}\log\boldsymbol{Q}_{k,i} \\
&= \sum_{i,k}\boldsymbol{Q}_{k,i}\log P(\boldsymbol{h}_i, \mathrm{c}_i = k) + H(\boldsymbol{Q})
\end{aligned}
$$

Equation 5.8 is the same objective with a restricted search space $\boldsymbol{U}(\boldsymbol{r}, \boldsymbol{c})$ and an extra parameter λ controlling the entropy. Both terms work together to prevent cluster collapse [GDV19, YCA20].

A.5 Chapter 6 Appendices

A.5.1 Decision Tree Visualizations

We visualize the decision tree which makes the prediction process explainable. Figure A.5 shows the impurity details of the decision tree shown in Figure 6.9 (b) and Figure A.6 shows a sample decision tree for the BACE dataset.

Figure A.4: Local and global explanation visualizations for SymGNN on more MG nodes.

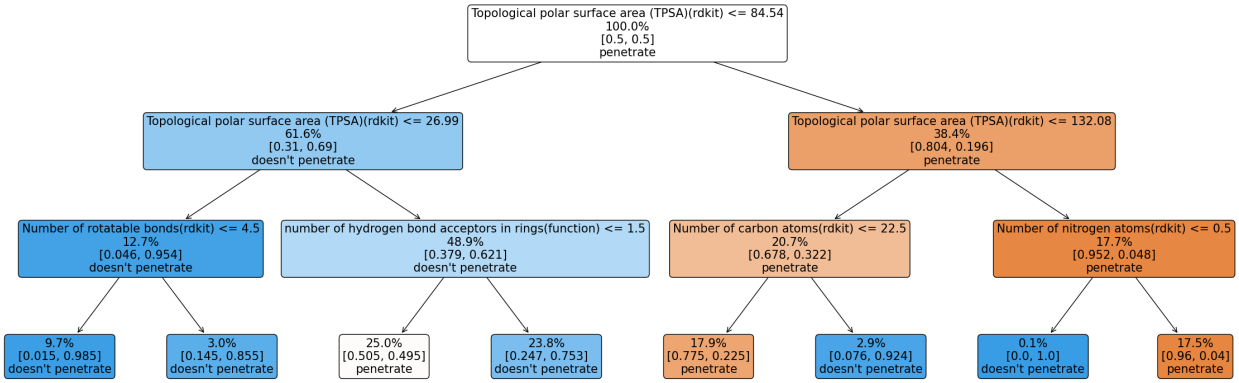


Figure A.5: The decision tree for AutoMolCo-induced CM classification on BBBP.

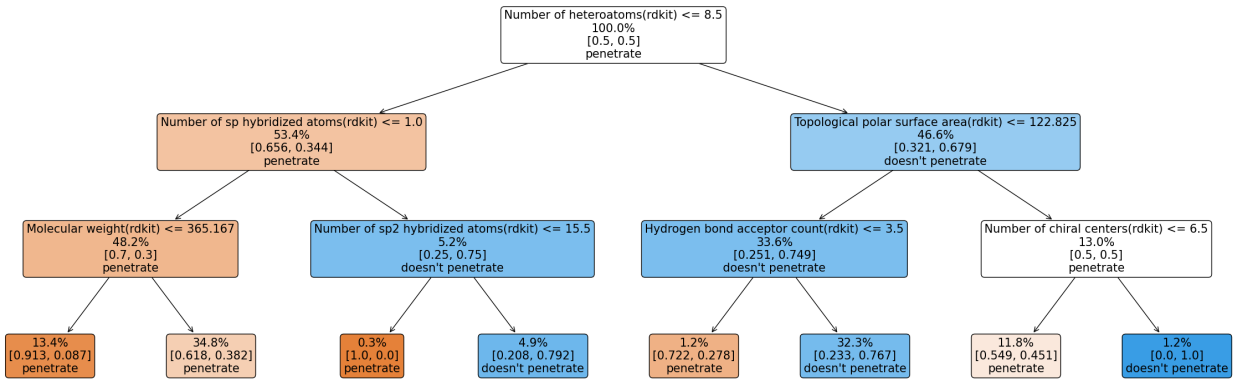


Figure A.6: The decision tree for AutoMolCo-induced CM classification on BACE.

REFERENCES

[AEL18] DT Ahneman, JG Estrada, S Lin, SD Dreher, and AG Doyle. "Predicting reaction performance in c–n cross-coupling using machine learning." Science, 2018.

[Aka73] Hirotugu Akaike. "Information theory and an extension of the maximum likelihood principle." In 2nd International Symposium on Information Theory, pp. 267–281. Akademiai Kiado, 1973.

[Aka74] Hirotugu Akaike. "A new look at the statistical model identification." IEEE Transactions on Automatic Control, **19**(6):716–723, 1974.

[ALC22] Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. "A review on language models as knowledge bases." arXiv preprint arXiv:2204.06031, 2022.

[Avd12] Alex Avdeef. Absorption and drug development: solubility, permeability, and charge state. John Wiley & Sons, 2012.

[BA19] Federico Baldassarre and Hossein Azizpour. "Explainability Techniques for Graph Convolutional Networks.", 2019.

[Bar94] Serguei Barannikov. "The framed Morse complex and its invariants." Advances in Soviet Mathematics, **21**:93–116, 1994.

[BB11] Ludovic Berthier and Giulio Biroli. "Theoretical perspective on the glass transition and amorphous materials." Reviews of modern physics, **83**(2):587, 2011.

[BD13] Narottam P Bansal and Robert H Doremus. Handbook of glass properties. Elsevier, 2013.

[BEL13] Ulrik Brandes, Markus Eiglsperger, Jürgen Lerner, and Christian Pich. "Graph markup language (GraphML).", 2013.

[BFZ21] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. "Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting.", 2021.

[BGA20] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. "Spectral clustering with graph neural networks for graph pooling." In International Conference on Machine Learning, pp. 874–883. PMLR, 2020.

[BHJ19] Maxime Bucher, Stéphane Herbin, and Frédéric Jurie. "Semantic bottleneck for computer vision tasks." In Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part II 14, pp. 695–712. Springer, 2019.

[Bil17] Patrick Billingsley. Probability and measure. John Wiley & Sons, 2017.

[BKG20] V. Bapst, T. Keck, A. Grabska-Barwińska, C. Donner, E. D. Cubuk, S. S. Schoenholz, A. Obika, A. W. R. Nelson, T. Back, D. Hassabis, and P. Kohli. "Unveiling the predictive power of static structure in glassy systems." Nature Physics, **16**(4):448–454, April 2020.

[BKS22] Ilyes Batatia, David P Kovacs, Gregor Simm, Christoph Ortner, and Gábor Csányi. "MACE: Higher order equivariant message passing neural networks for fast and accurate force fields." Advances in Neural Information Processing Systems, **35**:11423–11436, 2022.

[BM96] GT Barkema and Normand Mousseau. "Event-based relaxation of continuous disordered systems." Physical review letters, **77**(21):4358, 1996.

[BM05] Mustafa Bilgic and Raymond J Mooney. "Explaining recommendations: Satisfaction vs. promotion." In Beyond personalization workshop, IUI, volume 5, p. 153, 2005.

[BMP24] Vaibhav Bihani, Sajid Mannan, Utkarsh Pratiush, Tao Du, Zhimin Chen, Santiago Miret, Matthieu Micoulaut, Morten M Smedskjaer, Sayan Ranu, and NM Anoop Krishnan. "EGraFFBench: evaluation of equivariant graph neural network force fields for atomistic simulations." Digital Discovery, **3**(4):759–768, 2024.

[BMS22] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E Smidt, and Boris Kozinsky. "E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials." Nature communications, **13**(1):2453, 2022.

[Bol84] Béla Bollobás. "The evolution of sparse graphs, Graph theory and combinatorics (Cambridge, 1983).", 1984.

[BOT92] Peter Borm, Guillerom Owen, and Stif Tijs. "On the position value for communication situations." SIAM Journal on Discrete Mathematics, **5**(3):305–320, 1992.

[BZS13] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs." arXiv preprint arXiv:1312.6203, 2013.

[CBJ18] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. "Deep Clustering for Unsupervised Learning of Visual Features.", 2018.

[CBR20] Zhi Chen, Yijie Bei, and Cynthia Rudin. "Concept whitening for interpretable image recognition." Nature Machine Intelligence, **2**(12):772–782, 2020.

[CCV20] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. "Can graph neural networks count substructures?" arXiv preprint arXiv:2002.04025, 2020.

[CGR20] Kamal Choudhary, Kevin F Garrity, Andrew CE Reid, Brian DeCost, Adam J Biacchi, Angela R Hight Walker, Zachary Trautt, Jason Hattrick-Simpers, A Gilad Kusne, Andrea Centrone, et al. "The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design." npj computational materials, **6**(1):173, 2020.

[CHL06] Jin Chen, Wynne Hsu, Mong Li Lee, and See-Kiong Ng. "Nemofinder: Dissecting genome-wide protein-protein interactions with meso-scale network motifs." In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 106–115, 2006.

[CIS17] Ekin Dogus Cubuk, RJS Ivancic, Samuel S Schoenholz, DJ Strickland, Anindita Basu, ZS Davidson, Julien Fontaine, Jyo Lyn Hor, Y-R Huang, Y Jiang, et al. "Structure-property relationships from universal signatures of plasticity in disordered solids." Science, **358**(6366):1033–1037, 2017.

[CLK18] Yixin Cao, Jindong Li, Binquan Kou, Chengjie Xia, Zhifeng Li, Rongchang Chen, Honglan Xie, Tiqiao Xiao, Walter Kob, Liang Hong, et al. "Structural and topological nature of plasticity in sheared granular materials." Nature communications, **9**(1):2911, 2018.

[CLM09] E Cances, Frédéric Legoll, M-C Marinica, K Minoukadeh, and F Willaime. "Some improvements of the activation-relaxation technique method for finding transition pathways on potential energy surfaces." The Journal of chemical physics, **130**(11), 2009.

[CM11] YQ Cheng and E Ma. "Atomic-level structure and structure–property relationship in metallic glasses." Progress in materials science, **56**(4):379–473, 2011.

[CMM20] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments." In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pp. 9912–9924. Curran Associates, Inc., 2020.

[CP15] Kee-Choo Chung and Hwangseo Park. "Accuracy enhancement in the estimation of molecular hydration free energies by implementing the intramolecular hydrogen bond effects." Journal of Cheminformatics, **7**:1–12, 2015.

[CSW19] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. "L-Shapley and C-Shapley: Efficient Model Interpretation for Structured Data." In International Conference on Learning Representations, 2019.

[Cut13] Marco Cuturi. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport." In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.

[CYZ21] Kewei Cheng, Ziqing Yang, Ming Zhang, and Yizhou Sun. "UniKER: A Unified Framework for Combining Embedding and Definite Horn Rule Reasoning for Knowledge Graph Inference." In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 9753–9771, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[DBY19] Nima Dehmamy, Albert-László Barabási, and Rose Yu. "Understanding the representation power of graph neural networks in learning graph topology." arXiv preprint arXiv:1907.05008, 2019.

[DCD91] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity." Journal of medicinal chemistry, **34**(2):786–797, 1991.

[DG17] Piotr Dabkowski and Yarin Gal. "Real time image saliency for black box classifiers." arXiv preprint arXiv:1705.07857, 2017.

[Dij59] Edsger W Dijkstra. "A note on two problems in connexion with graphs." Numerische mathematik, **1**(1):269–271, 1959.

[DKW05] Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis. "Frequent substructure-based approaches for classifying chemical compounds." IEEE Transactions on Knowledge and Data Engineering, **17**(8):1036–1050, 2005.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the Royal Statistical Society: Series B (Methodological), **39**(1):1–22, 1977.

[DLR18] Jeffrey De Fauw, Joseph R Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan O'Donoghue, Daniel Visentin, et al. "Clinically applicable deep learning for diagnosis and referral in retinal disease." Nature medicine, **24**(9):1342–1350, 2018.

[DM65] Morton Davis and Michael Maschler. "The kernel of a cooperative game." Naval Research Logistics Quarterly, **12**(3):223–259, 1965.

[DM21] Alexandre Duval and Fragkiskos D Malliaros. "GraphSVX: Shapley Value Explanations for Graph Neural Networks." arXiv preprint arXiv:2104.10482, 2021.

[DMJ23] Alexandre Duval, Simon V Mathis, Chaitanya K Joshi, Victor Schmidt, Santiago Miret, Fragkiskos D Malliaros, Taco Cohen, Pietro Liò, Yoshua Bengio, and Michael Bronstein. "A Hitchhiker's Guide to Geometric GNNs for 3D Atomic Systems." arXiv preprint arXiv:2312.07511, 2023.

[DPF14] Jun Ding, Sylvain Patinet, Michael L Falk, Yongqiang Cheng, and Evan Ma. "Soft spots and their structural signature in a metallic glass." Proceedings of the National Academy of Sciences, **111**(39):14052–14056, 2014.

[DS01] Pablo G Debenedetti and Frank H Stillinger. "Supercooled liquids and the glass transition." Nature, **410**(6825):259–267, 2001.

[DSH23] Alexandre Agm Duval, Victor Schmidt, Alex Hernández-Garcıa, Santiago Miret, Fragkiskos D Malliaros, Yoshua Bengio, and David Rolnick. "Faenet: Frame averaging equivariant gnn for materials modeling." In International Conference on Machine Learning, pp. 9013–9033. PMLR, 2023.

[EDR23] Takeshi Egami, Wojciech Dmowski, and Chae Woo Ryu. "Medium-range order resists deformation in metallic liquids and glasses." Metals, **13**(3):442, 2023.

[EH08] Herbert Edelsbrunner, John Harer, et al. "Persistent homology-a survey." Contemporary mathematics, **453**(26):257–282, 2008.

[FIE14] Yue Fan, Takuya Iwashita, and Takeshi Egami. "How thermally activated deformation starts in metallic glass." Nature communications, **5**(1):5083, 2014.

[FIE17] Yue Fan, Takuya Iwashita, and Takeshi Egami. "Energy landscape-driven non-equilibrium evolution of inherent structure in disordered material." Nature communications, **8**(1):15417, 2017.

[fil18] Yuval Filmus (https://cs.stackexchange.com/users/683/yuval filmus). "number of connected subgraphs of G with at most k $>$ 0 vertices." stackexchange, 2018.

[FKA21] Thorben Funke, Megha Khosla, and Avishek Anand. "Zorro: Valid, sparse, and stable explanations in graph neural networks." arXiv preprint arXiv:2105.08621, 2021.

[FL11] Michael L Falk and James S Langer. "Deformation and failure of amorphous, solidlike materials." Annu. Rev. Condens. Matter Phys., **2**(1):353–373, 2011.

[FL19] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric." In ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

[FMT04] Christos Faloutsos, Kevin S McCurley, and Andrew Tomkins. "Fast discovery of connection subgraphs." In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 118–127, 2004.

[GBS20] Azin Ghazimatin, Oana Balalau, Rishiraj Saha Roy, and Gerhard Weikum. "PRINCE: Provider-side interpretability with counterfactual explanations in recommender systems." In Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 196–204, 2020.

[GC08] Cristiano RW Guimarães and Mario Cardozo. "MM-GB/SA rescoring of docking poses in structure-based lead optimization." Journal of chemical information and modeling, **48**(5):958–970, 2008.

[GDL23] Jiayan Guo, Lun Du, and Hengyu Liu. "GPT4Graph: Can Large Language Models Understand Graph Structured Data? An Empirical Evaluation and Benchmarking." arXiv preprint arXiv:2305.15066, 2023.

[GDV19] Aude Genevay, Gabriel Dulac-Arnold, and Jean-Philippe Vert. "Differentiable deep clustering with cluster size constraints." arXiv preprint arXiv:1910.09036, 2019.

[GGG20] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. "Directional message passing for molecular graphs." arXiv preprint arXiv:2003.03123, 2020.

[GGM20] Johannes Gasteiger, Shankari Giri, Johannes T Margraf, and Stephan Günnemann. "Fast and uncertainty-aware directional message passing for non-equilibrium molecules." arXiv preprint arXiv:2011.14115, 2020.

[GGN18] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. "Allennlp: A deep semantic natural language processing platform." arXiv preprint arXiv:1803.07640, 2018.

[GGN23] Taicheng Guo, Kehan Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. "What can Large Language Models do in chemistry? A comprehensive benchmark on eight tasks." arXiv preprint arXiv:2305.18365, 2023.

[GJJ20] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. "Generalization and representational limits of graph neural networks." In International Conference on Machine Learning, pp. 3419–3430. PMLR, 2020.

[GSZ22] Zhichun Guo, William Shiao, Shichang Zhang, Yozen Liu, Nitesh Chawla, Neil Shah, and Tong Zhao. "Linkless Link Prediction via Relational Distillation." arXiv preprint arXiv:2210.05801, 2022.

[GWB02] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. "Gene selection for cancer classification using support vector machines." Machine learning, **46**:389–422, 2002.

[Ham99] Gérard Hamiache. "A value with incomplete communication." Games and Economic Behavior, **26**(1):59–78, 1999.

[Ham01] Gérard Hamiache. "Associated consistency and Shapley value." International Journal of Game Theory, **30**(2):279–289, 2001.

[HCS21] Yu Hao, Xin Cao, Yufan Sheng, Yixiang Fang, and Wei Wang. "Ks-gnn: Keywords search over incomplete graphs via graphs neural network." Advances in Neural Information Processing Systems, **34**:1700–1712, 2021.

[HDW20] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. "GPT-GNN: Generative Pre-Training of Graph Neural Networks." In Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2020.

[HFZ20] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. "Open Graph Benchmark: Datasets for Machine Learning on Graphs.", 2020.

[HH13] Brian C Hall and Brian C Hall. Lie groups, Lie algebras, and representations. Springer, 2013.

[Him97] Michael Himsolt. "GML: Graph modelling language." University of Passau, 1997.

[HK20] Kaveh Hassani and Amir Hosein Khasahmadi. "Contrastive multi-view representation learning on graphs." In International Conference on Machine Learning, pp. 4116–4126. PMLR, 2020.

[HKR00] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. "Explaining collaborative filtering recommendations." In Proceedings of the 2000 ACM conference on Computer supported cooperative work, pp. 241–250, 2000.

[HLG20] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. "Strategies for Pre-training Graph Neural Networks." In International Conference on Learning Representations, 2020.

[HM89] S Hart and A Mas-Colell. "Potential, value, and consistency." Econometrica, **57**(3):589–614, 1989.

[HN20] Gérard Hamiache and Florian Navarro. "Associated consistency, value and graphs." International Journal of Game Theory, **49**(1):227–249, 2020.

[HNH16] Yasuaki Hiraoka, Takenobu Nakamura, Akihiko Hirata, Emerson G Escolar, Kaname Matsue, and Yasumasa Nishiura. "Hierarchical structures of amorphous solids characterized by persistent homology." Proceedings of the National Academy of Sciences, **113**(26):7035–7040, 2016.

[Hoo85] William G Hoover. "Canonical dynamics: Equilibrium phase-space distributions." Physical review A, **31**(3):1695, 1985.

[HYL17] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." In Advances in neural information processing systems, pp. 1024–1034, 2017.

[HYT20] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. "GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks.", 2020.

[IZK20] Vassilis N Ioannidis, Da Zheng, and George Karypis. "Few-shot link prediction via graph neural networks for covid-19 drug-repurposing." arXiv preprint arXiv:2007.10261, 2020.

[JGS22] Antoine Jay, Miha Gunde, Nicolas Salles, Matic Poberžnik, Layla Martin-Samos, Nicolas Richard, Stefano de Gironcoli, Normand Mousseau, and Anne Hémeryck. "Activation–Relaxation Technique: An efficient way to find minima and saddle points of potential energy surfaces." Computational Materials Science, **209**:111363, 2022.

[JL08] Svante Janson and Malwina J Luczak. "Asymptotic normality of the k-core in random graphs." The annals of applied probability, **18**(3):1085–1137, 2008.

[JLH23] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. "Large Language Models on Graphs: A Comprehensive Survey." arXiv preprint arXiv:2312.02783, 2023.

[JW02] Glen Jeh and Jennifer Widom. "Simrank: a measure of structural-context similarity." In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 538–543, 2002.

[Kat53] Leo Katz. "A new status index derived from sociometric analysis." Psychometrika, **18**(1):39–43, 1953.

[KB17] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization.", 2017.

[KCZ22] Katelyn A Kirchner, Daniel R Cassar, Edgar D Zanotto, Madoka Ono, Seong H Kim, Karan Doss, Mikkel L Bødker, Morten M Smedskjaer, Shinji Kohara, Longwen Tang, et al. "Beyond the average: spatial and temporal fluctuations in oxide glass-forming systems." Chemical reviews, **123**(4):1774–1840, 2022.

[KIM04] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. "Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs." Bioinformatics, **20**(11):1746–1758, 2004.

[KKU06] Atsushi Kajii, Hiroyuki Kojima, and Takashi Ui. "A refinement of the Myerson value." IMS Preprint Series, **25**, 2006.

[KLJ21] Kamil Kamiński, Jan Ludwiczak, Maciej Jasiński, Adriana Bukala, Rafal Madaj, Krzysztof Szczepaniak, and Stanisław Dunin-Horkawicz. "Rossmann-toolbox: a deep learning-based protocol for the prediction and design of cofactor specificity in Rossmann fold proteins." Briefings in Bioinformatics, **23**(1):bbab371, 09 2021.

[KNT20] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. "Concept bottleneck models." In International conference on machine learning, pp. 5338–5348. PMLR, 2020.

[KW16] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907, 2016.

[KW17] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks.", 2017.

[KWY17] NM Anoop Krishnan, Bu Wang, Yingtian Yu, Yann Le Pape, Gaurav Sant, and Mathieu Bauchy. "Enthalpy landscape dictates the irradiation-induced disordering of quartz." Physical Review X, **7**(3):031019, 2017.

[Lan10] Greg Landrum. "RDKit: Open-source cheminformatics." `https://www.rdkit.org`, 2010. Accessed: Nov 22, 2023.

[LC01] Stan Lipovetsky and Michael Conklin. "Analysis of regression in game theory approach." Applied Stochastic Models in Business and Industry, **17**(4):319–330, 2001.

[LCX20] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. "Parameterized Explainer for Graph Neural Network." In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pp. 19620–19631. Curran Associates, Inc., 2020.

[LFS19] Max Losch, Mario Fritz, and Bernt Schiele. "Interpretability beyond classification output: Semantic bottleneck networks." arXiv preprint arXiv:1907.10882, 2019.

[LGN23] Kin Long Kelvin Lee, Carmelo Gonzales, Marcel Nassar, Matthew Spellings, Mikhail Galkin, and Santiago Miret. "Matsciml: A broad, multi-task benchmark for solid-state materials modeling." arXiv preprint arXiv:2309.05934, 2023.

[LK07] David Liben-Nowell and Jon Kleinberg. "The link-prediction problem for social networks." Journal of the American society for information science and technology, **58**(7):1019–1031, 2007.

[LL17] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions." In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pp. 4765–4774. Curran Associates, Inc., 2017.

[LLD97] Christopher A Lipinski, Franco Lombardo, Beryl W Dominy, and Paul J Feeney. "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings." Advanced drug delivery reviews, **23**(1-3):3–25, 1997.

[LLL12] Zengqian Liu, Ran Li, Gang Liu, Wenhuang Su, Hui Wang, Yan Li, Minjie Shi, Xuekun Luo, Guojuan Wu, and Tao Zhang. "Microstructural tailoring and improvement of mechanical properties in CuZr-based bulk metallic glass composites." Acta Materialia, **60**(6-7):3128–3139, 2012.

[LLL21] Wanyu Lin, Hao Lan, and Baochun Li. "Generative causal explanations for graph neural networks." In International Conference on Machine Learning, pp. 6666–6679. PMLR, 2021.

[LLW19] Chengqiang Lu, Qi Liu, Chao Wang, Zhenya Huang, Peize Lin, and Lixin He. "Molecular property prediction: A multilevel quantum interactions modeling perspective." In Proceedings of the AAAI conference on artificial intelligence, volume 33, pp. 1052–1060, 2019.

[LLW21] Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Haiyang Yu, Zhao Xu, Jingtun Zhang, Yi Liu, Keqiang Yan, Haoran Liu, Cong Fu, Bora M Oztekin, Xuan Zhang, and Shuiwang Ji. "DIG: A Turnkey Library for Diving into Graph Deep Learning Research." Journal of Machine Learning Research, **22**(240):1–9, 2021.

[LLW22] Wanyu Lin, Hao Lan, Hao Wang, and Baochun Li. "OrphicX: A Causality-Inspired Latent Variable Model for Interpreting Graph Neural Networks." arXiv preprint arXiv:2203.15209, 2022.

[LNH09] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. "Learning to detect unseen object classes by between-class attribute transfer." In 2009 IEEE conference on computer vision and pattern recognition, pp. 951–958. IEEE, 2009.

[LOL18] Bruno Lepri, Nuria Oliver, Emmanuel Letouzé, Alex Pentland, and Patrick Vinck. "Fair, transparent, and accountable algorithmic decision-making processes." Philosophy & Technology, **31**(4):611–627, 2018.

[LOS21] Markus Langer, Daniel Oster, Timo Speith, Holger Hermanns, Lena Kästner, Eva Schmidt, Andreas Sesing, and Kevin Baum. "What do we want from Explainable Artificial Intelligence (XAI)?–A stakeholder perspective on XAI and a conceptual

model guiding interdisciplinary XAI research." Artificial Intelligence, **296**:103473, 2021.

[LPL21] Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. "Adsgnn: Behavior-graph augmented relevance modeling in sponsored search." In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 223–232, 2021.

[LS22] Yi-Lun Liao and Tess Smidt. "Equiformer: Equivariant graph attention transformer for 3d atomistic graphs." arXiv preprint arXiv:2206.11990, 2022.

[LTT22] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. "Cf-gnnexplainer: Counterfactual explanations for graph neural networks." In International Conference on Artificial Intelligence and Statistics, pp. 4499–4511. PMLR, 2022.

[LZX20] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven C.H. Hoi. "Prototypical Contrastive Learning of Unsupervised Representations." arXiv preprint arXiv:2005.04966, 2020.

[MBB12] Normand Mousseau, Laurent Karim Béland, Peter Brommer, Jean-François Joly, Fedwa El-Mellouhi, Eduardo Machado-Charry, Mihai-Cosmin Marinica, and Pascal Pochet. "The activation-relaxation technique: Art nouveau and kinetic art." Journal of Atomic and Molecular Physics, **2012**, 2012.

[MBK22] Lucie Charlotte Magister, Pietro Barbiero, Dmitry Kazhdan, Federico Siciliano, Gabriele Ciravegna, Fabrizio Silvestri, Mateja Jamnik, and Pietro Lio. "Encoding concepts in graph neural networks." arXiv preprint arXiv:2207.13586, 2022.

[MKB20] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. "TUDataset: A collection of benchmark datasets for learning with graphs.", 2020.

[MKO09] MI Mendelev, MJ Kramer, RT Ott, DJ Sordelet, D Yagodin, and PJPM Popel. "Development of suitable interatomic potentials for simulation of liquid and amorphous Cu–Zr alloys." Philosophical Magazine, **89**(11):967–987, 2009.

[MKS21] Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Liò. "Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks." arXiv preprint arXiv:2107.11889, 2021.

[MNV20] Gonzalo Mena, Amin Nejatbakhsh, Erdem Varol, and Jonathan Niles-Weed. "Sinkhorn EM: an expectation-maximization algorithm based on entropic optimal transport." arXiv preprint arXiv:2006.16548, 2020.

[MSI02] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. "Network motifs: simple building blocks of complex networks." Science, **298**(5594):824–827, 2002.

[MSW09] Dong Ma, Alexandru Dan Stoica, and X-L Wang. "Power-law scaling and fractal nature of medium-range order in metallic glasses." Nature materials, **8**(1):30–34, 2009.

[Mye77] Roger B Myerson. "Graphs and cooperation in games." Mathematics of operations research, **2**(3):225–229, 1977.

[MZ75] KV Mardia and PJ Zemroch. "Algorithm AS 86: The von Mises distribution function." Journal of the Royal Statistical Society. Series C (Applied Statistics), **24**(2):268–272, 1975.

[MZX21] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. "UltraGCN: ultra simplification of graph convolutional networks for recommendation." In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 1253–1262, 2021.

[NCG21] Keita Nomoto, Anna V Ceguerra, Christoph Gammer, Bosong Li, Huma Bilal, Anton Hohenwarter, Bernd Gludovatz, Jürgen Eckert, Simon P Ringer, and Jamie J Kruzic. "Medium-range order dictates local hardness in bulk metallic glasses

." Materials today, **44**:48–57, 2021.

[NFM18] Alexandre Nicolas, Ezequiel E Ferrero, Kirsten Martens, and Jean-Louis Barrat. "Deformation and flow of amorphous solids: Insights from elastoplastic models." Reviews of Modern Physics, **90**(4):045006, 2018.

[NMT15] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. "A review of relational machine learning for knowledge graphs." Proceedings of the IEEE, **104**(1):11–33, 2015.

[Nos84] Shuichi Nosé. "A unified formulation of the constant temperature molecular dynamics methods." The Journal of chemical physics, **81**(1):511–519, 1984.

[Oba18] Ippei Obayashi. "Volume-optimal cycle: Tightest representative cycle of a generator in persistent homology." SIAM Journal on Applied Algebra and Geometry, **2**(4):508–534, 2018.

[ODN23] Tuomas Oikarinen, Subhro Das, Lam Nguyen, and Lily Weng. "Label-free Concept Bottleneck Models." In International Conference on Learning Representations, 2023.

[Pel86] Bezalel Peleg. "On the reduced game property and its converse." International Journal of Game Theory, **15**(3):187–200, 1986.

[PGW10] S Pauly, S Gorantla, G Wang, U Kühn, and J Eckert. "Transformation-mediated ductility in CuZr-based bulk metallic glasses." Nature materials, **9**(6):473–477, 2010.

[PKF20] Namyong Park, Andrey Kan, Christos Faloutsos, and Xin Luna Dong. "J-Recs: Principled and Scalable Recommendation Justification." In 2020 IEEE International Conference on Data Mining (ICDM), pp. 1208–1213. IEEE, 2020.

[PKR19] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. "Explainability methods for graph convolutional neural networks." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10772–10781, 2019.

[PL05] H. Pajouhesh and G.R. Lenz. "Medicinal Chemical Properties of Successful Central Nervous System Drugs." NeuroRx, **2**(4):541–553, 2005.

[Pli95] Steve Plimpton. "Fast parallel algorithms for short-range molecular dynamics." Journal of computational physics, **117**(1):1–19, 1995.

[PPG01] Carl O Pabo, Ezra Peisach, and Robert A Grant. "Design and selection of novel Cys2His2 zinc finger proteins." Annual review of biochemistry, **70**(1):313–340, 2001.

[PRL19] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. "Language models as knowledge bases?" arXiv preprint arXiv:1909.01066, 2019.

[PSW96] Boris Pittel, Joel Spencer, and Nicholas Wormald. "Sudden emergence of a giantk-core in a random graph." Journal of Combinatorial Theory, Series B, **67**(1):111–151, 1996.

[PVF16] Sylvain Patinet, Damien Vandembroucq, and Michael L Falk. "Connecting local yield stresses with plastic activity in amorphous solids." Physical review letters, **117**(4):045501, 2016.

[PVG11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, **12**:2825–2830, 2011.

[RBX20] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying WEI, Wenbing Huang, and Junzhou Huang. "Self-Supervised Graph Transformer on Large-Scale Molecular Data." In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors,

Advances in Neural Information Processing Systems, volume 33, pp. 12559–12571. Curran Associates, Inc., 2020.

[REW19] Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. Deep Learning for the Life Sciences. O'Reilly Media, 2019. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837.

[RK07] Ben Roberts and Dirk P Kroese. "Estimating the Number of st Paths in a Graph." J. Graph Algorithms Appl., **11**(1):195–214, 2007.

[RKK19] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. "On the convergence of adam and beyond." arXiv preprint arXiv:1904.09237, 2019.

[RNE22] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, et al. "Graph neural networks for materials science and chemistry." Communications Materials, **3**(1):93, 2022.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "" Why should i trust you?" Explaining the predictions of any classifier." In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1135–1144, 2016.

[RWB16] Brandon J. Reizman, Yi-Ming Wang, Stephen L. Buchwald, and Klavs F. Jensen. "Suzuki–Miyaura cross-coupling optimization enabled by automated feedback." React. Chem. Eng., **1**:658–666, 2016.

[SBB20] Søren S Sørensen, Christophe AN Biscio, Mathieu Bauchy, Lisbeth Fajstrup, and Morten M Smedskjaer. "Revealing hidden medium-range order in amorphous materials using topological data analysis." Science Advances, **6**(37):eabc2320, 2020.

[SCD17] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. "Grad-cam: Visual explanations from deep networks via gradient-based localization." In Proceedings of the IEEE international conference on computer vision, pp. 618–626, 2017.

[SDS98] Srikanth Sastry, Pablo G Debenedetti, and Frank H Stillinger. "Signatures of distinct dynamical regimes in the energy landscape of a glass-forming liquid." Nature, **393**(6685):554–557, 1998.

[Sha53] Lloyd Shapley. "A value fo n-person Games." Ann. Math. Study28, Contributions to the Theory of Games, ed. by HW Kuhn, and AW Tucker, pp. 307–317, 1953.

[SHK11] Jan Schroers, Thomas M. Hodges, Golden Kumar, Hari Raman, Anthony J. Barnes, Quoc Pham, and Theodore A. Waniuk. "Thermoplastic blow molding of metals." Materials Today, **14**(1):14–19, 2011.

[SHV20] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. "InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization." In International Conference on Learning Representations, 2020.

[SHW22] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. "E(n) Equivariant Graph Neural Networks.", 2022.

[SHY11] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks." Proceedings of the VLDB Endowment, **4**(11):992–1003, 2011.

[SK14] Erik Štrumbelj and Igor Kononenko. "Explaining prediction models and individual predictions with feature contributions." Knowledge and information systems, **41**(3):647–665, 2014.

[SKB18] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. "Modeling relational data with graph convolutional networks." In European semantic web conference, pp. 593–607. Springer, 2018.

[SKE19] Murat Cihan Sorkun, Abhishek Khetan, and Süleyman Er. "AqSolDB, a curated reference set of aqueous solubility and 2D descriptors for a diverse set of compounds." Scientific data, **6**(1):143, 2019.

[SKS17] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions." Advances in neural information processing systems, **30**, 2017.

[Sla99] Gregory G Slabaugh. "Computing Euler angles from a rotation matrix." Retrieved on August, **6**(2000):39–63, 1999.

[SLA06] HW Sheng, WK Luo, FM Alamgir, JM Bai, and E Ma. "Atomic packing and short-to-medium-range order in metallic glasses." Nature, **439**(7075):419–425, 2006.

[SLY21] Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. "Graph neural networks for friend ranking in large-scale social platforms." In Proceedings of the Web Conference 2021, pp. 2535–2546, 2021.

[Sob75] AI Sobolev. "Characterization of the principle of optimality for cooperative games through functional equations." Mathematical Methods in the Social Sciences, Vipusk, **6**:92–151, 1975.

[SP19] Donghee Shin and Yong Jin Park. "Role of fairness, accountability, and transparency in algorithmic affordance." Computers in Human Behavior, **98**:277–284, 2019.

[SSD02] Francis W Starr, Srikanth Sastry, Jack F Douglas, and Sharon C Glotzer. "What do we learn from the local geometry of glass-forming liquids?" Physical review letters, **89**(12):125501, 2002.

[SUG21] Kristof Schütt, Oliver Unke, and Michael Gastegger. "Equivariant message passing for the prediction of tensorial properties and molecular spectra." In International Conference on Machine Learning, pp. 9377–9388. PMLR, 2021.

[SW15] BA Sun and WH Wang. "The fracture of bulk metallic glasses." Progress in Materials Science, **74**:211–307, 2015.

[SYS20] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. "A deep learning approach to antibiotic discovery." Cell, **180**(4):688–702, 2020.

[TD22] Philipp Thölke and Gianni De Fabritiis. "Torchmd-net: Equivariant transformers for neural network based molecular potentials." arXiv preprint arXiv:2202.02541, 2022.

[Tel94] Lester G Telser. "The usefulness of core theory in economics." Journal of Economic Perspectives, **8**(2):151–164, 1994.

[TLH22] Xianfeng Tang, Yozen Liu, Xinran He, Suhang Wang, and Neil Shah. "Friend Story Ranking with Edge-Contextual Local Graph Convolutions." In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, pp. 1007–1015, 2022.

[TLM21] Longwen Tang, Han Liu, Gang Ma, Tao Du, Normand Mousseau, Wei Zhou, and Mathieu Bauchy. "The energy landscape governs ductility in disordered materials." Materials Horizons, **8**(4):1242–1252, 2021.

[TLS20] Xianfeng Tang, Yozen Liu, Neil Shah, Xiaolin Shi, Prasenjit Mitra, and Suhang Wang. "Knowing your fate: Friendship, action and temporal explanations for user engagement prediction on social apps." In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 2269–2279, 2020.

[TML20] Longwen Tang, Gang Ma, Han Liu, Wei Zhou, and Mathieu Bauchy. "Bulk metallic glasses' response to oscillatory stress is governed by the topography of the energy landscape." The Journal of Physical Chemistry B, **124**(49):11294–11298, 2020.

[TT10] Morgana Martin Trexler and Naresh N Thadhani. "Mechanical properties of bulk metallic glasses." Progress in Materials Science, **55**(8):759–839, 2010.

[TZY08] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. "Arnetminer: extraction and mining of academic social networks." In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 990–998, 2008.

[VCC17] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. "Graph attention networks." arXiv preprint arXiv:1710.10903, 2017.

[VT20] Minh Vu and My T. Thai. "PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks." In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pp. 12225–12235. Curran Associates, Inc., 2020.

[WDG23] Zhengxuan Wu, Karel D'Oosterlinck, Atticus Geiger, Amir Zur, and Christopher Potts. "Causal proxy models for concept-based model explanations." In International conference on machine learning, pp. 37313–37334. PMLR, 2023.

[WDM20] Qi Wang, Jun Ding, and Evan Ma. "Predicting the propensity for thermally activated β events in metallic glasses via interpretable machine learning.", 2020.

[Wer06] Sebastian Wernicke. "Efficient detection of network motifs." IEEE/ACM transactions on computational biology and bioinformatics, **3**(4):347–359, 2006.

[WPC20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. "A comprehensive survey on graph neural networks." IEEE transactions on neural networks and learning systems, **32**(1):4–24, 2020.

[WRF18] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. "MoleculeNet: A Benchmark for Molecular Machine Learning.", 2018.

[WS98] Duncan J Watts and Steven H Strogatz. "Collective dynamics of 'small-world'networks." nature, **393**(6684):440–442, 1998.

[WSZ20] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. "Graph neural networks in recommender systems: a survey." ACM Computing Surveys (CSUR), 2020.

[WWK08] Nikil Wale, Ian A Watson, and George Karypis. "Comparison of descriptor spaces for chemical compound retrieval and classification." Knowledge and Information Systems, **14**(3):347–375, 2008.

[WWX19] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. "Explainable reasoning over knowledge graphs for recommendation."

In Proceedings of the AAAI conference on artificial intelligence, volume 33, pp. 5329–5336, 2019.

[XBG23] Han Xuanyuan, Pietro Barbiero, Dobrik Georgiev, Lucie Charlotte Magister, and Pietro Liò. "Global concept-based interpretability for graph neural networks via neuron analysis." In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pp. 10675–10683, 2023.

[XFL18] Bin Xu, Michael L Falk, JF Li, and LT Kong. "Predicting shear transformation events in metallic glasses." Physical review letters, **120**(12):125503, 2018.

[XHL18] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How powerful are graph neural networks?" arXiv preprint arXiv:1810.00826, 2018.

[YBC24] Botao Yu, Frazier N. Baker, Ziqi Chen, Xia Ning, and Huan Sun. "LlaSMol: Advancing Large Language Models for Chemistry with a Large-Scale, Comprehensive, High-Quality Instruction Tuning Dataset.", 2024.

[YBY19] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. "Gnnexplainer: Generating explanations for graph neural networks." Advances in neural information processing systems, **32**:9240, 2019.

[YCA20] Asano YM., Rupprecht C., and Vedaldi A. "Self-labelling via simultaneous clustering and representation learning." In International Conference on Learning Representations (ICLR), 2020.

[YCS20] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. "Graph contrastive learning with augmentations." Advances in Neural Information Processing Systems, **33**, 2020.

[YG22] Zhaoning Yu and Hongyang Gao. "Molecular representation learning via heterogeneous motif graph neural networks." In International Conference on Machine Learning, pp. 25581–25594. PMLR, 2022.

[YHC18] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. "Graph convolutional neural networks for web-scale recommender systems." In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 974–983, 2018.

[YKA20] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. "On completeness-aware concept-based explanations in deep neural networks." Advances in neural information processing systems, **33**:20554–20565, 2020.

[YSW12] Hai-Bin Yu, Konrad Samwer, Y Wu, and Wei Hua Wang. "Correlation between β relaxation and self-diffusion of the smallest constituting atoms in metallic glasses." Physical review letters, **109**(9):095508, 2012.

[YWG18] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding." Advances in neural information processing systems, **31**, 2018.

[YYG20] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. "Explainability in graph neural networks: A taxonomic survey." arXiv preprint arXiv:2012.15445, 2020.

[YYG22] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. "Explainability in graph neural networks: A taxonomic survey." IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022.

[YYW21] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. "On Explainability of Graph Neural Networks via Subgraph Explorations." In Marina Meila and Tong Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pp. 12241–12252. PMLR, 18–24 Jul 2021.

[ZC04] Afra Zomorodian and Gunnar Carlsson. "Computing persistent homology." In Proceedings of the twentieth annual symposium on Computational geometry, pp. 347–356, 2004.

[ZC18] Muhan Zhang and Yixin Chen. "Link prediction based on graph neural networks." Advances in neural information processing systems, **31**, 2018.

[ZC20] Yongfeng Zhang and Xu Chen. "Explainable Recommendation: A Survey and New Perspectives." Foundations and Trends® in Information Retrieval, **14**(1):1–101, 2020.

[ZCH20] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph neural networks: A review of methods and applications." AI Open, **1**:57–81, 2020.

[ZCM23] Jiaxing Zhang, Zhuomin Chen, Hao Mei, Dongsheng Luo, and Hua Wei. "RegExplainer: Generating Explanations for Graph Neural Networks in Regression Task." arXiv preprint arXiv:2307.07840, 2023.

[Zha23] Jiawei Zhang. "Graph-ToolFormer: To Empower LLMs with Graph Reasoning Ability via Prompt Augmented by ChatGPT." arXiv preprint arXiv:2304.11116, 2023.

[ZHS20] Shichang Zhang, Ziniu Hu, Arjun Subramonian, and Yizhou Sun. "Motif-driven contrastive learning of graph representations." arXiv preprint arXiv:2012.12533, 2020.

[ZJS21] Tong Zhao, Tianwen Jiang, Neil Shah, and Meng Jiang. "A synergistic approach for graph anomaly detection with pattern mining and feature learning." IEEE Transactions on Neural Networks and Learning Systems, 2021.

[ZLS22] Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. "GStarX: Explaining Graph Neural Networks with Structure-Aware Cooperative Games.", 2022.

[ZLW22] Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. "Learning from Counterfactual Links for Link Prediction." In International Conference on Machine Learning, pp. 26911–26926. PMLR, 2022.

[ZLX20] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. "Revisiting graph neural networks for link prediction." Openreview, 2020.

[ZSZ19] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. "Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems." arXiv preprint arXiv:1905.13129, 2019.

[ZWH23] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, et al. "Artificial intelligence for science in quantum, atomistic, and continuum systems." arXiv preprint arXiv:2307.08423, 2023.

[ZZX21] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. "Neural bellman-ford networks: A general graph neural network framework for link prediction." Advances in Neural Information Processing Systems, **34**:29476–29490, 2021.