Explainable Mining of Graphs and Time Series: Algorithms and Applications

Meng-Chieh Lee

June 2025
CMU-CS-25-119

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Christos Faloutsos, Co-Chair
Leman Akoglu, Co-Chair
Geoffrey Gordon
Nina Mishra, Amazon

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Support: This research was supported in part by the the Pennsylvania Infrastructure Technology Alliance, Lehigh University, the Commonwealth of Pennsylvania's Department of Community and Economic Development, a gift from PNC, Fredkin Chair, and the National Institute of Food and Agriculture under Grant No. 2024-67021-43696. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Graph Mining, Network Effects, Random Walks, Belief Propagation, Node Classification, Graph Neural Networks, Linear Graph Neural Networks, Link Prediction, Information Theory, Usable Information, Large Language Models, Large Language Model Agents, Retrieval Augmented Generation, Knowledge Graphs, Graph Anomaly Detection, Minimum Description Length, Frequent Pattern Mining, Learnable Graph Kernel, Kernel Convolution Networks, Graph Property Prediction, Human-Trafficking Detection, Time Series Analysis, Anomaly Detection, Time Series Anomaly Detection, Self-Supervised Learning, Hyperparameter Optimization, Seizure Detection, Group Anomaly Detection

"The mind fears the heart, but the heart doesn't mind.
No I may not be perfect, but I'm loving this life.
...
I'm a million miles ahead of where I'm from,
But there's still another million miles to come."

— Tim Bergling (a.k.a. Avicii), "Trouble"

Abstract

Given a social network graph, how can we predict connections between users and determine whether they are based on shared hobbies or common friends? Given a database containing molecular graphs, how can we determine whether the graphs inhibit HIV replication based on substructures they frequently share? Similarly, in time series data from EEG recording, how can we identify seizures and explain why they are considered abnormal? Although recent machine learning methods have shown improved performance, many remain black-box models, making explainability challenging. This leads us to explainable artificial intelligence (XAI), which offers valuable insights through its explanations and is more practical for deployment in real-world applications.

In this thesis, we focus on developing explainable machine learning methods tailored for graphs and time series. Each method we propose is either inherently explainable, or designed to automatically provide data analysis or justification for its decisions. In each part, we present effective and general algorithms, and explore a broad range of applications.

In the first part, we focus on node-level graph mining. We propose algorithms to analyze various types of information in graphs, e.g., the network effects of the graph structure, and the usable information in the node features. Our proposed linear methods are not only inherently interpretable and fast, but also outperform baselines in solving node classification and link prediction tasks. In node classification, our method improves the accuracy by *10.3%* over the second-best baseline, while being *2.5 times* faster. In link prediction, our method achieves an *average rank 1.1*, outperforming baselines on *11 out of 12* real-world datasets. In the application of graph retrieval-augmented generation, our agentic method achieves an average relative improvement of *51%*.

In the second part, we focus on graph-level graph mining. We discover frequent substructures using the minimum description length (MDL) principle and learnable graph kernels. In graph anomaly detection, our MDL-based method is *58 times* faster than the second-best baseline, while achieving *1.3 times* higher average precision. In graph regression, our method with learnable graph kernels improves the mean absolute error by *14.3%*. In the application of human trafficking detection, our method detects human trafficking ads with *84%* precision, while requiring only *8 hours* to process *4 million* documents.

In the third part, we focus on time series mining, with an emphasis on time series anomaly detection. Our self-supervised method effectively identifies the ground truth hyperparameters of anomalies in time series data, resulting in an *average rank 2.2* compared to the baselines. In the application involving medical EEG signals, unlike traditional point anomaly detection methods, we focus on identifying group anomalies that occur within a short period and exhibit similar abnormal patterns. Our method is fast and scalable, and discovers and ranks both point and group anomalies in *2 minutes* for *1 million* data points on a stock machine.

Acknowledgments

No words can truly convey my gratitude to my advisor, Christos Faloutsos. He continually inspires me through his boundless enthusiasm and unwavering passion for research. Throughout my Ph.D., he consistently provided patient guidance and actively engaged with me in brainstorming research ideas. Even on weekends, he remains readily accessible. Nevertheless, he means much more to me than just an advisor. The lessons I learned from him also guided me through the obstacles in life. He once told me that doing research is not about writing a paper that everyone likes, but about delivering work that I would be proud of. His words continue to encourage me to follow my heart and stay true to myself.

I also want to thank my other thesis committee members: Leman Akoglu, Geoffrey Gordon, and Nina Mishra, who provided me with much insightful advice. I especially thank my co-advisor Leman, who is one of the smartest people I have ever met. She finds solutions quickly and reliably, no matter what research problems we face. I deeply appreciate her guidance throughout our collaboration and have gained invaluable knowledge from her.

I feel incredibly fortunate to be working with so many talented people. As senior students in the group, Shubhranshu and Jaemin have mentored me, helping me grow in every aspect, and I am truly grateful to them. They are also great friends and I really enjoy being around them. I thank my other dependable colleagues in the group: Saranya, Catalina, Minji, Karish, Namyong, Robson, Mirela, Yue, Xueying, and Lingxiao. We share many wonderful memories, from taking classes together to collaborating on research. I also thank those amazing collaborators during my internships: Yonatan, Nina, Jian, Xiang, Da, Qi, Costas, Zhen, Soji, Vassilis, and Huzefa. Finally, I want to thank my advisor from my master's program, Vincent S. Tseng, and my mentor, Yu Huang, both of whom have played a significant role in shaping my development as a researcher.

My parents spare no effort in supporting me as I pursue a dream that others deem unrealistic. My father opens my eyes to the world. His remarkable career achievements have helped me realize that the whole world is my stage. My mother teaches me what it means to be kind, and no matter what, she always has my back. She once told me, "Raising you is not about getting anything in return; it is about supporting you in experiencing what you should in your one and only life." They dedicate everything to their children, and in my eyes, they are perfect.

I want to dedicate this final paragraph to my wife, Ruby. I am the luckiest person in the world to have her by my side. We share both joy and tears together, supporting each other through every high and low. Because of her, I find the strength to keep going, regardless of how difficult things get. I am confident that we will create even more happiness and overcome even greater challenges in life together.

My heartfelt thanks go to everyone who has supported me along the way. I could not have done this without each and every one of you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the last decade, many effective machine learning (ML) and deep learning methods have been proposed to solve a variety of problems in graphs and time series. However, the majority of the methods are designed to optimize performance, often neglecting the importance of model transparency. In other words, these black-box methods are neither inherently explainable nor provide explanations for their decisions.

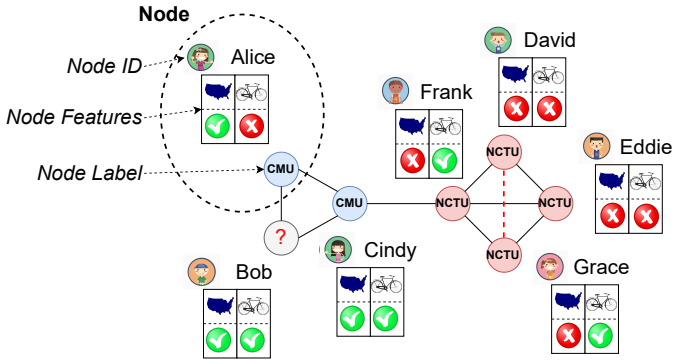
For this reason, in recent years, explainable artificial intelligence (XAI) has gained a lot of attention. These approaches are designed not only to provide explanations but also to remain effective. XAI paves the way for ML methods to be adopted in the real world, especially in domains that require well-justified solutions. These domains include legal, medical, financial, and so on. For example, if an ML method is developed to help a physician make medical decisions, it must provide explanations. It is crucial that the method and the doctor complement each other by allowing the doctor to understand why the decision was made. Another example is financial data, where an explainable method can provide insights into suspicious activities, allowing domain experts to investigate further.

Among the many types of data, graphs and time series are two of the most common in the real world. Graphs, including social networks and financial networks, have been applied to numerous applications to identify the properties of individual nodes and the relationships between nodes (Figure 1.1a). Meanwhile, many applications operate on databases containing multiple graphs (Figure 1.1b), such as anomaly detection and molecular property prediction. Similarly, time series data have been widely used in monitoring applications in various systems, such as server machine metrics and EEG recordings, enabling us to detect anomalies in time series (Figure 1.1c). Hence, the research questions we aim to answer are:

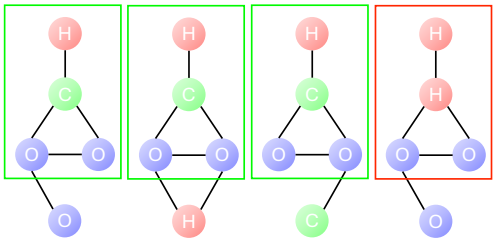
RQ1. **Node-Level Graph Mining:** How can we determine the information in the graph that is useful for solving node-level graph tasks effectively?

RQ2. **Graph-Level Graph Mining:** How can we identify frequent substructures in a graph database and leverage them for downstream graph-level graph tasks?

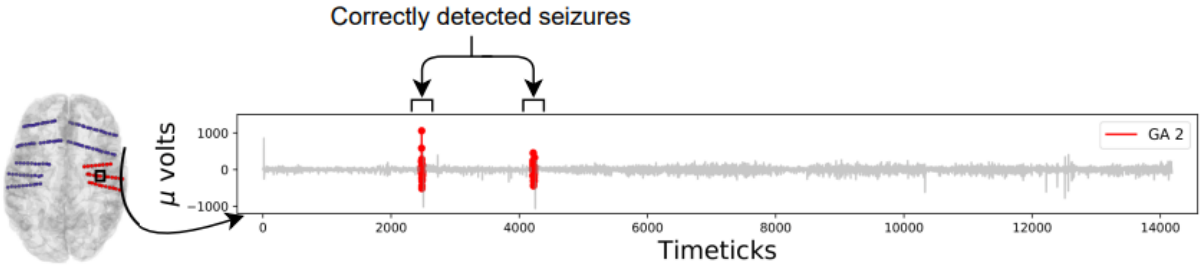
RQ3. **Time Series Mining:** How can we detect anomalies in time series data?



(a) Part I: Node Classification and Link Prediction



(b) Part II: Frequent Substructure Mining



(c) Part III: Seizure Detection in EEG Recording

Figure 1.1: **Applications on graphs and time series.** (a) **Node-Level Graph Mining** (Part I): We solve tasks such as node classification and link prediction in an attributed graph. (b) **Graph-Level Graph Mining** (Part II): We detect frequent substructures in a graph database and leverage them to solve tasks, such as graph anomaly detection and graph regression. (c) **Time Series Mining** (Part III): We detect anomalous time periods in a time series data.

1.2 Overview and Contributions

In this thesis, we focus on introducing explainable ML methods tailored for graphs and time series. Our methods are carefully designed to be either:

1. *Inherently explainable*, such as linear models, or
2. *Capable of providing explanations* for the dataset or the decisions they make.

More specifically, we develop algorithms to address fundamental ML problems, as well as solve a diverse range of real-world applications with domain-specific insights. As shown in Table 1.1, our contributions can be divided into three topics: node-level graph mining (Part I), graph-level graph mining (Part II), and time series mining (Part III); as well as two categories: algorithm and application. We always support the necessity of the designs in our proposed methods through careful data analysis or systematic analysis of existing algorithms.

Topic	Category	Task	Method	Chapter	Link
Part I: *Node*-Level Graph Mining	Algorithm	Node Classification	NetEffect	§ 3	[PDF]
			SlimG	§ 4	[PDF]
		" & Link Prediction	NetInfoF	§ 5	[PDF]
	Application	Graph RAG	HybGRAG	§ 6	[PDF]
Part II: *Graph*-Level Graph Mining	Algorithm	Graph Anomaly Detection	GAWD	§ 7	[PDF]
		" & Graph Regression	RWK$^+$	§ 8	[PDF]
	Application	Human Trafficking Detection	DeltaShield	§ 9	[PDF]
Part III: Time Series Mining	Algorithm	Anomaly Detection	TSAP	§ 10	[PDF]
	Application	Seizure Detection	gen^{2}Out	§ 11	[PDF]

Table 1.1: **Thesis Overview.**

1.2.1 Part I: Node-Level Graph Mining

Algorithm In Chapter 3, we propose NetEffect, including a statistical test to identify if there are network effects (i.e. homophily, heterophily, both or none) in the given graph without node features. Thanks to our estimated compatibility matrix, NetEffect is *12.9%* more accurate and *3.4*× faster for node classification. In Chapter 4, we further consider the node features and propose SlimG, a linear graph neural network (GNN) that effectively classifies the nodes in both homophily and heterophily graphs. Compared to non-linear GNNs, SlimG is explainable, while being *10.3%* more accurate and *2.5*× faster. In Chapter 5, we extend this idea and propose NetInfoF to measure and exploit usable information in graphs for both node classification and link prediction. NetInfoF is the first linear GNN that is generalized to link prediction and achieves an *average rank 1.1* among state-of-the-art baselines.

Application In Chapter 6, we propose HybGRAG, a retrieval-augmented generation (RAG) method designed to handle "hybrid" questions that require both relational and textual information to be answered. By addressing the fundamental challenges we identified, HybGRAG achieves an average relative improvement of *51%*.

Impact NetInfoF was accepted for spotlight presentation (top 5%) at ICLR 2024.

1.2.2 Part II: Graph-Level Graph Mining

Algorithm In Chapter 7, we propose GAWD, a minimum description length (MDL) method that identifies anomalous graphs in a database containing many graphs, based on frequent substructure mining. GAWD is up to *58*× faster, while being *1.3*× better in average precision. In Chapter 8, we further explore learning frequent substructures with graph kernels and propose an improved random walk kernel, RWK$^+$, which can be used to extract structural features from

the graph database. Applied to a large graph regression benchmark, a model using features extracted by RWK$^+$ outperforms the baseline by *14.3%* in mean absolute error.

Application In Chapter 9, we propose DeltaShield, which detects shared templates across millions of escort advertisements based on MDL and allows incremental updates by representing them as graphs. DeltaShield detects human-trafficking advertisements with *84%* precision, while requiring only *8 hours* for *4 million* documents.

Impact DeltaShield received media coverage from WPXI[†].

1.2.3 Part III: Time Series Mining

Algorithm In Chapter 10, we introduce TSAP that automatically fine-tunes the best hyperparameters for creating pseudo-time-series anomalies, to learn the anomaly detector in a self-supervised manner. TSAP achieves an *average rank 2.2* in F1 score and identifies the true hyperparameters of anomalies.

Application In Chapter 11, we propose gen^{2}Out to detect seizures (group anomalies) in time series EEG recordings and to distinguish them from irrelevant noise (point anomalies). gen^{2}Out is the first to detect both types of anomaly and takes only *2 minutes* to run on *1 million* data points.

[†]https://www.wpxi.com/news/top-stories/carnegie-mellon-university-researchers-develop-human-trafficking-algorithm/4OGNYHSW3VHZDOGFVC2Y6QHETI/

Chapter 2

Background

2.1 Explainability

Many explanation methods have been proposed to improve the explainability of machine learning models [Lip18, DBH18, BH21], including feature importance-based methods [RSG16, LL17, RSG18], counterfactual explanations [WMR17, VDH20], and contrastive explanations [Mil21, JSR+21]. However, most methods remain post-hoc, interpreting existing black-box models after they have been trained and evaluated. In contrast, our proposed methods in this thesis not only achieve state-of-the-art performance on the given tasks, but also generate faithful explanations that are integrated into the task-solving process.

We compare our proposed methods with existing explanation methods in Table 2.1 and show that our methods fulfill all desired properties, including faithfulness, explicitness, and stability [AJ18]. Faithfulness denotes whether the explanation accurately reflects the decision-making process; explicitness denotes whether the explanation clearly demonstrates the reasoning behind a decision; stability denotes whether small perturbations in the input lead to different explanations. A key distinction is that our proposed methods are capable of solving the task and achieving state-of-the-art performance, while most explanation methods are post-hoc and not designed to perform the task, making comparison of explanations less meaningful, as the methods serve fundamentally different purposes.

Whereas most post-hoc explanation methods are not considered faithful [Rud19], our methods generate explanations that are integrated into the task-solving process and are therefore considered faithful by design. For example, SlimG (Chapter 4) and NetInfoF (Chapter 5) are linear models, where the learned coefficients directly reflect the influence of input features. As argued by [AJ18], linear methods are faithful. Similarly, DeltaShield (Chapter 9) and gen^{2}Out (Chapter 11) provide faithful explanations by explicitly visualizing the intermediate steps of the task-solving process; scatter-plots have been successfully used before for explanations, such as in the LookOut paper [GES+18].

Table 2.1: **Our explainable ML methods match all specs**, while baselines miss one or more. '?' denotes that it depends on specific methods.

Property	Feature-Importance [RSG16, LL17, RSG18]	Counterfactual [WMR17, VDH20]	Contrastive [Mil21, JSR+21]	**Our Methods**
Integrated vs. Post-Hoc	Post-Hoc	(Mostly) Post-Hoc	(Mostly) Post-Hoc	Integrated
1. Faithful		?		✔
2. Explicit	✔	✔	✔	✔
3. Stable		?	?	✔
4. Task-Solving				✔

2.2 Graphs

Definition 2.1: Homogeneous graph

A homogeneous graph G is a data structure that consists of nodes $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where both the nodes and edges are of the same type.

For example, in a social network graph, the nodes represent the users, and the edges indicate whether two users are friends. The structure of a graph with $n = |\mathcal{V}|$ nodes can be represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $\mathbf{A}_{uv} = 1$ if $(u, v) \in \mathcal{E}$, and 0 otherwise. Each node i has a unique label $l(i) \in \{1, \ldots, c\}$, where c is the number of classes. The set of labels $\mathbf{y} = \{l(1), \ldots, l(n)\}$ can be represented by a one-hot matrix $\mathbf{Y} \in \mathbb{R}^{n \times c}$. Graphs that do not contain node features are called *featureless* graphs, while graphs that contain node features or attributes $\mathbf{X} \in \mathbb{R}^{n \times f}$, where f is the number of features, are called *node-attributed* graphs.

Definition 2.2: Knowledge graph

A knowledge graph (KG) is a heterogeneous graph that contains triplets $K = \{(u, r, t) \mid u, t \in \mathcal{N}, r \in \mathcal{R}\}$, where $\mathcal{N}$ and $\mathcal{R}$ denote the sets of entities (nodes) and relations (edges), respectively, and each entity $u \in \mathcal{N}$ has a type $t(u) \in \mathcal{T}$.

For example, in the Wikipedia KG, a triplet can represent a fact such as {Washington D.C., *is-capital-of,* USA}, where the type of head entity is *city* and the type of tail entity is *country*.

2.3 Graph Databases

Definition 2.3: Graph database

A graph database consists of I graphs $\mathcal{G} = \{G_1, \ldots, G_I\}$, where each graph $G_i(\mathcal{V}_i, \mathcal{E}_i)$ has a set of nodes $\mathcal{V}_i$ and a set of edges $\mathcal{E}_i$.

For example, in a molecule graph database, each graph represents a molecule, where the nodes are atoms, and the edges are chemical bonds. If the database is node-labeled and weighted, each node $v \in \mathcal{V}_i$ has a label $l(v) \in \mathcal{T}$, where $\mathcal{T}$ is the set of unique node labels, and each edge $(u, v) \in \mathcal{E}_i$ is associated with a weight $w(u, v)$. If the database is node-attributed, each graph G_i is associated with a node feature matrix $\mathbf{X}_{G_i} \in \mathbb{R}^{|\mathcal{V}_i| \times f}$, where f is the feature dimension.

2.4 Time Series Anomaly Detection

Definition 2.4: Univariate time series

A univariate time series $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ is a sequentially ordered collection of T data points, where $x_i \in \mathbb{R}$ corresponds to a scalar observation at time step i.

For example, a time series can represent an EEG recording, or the memory utilization of a monitoring server, collected every second.

Anomaly detection in time series can be divided into two categories: sequence-level and point-level. In sequence-level anomaly detection, given a set of time series $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, the task is to detect anomalous time series by assigning a label $y_i \in \{-1, +1\}$ to $\mathbf{x}_i \in \mathcal{X}$. In point-level anomaly detection, given a time series $\mathbf{x}$, the task is to detect anomalous time points by assigning labels $\mathbf{y} = \{y_1, \ldots, y_T\}$ to $\mathbf{x}$, where $y \in \{-1, +1\}$.

2.5 Linear Algebra Preliminaries

This section reviews key linear algebra operations used throughout this thesis.

Definition 2.5: Hadamard product

The Hadamard product for two matrices $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\mathbf{Y} \in \mathbb{R}^{m \times n}$ is as follows:

$$(\mathbf{X} \odot \mathbf{Y})_{ij} = \mathbf{X}_{ij}\mathbf{Y}_{ij}, \tag{2.1}$$

where $\mathbf{X}_{ij}$ is the element at row i and column j.

Definition 2.6: Vectorization

The vectorization for a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ is as follows:

$$\text{vec}(\mathbf{X}) = [\mathbf{X}_{11}, \cdots, \mathbf{X}_{m1}, \mathbf{X}_{12}, \cdots, \mathbf{X}_{m2}, \cdots, \mathbf{X}_{mn}]^\intercal, \tag{2.2}$$

where $\mathbf{X}_{ij}$ is the element at row i and column j.

Definition 2.7: Kronecker product

The Knronecker product for two matrices $\mathbf{X} \in \mathbb{R}^{m \times n}$ and $\mathbf{Y} \in \mathbb{R}^{p \times q}$ is as follows:

$$\mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} \mathbf{X}_{11}\mathbf{Y} & \mathbf{X}_{12}\mathbf{Y} & \cdots & \mathbf{X}_{1n}\mathbf{Y} \\ \mathbf{X}_{21}\mathbf{Y} & \mathbf{X}_{22}\mathbf{Y} & \cdots & \mathbf{X}_{2n}\mathbf{Y} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_{m1}\mathbf{Y} & \mathbf{X}_{m2}\mathbf{Y} & \cdots & \mathbf{X}_{mn}\mathbf{Y} \end{bmatrix}, \tag{2.3}$$

where $\mathbf{X}_{ij}$ is the element at row i and column j, and the size of $\mathbf{X} \otimes \mathbf{Y}$ is $pm \times qn$.

Definition 2.8: Mixed Kronecker matrix-vector product property

The mixed Kronecker matrix-vector product for matrices $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ is as follows:

$$(\mathbf{X} \otimes \mathbf{Y})\text{vec}(\mathbf{Z}) = \text{vec}(\mathbf{Y}\mathbf{Z}\mathbf{X}^\intercal) \tag{2.4}$$

2.6 Acronyms

Table 2.2 lists the acronyms used in this thesis, sorted in alphabetical order.

Table 2.2: Table of Acronyms.

Acronym	Definition
AP	Average Precision
ARI	Adjusted Rand Index
AUC	Area Under Curve
BP	Belief Propagation
ES	Early-Stopping
GCN	Graph Convolutional Networks
GNE	Generalized Network-Effects
GNN	Graph Neural Network
GRAG	Graph Retrieval-Augmented Generation
HQA	Hybrid Question Answering
HT	Human Trafficking
ICL	In-Context Learning
IF	Isolation Forest
KBQA	Knowledge Base Question Answering
KCN	Kernel Convolution Network
KG	Knowledge Graph
LLM	Large Language Model
LR	Logistic Regression
MAE	Mean Absolute Error
MDL	Minimum Description Length
ML	Machine Learning
MSA	Multiple Sequence Alignment
NUI	Network Usable Information
ODQA	Open-Domain Question Answering
PCA	Principal Component Analysis
POA	Partial Order Alignment
PPR	Personalized PageRank
RAG	Retrieval-Augmented Generation
ROC	Receiver Operating Characteristic
RWK	Random Walk Kernel
SGC	Simple Graph Convolution
SKB	Semi-Structured Knowledge Base
SSL	Self-Supervised Learning
SVD	Singular Value Decomposition
TSAD	Time Series Anomaly Detection
VSS	Vector Similarity Search
XAI	Explainable Artificial Intelligence

Part I

Node-Level Graph Mining

Overview of Part I

Given a graph, how can we solve node classification and link prediction?
How can we identify which information in the graph is useful for the given node-level tasks?

In a graph, the structure may or may not be useful for inferring node labels. Similarly, in an attributed graph, node features are not necessarily useful for the task. In this chapter, given a graph, we want an explanation of whether graph tasks can be effectively solved. In other words, we aim to determine which aspects of the graph, such as its structure and node features, are useful for the task.

We address this problem in three different settings of increasing difficulty and propose three corresponding **algorithms**:

- § 3: Node classification in a featureless graph – NetEffect solves it and provides a statistical test to determine whether nodes with different class labels are randomly connected.
- § 4: Node classification in an attributed graph – SlimG solves it by identifying whether the graph structure or node features are useful based on the learned weights.
- § 5: Node classification and link prediction in an attributed graph – NetInfoF solves both and quantitatively measures how informative the graph structure and node features are.

We further extend this problem to a real-world graph **application**:

- § 6: In graph retrieval-augmented generation (GRAG), HybGRAG determines if a question should be answered by the information from knowledge graphs or text documents.

Chapter 3

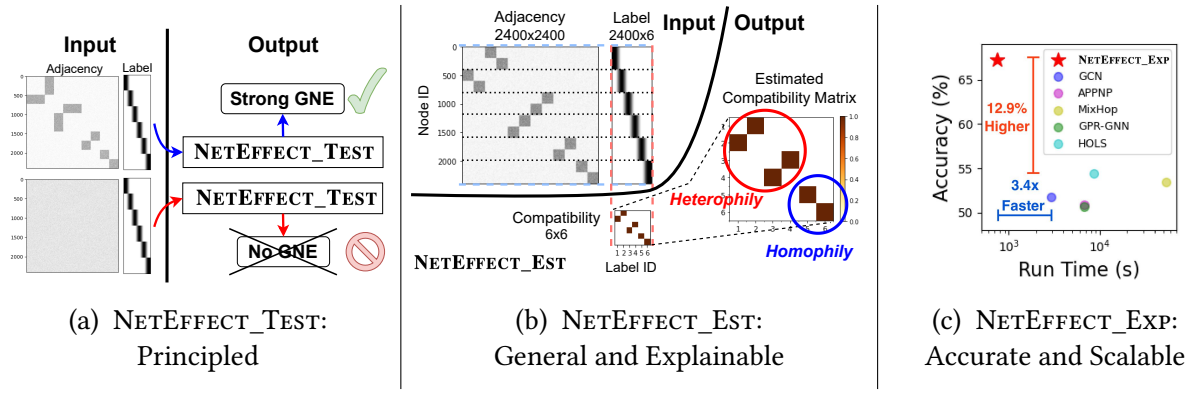
NetEffect: Discovery and Exploitation of Generalized Network Effects

Chapter based on work that appeared at PAKDD 2024 [LSYF24] [PDF].

Given a large graph with few node labels, how can we (a) identify whether there is *generalized network-effects* (GNE) or not, (b) estimate GNE to explain the interrelations among node classes, and (c) exploit GNE efficiently to improve the performance on downstream tasks? The knowledge of GNE is valuable for various tasks like node classification and targeted advertising. However, identifying GNE such as homophily, heterophily or their combination is challenging in real-world graphs due to limited availability of node labels and noisy edges.

In this chapter, we propose NetEffect, a graph mining approach to address the above issues, enjoying the following properties: (i) *Principled*: a statistical test to determine the presence of GNE in a graph with few node labels; (ii) *General* and *Explainable*: a closed-form solution to estimate the specific type of GNE observed; and (iii) *Accurate* and *Scalable*: the integration of GNE for accurate and fast node classification.

Applied on real-world graphs, NetEffect discovers unexpected absences of GNE in numerous graphs, which were recognized to exhibit heterophily. Further, we show that incorporating GNE is effective on node classification. On a million-scale graph, NetEffect achieves over $7\times$ speedup (*14 minutes* vs. 2 hours) compared to most competitors.



(a) NetEffect_Test: Principled

(b) NetEffect_Est: General and Explainable

(c) NetEffect_Exp: Accurate and Scalable

Figure 3.1: **NetEffect works well in node classification**, thanks to its three novel contributions: (a) NetEffect_Test statistically tests the existence of GNE. (b) NetEffect_Est explains the graph with the x-ophily compatibility matrix. (c) NetEffect_Exp wins and is fast.

3.1 Introduction

Given a large graph with few node labels and no node features, how can we check whether the graph structure is useful for classifying nodes or not? Node classification is often used to infer labels on large real-world graphs. Since manual labeling is expensive, it is common that only a few node labels are available. For example, in a million-scale social network, identifying even a fraction (say 5%) of users' groups is prohibitive, limiting the application of methods that assume many labels are given. With the prevalence of graphs in industry and academia alike, there is a growing need among users to know whether these graph structures provide meaningful information for inference tasks. Therefore, before investing a huge amount of time and resources into potentially unsuccessful experiments, a preliminary test is earnestly needed.

That is to say, we want to know whether the given graph has *generalized network-effects* (GNE) or not. A graph with GNE provides meaningful information through the structure that can be used to identify the labels of nodes. For example, "engineers tend to have engineer friends" denotes homophily, while "talkative person tends to make friends with silent ones" denotes heterophily. It is thus important to distinguish which GNE the graph has, i.e., homophily, heterophily, or both (which we call "x-ophily"), if there is any. We define x-ophily as follows:

Definition 3.1: x-ophily Generalized Network-Effects

An x-ophily graph exhibits both homophily and heterophily *generalized network-effects* (GNE).

For example, in an x-ophily social network, lawyers only make friends with other lawyers (homophily), while managers are friends with engineers but not with other managers (heterophily). Given c classes, an intuitive way to describe GNE is via a $c \times c$ compatibility matrix, which shows the relative influence between each class pair. It can be used to explain the graph property, as well as be exploited to better assign the labels in the graph.

However, identifying GNE is commonly neglected in literature: inference-based methods such as belief propagation (BP) [WGKM18, EKF20, GGKF15, WG20] often assume that the relationship is given by domain experts; most graph neural networks (GNNs) assume homophily [KW17, KBG19, WJZ+19]. Although some works [LHL+21, MLST22, ZYZ+20] use homophily statistics to analyze graphs, our work is very different for three reasons. First, they are designed to identify the absence of homophily, and thus cannot clearly distinguish GNE, which includes different non-homophily cases, i.e., heterophily, both, or no GNE. Second, to compute accurate statistics, they use all the node labels in the graph, which is impractical during node classification. Finally, their analyses rely heavily on the results of GNNs, which means in addition to the graph structure, the node features also significantly influence the conclusions of GNE. In contrast, our work aims to answer following research questions:

RQ1. **Hypothesis Testing**: How to identify whether the given graph has GNE or not, with only few labels?

RQ2. **Estimation**: How to estimate GNE in a principled way, and explain the patterns of node connection based on the estimation?

RQ3. **Exploitation**: How to effectively and efficiently exploit GNE on node classification with only a few node labels?

We propose NetEffect, with 3 contributions as the corresponding solutions:

1. **Principled: NetEffect_Test** uses statistical tests to decide whether GNE exists at all. Figure 3.1a shows how it works, and Figure 3.2 shows its discovery, where many large real-world datasets known as heterophily graphs have little GNE.
2. **General and Explainable: NetEffect_Est** explains whether the graph is homophily, heterophily, or x-ophily by precisely estimating the compatibility matrix with the derived closed-form formula. In Figure 3.1b, it explains the interrelations of classes by the estimated compatibility matrix, which implies x-ophily.
3. **Accurate and Scalable: NetEffect_Exp** efficiently exploits GNE to perform better in node classification. It wins in both accuracy and time on a million-scale heterophily graph "Pokec-Gender", only requiring *14 minutes* (Figure 3.1c).

Reproducibility: Our code is publicly available at `https://github.com/mengchillee/NetEffect`.

3.2 Background and Related Work

Table 3.1 presents qualitative comparison of state-of-the-art approaches against our proposed NetEffect. Notice that only NetEffect fulfills all the specs.

3.2.1 Background

An overview of symbols and acronyms is provided in Table 3.2 and Table 3.3, respectively. Let G be an undirected, unweighted and featureless graph with n nodes, m edges and an adjacency matrix $\mathbf{A}$. Each node i has a unique label $l(i) \in \{1, \ldots, c\}$, where c is the number of classes. Let $\mathbf{E} \in \mathbb{R}^{n \times c}$ be the initial belief matrix with prior information, i.e., labeled nodes. $\mathbf{E}_{ik} = 1$ if $l(i) = k$, and $\mathbf{E}_{ik} = 0$ if $l(i) \neq k$. For nodes without labels, their entries are set to $1/c$. $\mathbf{H} \in \mathbb{R}^{c \times c}$

Table 3.1: **NetEffect matches all specs**, while baselines miss one or more. '?' and 'N/A' denote unclear and not applicable.

Property		BP [GGKF15, KKK+11]	HOLS [EKF20]	General GNNs [KW17, KBG19]	Het. GNNs [APK+19, CPLM21]	NetEffect
1. Principled	1.1. Statistical Test	✔	✔			✔
	1.2. Convergence Guarantee	✔	✔			✔
2. Explainable	2.1 Compatibility Matrix Estimation	N/A	N/A			✔
3. General	3.1 Handle Heterophily	?	?	?	✔	✔
	3.2 Handle GNE	?	?			✔
4. Scalable	4.1. Linear Complexity	✔		✔	✔	✔
	4.2. Thrifty	✔	✔	?	?	✔

is a row-normalized compatibility matrix, where $\mathbf{H}_{ku}$ is the relative influence of class k on class u. The residual of a matrix $\mathbf{E}$ around $1/c$ is $\hat{\mathbf{E}} = \mathbf{E} - 1/c \times \mathbf{1}$, where $\mathbf{1}$ is matrix of ones.

3.2.2 Belief Propagation

FaBP [KKK+11] and LinBP [GGKF15] accelerate belief propagation (BP) by approximating the final belief assignment. In particular, LinBP approximates the final belief as:

$$\hat{\mathbf{B}} = \hat{\mathbf{E}} + \mathbf{A}\hat{\mathbf{B}}\hat{\mathbf{H}}, \tag{3.1}$$

where $\hat{\mathbf{B}}$ is a residual final belief matrix, initialized with zeros. The compatibility matrix $\hat{\mathbf{H}}$ and initial beliefs $\hat{\mathbf{E}}$ are centered around $1/c$ to ensure convergence. HOLS [EKF20] is a BP-based method, which propagates the labels by weighing with higher-order cliques.

3.2.3 Analysis by Homophily Statistics

Many studies [LHL+21, MLST22, ZYZ+20] utilize homophily ratio to measure how common the labels of the connected node pairs share the same class. Our work focuses on very different aspects, as discussed in the introduction.

Table 3.2: Table of Symbols and Definitions.

Symbol	Definition
G	Undirected, unweighted and featureless graph
$l(i)$	Label of node i
n	Number of nodes
c	Number of classes
$\mathbf{A} \in \mathbb{R}^{n \times n}$	Adjacency matrix
$\mathbf{E} \in \mathbb{R}^{n \times c}$	Initial belief matrix
$\mathbf{B} \in \mathbb{R}^{n \times c}$	Final belief matrix
$\mathbf{H} \in \mathbb{R}^{c \times c}$	Compatibility matrix
$\hat{\cdot}$	Residual of a matrix, e.g., $\hat{\mathbf{E}} = \mathbf{E} - 1/c \times \mathbf{1}$
$\mathcal{E}$	Set of edges
$\mathcal{P}$	Set of priors
B	Number of rounds for NetEffect_Test
$\mathbf{F} \in \mathbb{R}^{c \times c}$	p-value table
$\mathbf{A}^* \in \mathbb{R}^{n \times n}$	Emphasis matrix
$\mathbf{W}' \in \mathbb{R}^{n \times n}$	Approximated proximity matrix
$\mathbf{D} \in \mathbb{R}^{n \times n}$	Degree matrix with node degrees along diagonal
$N(i)$	Neighbors of node i
L	Length of random walks
M	Trials of random walks
d	Rank for decomposition
$\mathbf{H}^* \in \mathbb{R}^{c \times c}$	Compatibility matrix estimated with $\mathbf{A}^*$
f	Scaling factor for propagation

Table 3.3: Table of Acronyms.

Acronym	Definition
GNE	Generalized Network-Effects
BP	Belief Propagation
SVD	Singular Value Decomposition
GNN	Graph Neural Network
GCN	Graph Convolutional Networks

3.2.4 Node Classification

GCN [KW17] and APPNP [KBG19] incorporate neighborhood information to do better predictions and assume homophily. MixHop [APK+19], GPRGNN [CPLM21], and H_2GCN [ZYZ+20] make no assumption of homophily. Nevertheless, H_2GCN requires too much memory and thus can not handle large graphs. LINKX [LHL+21] introduces multiple large heterophily datasets, but it is not applicable to graphs without node features.

3.3 Proposed GNE Test

Given a graph with few labels, how can we identify whether the graph has *generalized network-effects* (GNE) or not? In other words, how can we check whether the graph structure is useful for inferring node labels? We propose NetEffect_Test, a statistical approach to identify the presence of GNE in a graph. Applying it to real-world graphs, we show that many popular heterophily graphs exhibit little GNE.

3.3.1 NetEffect_Test

We first provide two main definitions regarding GNE:

Definition 3.2

If the nodes with class c_i in a graph tend to connect randomly to the nodes with all classes $1, \ldots, c$ (with no specific preference), class c_i has no GNE.

Definition 3.3

If all classes in a graph have no GNE, this graph has no GNE.

We distinguish heterophily graphs from those with no GNE by the definition. In heterophily graphs, the nodes of a specific class are likely to be connected to the nodes of other classes, such as in bipartite graphs that connect different classes of nodes. In this case, knowing the label of a node gives meaningful information about the labels of its neighbors. On the other hand, if a graph has little GNE, knowing the label of a node gives no useful information about its neighbors. In other words, the structural information of a graph is not useful to infer the unknown labels of nodes.

Next, we describe how we propose to determine the existence or absence of GNE. In the inner loop, we need to decide whether class c_i (say, "talkative people"), has statistically more, or fewer edges to class c_j (say, "silent people"). We propose to use Pearson's χ^2 test for that. Specifically, given a class pair (c_i, c_j), the input to the test is a 2×2 contingency table containing the counts of edges that connect pairs of nodes whose labels are in $\{c_i, c_j\}$. The null hypothesis of the test is:

Null Hypothesis 3.1

Edges are equally likely to exist between nodes of the same class and those of different classes.

If the p-value from the test is no less than 0.05, we accept the null hypothesis, which represents that the chosen class pair (c_i, c_j) exhibits no statistically significant GNE in the graph. Then we call them *mutually indistinguishable*:

Definition 3.4: Mutually indistinguishable

Two classes c_i and c_j are *mutually indistinguishable* if we cannot reject the null hypothesis above.

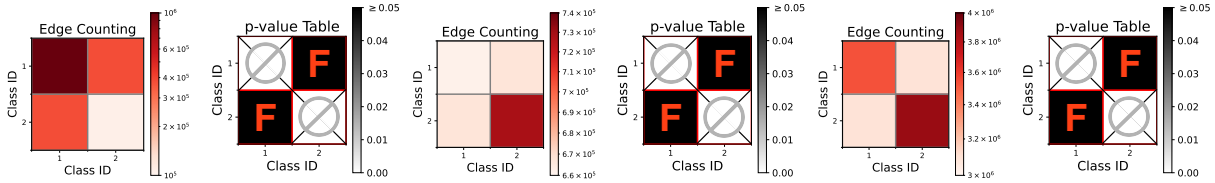
Algorithm 3.1: NetEffect_Test

Data: Edges $\mathcal{E}$ and priors $\mathcal{P}$
Result: p-value table $\mathbf{F}$

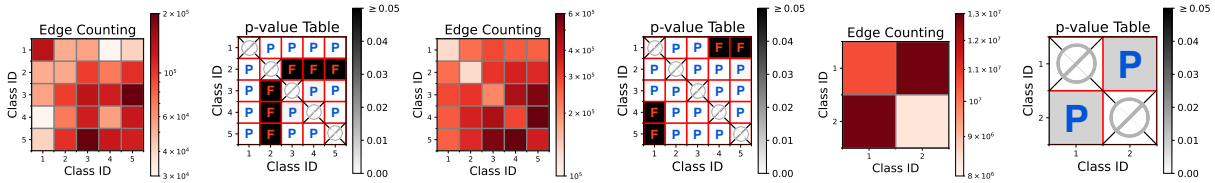
Extract $\mathcal{E}^{'}$ such that $(i,j) \in \mathcal{E}, i,j \in \mathcal{P}\ \forall (i,j) \in \mathcal{E}^{'}$;
$\mathbf{T} \leftarrow \mathbf{O}_{c\times c}$; // Test statistic table
for $b_1 = 1, ..., B$ **do**
  for $c_1 = 1, ..., c-1$ **do**
    for $c_2 = c_1 + 1, ..., c$ **do**
      $\mathbf{V} \leftarrow \mathbf{O}_{2\times 2}$; // Contingency table
      for $(i,j) \in$ *Sampled*$(\mathcal{E}^{'})$ **do**
        if $l(i) = l(j) = c_1$ *or* $l(i) = l(j) = c_2$ **then**
          $\mathbf{V}_{11} \leftarrow \mathbf{V}_{11} + 2$;
        end
        else if $(l(i) = c_1$ *and* $l(j) = c_2)$ *or* $(l(i) = c_2$ *and* $l(j) = c_1)$ **then**
          $\mathbf{V}_{21} \leftarrow \mathbf{V}_{21} + 1$ and $\mathbf{V}_{12} \leftarrow \mathbf{V}_{12} + 1$;
        end
      end
      $T = \chi^2$-Test-Statistic$(\mathbf{V}/2)$;
      $\mathbf{T}_{c_1c_2} \leftarrow \mathbf{T}_{c_1c_2} + T/B$ and $\mathbf{T}_{c_2c_1} \leftarrow \mathbf{T}_{c_2c_1} + T/B$;
    end
  end
end
Compute p-value table $\mathbf{F}_{c\times c}$ with average statistics in $\mathbf{T}$;
Return $\mathbf{F}$;

Novel Implementation Details The detailed procedure of NetEffect_Test is in Algorithm 3.1. A practical challenge on the test is that if the numbers in the table are too large, p-value becomes very small and meaningless [LJS13]. Uniform edge sampling can be a natural solution, but sampling for only a single round can be unstable and output very different results. To address this, we combine p-values from different random sampling by Universal Inference [WRB20]. We firstly sample edges to add to the contingency table until the frequency is above a specified threshold, and compute the χ^2 test statistic for each class pair. Next, following Universal Inference, we repeat the procedure for random samples of edges for B rounds and average the statistics. At last, we use the average statistics to compute the p-value table $\mathbf{F}_{c\times c}$ of χ^2 tests. Our NetEffect_Test is robust to noisy edges thanks to the sampling process, and works well given either a few or many node labels. Given a few observations, the χ^2 test works well when the frequency in the contingency table is at least 5; given many observations, our sampling trick ensures the correctness and consistency of the computed p-value by limiting the frequency to no more than 500.

If a class accepts the null hypotheses with all other classes, this class has little GNE, and satisfies Definition 3.2. Moreover, if all classes exhibit little GNE, the whole graph satisfies Definition 3.3. In that case, no label propagation methods will help with node classification.



(a) "Genius": No GNE (b) "Penn94": No GNE (c) "Twitch": No GNE



(d) "arXiv-Year": Weak GNE (e) "Patent-Year": Weak GNE (f) "Pokec-Gender": Strong GNE

Datasets	Genius	Penn94	Twitch	Patent-Year	Pokec-Gender	arXiv-Year	Synthetic
# of Nodes	422K	42KM	168K	1.3M	1.6M	169K	1.2M
# of Edges	985K	1.4M	6.8M	4.3M	22.3M	1.2M	34.0M
# of Classes	2	2	2	5	2	5	6
Edge Homophily	0.618	0.470	0.545	0.132	0.425	0.222	0.314
$\hat{h}$	0.080	0.046	0.090	0.000	0.000	0.272	0.245
GNE	No GNE	No GNE	No GNE	Heterophily	Heterophily	x-ophily	x-ophily

(g) Homophily statistics of graphs and their GNE

Figure 3.2: **NetEffect_Test discovers real-world heterophily graphs with little GNE.** For each graph, we report the edge counting on the left (not available in practice), and the p-value table output from NetEffect_Test on the right, where "P" denotes the presence of GNE, and "F" denotes the absence of GNE.

3.3.2 Discoveries on Real-World Graphs

We apply NetEffect_Test to 6 real-world graphs and analyze their GNE. For each dataset, we sample 5% of node labels and compute the p-value table using NetEffect_Test. This is because: a) only few labels are available in most node classification tasks in practice, and thus it is reasonable to make the same assumption in the analysis, and b) NetEffect_Test can analyze GNE even from partial observations. B is set to 1000 to output stable results. Based on Definition 3.3, our surprising discoveries are:

Discovery 3.1: Little GNE

NetEffect_Test identifies the lack of GNE in "Genius" [LB21], "Penn94" [TMP11], and "Twitch" [RS21].

They are widely known as heterophily graphs. In "Genius" (Figure 3.2a), we see that both classes 1 and 2 tend to connect to class 1, making class 2 indistinguishable by the graph structure. NetEffect_Test thus accepts the null hypothesis, and identifies the lack of GNE. We can observe a similar phenomenon in "Penn94" (Figure 3.2b). "Twitch" (Figure 3.2c) used to be considered as a heterophily graph because of its weak homophily effect, but NetEffect_Test finds that each of the classes uniformly connects to both classes, and thus it has little GNE.

Discovery 3.2: Heterophily and x-ophily

NetEffect_Test identifies GNE in "Arxiv-Year", "Patent-Year", and "Pokec-Gender".

While "Patent-Year" and "Pokec-Gender" exhibit heterophily (Figure 3.2e and 3.2f), "Arxiv-Year" exhibits x-ophily, i.e., not straight homophily or heterophily (Figure 3.2d). They are thus used in our experiments.

Discovery 3.3: Weak vs strong GNE

NetEffect_Test identifies weak, and strong GNE: "Arxiv-Year" and "Patent-Year" exhibit weak GNE; and "Pokec-Gender" exhibits strong GNE.

We consider graphs to have weak GNE if there exists at least one class which is not distinguishable from some other classes. Such graphs limit the accuracy of node classification, compared with graphs with strong GNE (i.e., all classes have GNE), regardless of the specific method used for classification.

Discussion of Homophily Statistics In Figure 3.2g, we report two homophily statistics. Edge homophily [ZYZ+20] is the edge ratio that connect two nodes with the same class, and $\hat{h}$ [LHL+21] is an improved metric which is insensitive to the class number and size. We find even using all labels, they are not enough to capture the interrelations of all class pairs in detail, and the graphs with low homophily statistics are not guaranteed to be heterophily. They can only detect the absence of homophily, instead of distinguishing different non-homophily cases, including heterophily, x-ophily, and no GNE. In contrast, our NetEffect_Test identifies whether the graph exhibits GNE or not from only a few labels.

3.4 Proposed GNE Estimation

Given that a graph exhibits GNE, how can we estimate the all-pair relations between classes? A *compatibility matrix* is a natural strategy to describe the relations, which has been widely used in the literature. We propose NetEffect_Est, which turns the compatibility matrix estimation into an optimization problem based on a closed-form formula. NetEffect_Est not only over-

comes the limitation of naive edge counting, but is also robust to noisy observations even with few observed labels.

3.4.1 Why NOT Edge Counting

The graph in Figure 3.3a exhibits heterophily between class pairs (1, 2) and (3, 4), while it exhibits homophily in classes 5 and 6. A compatibility matrix is commonly used in existing studies, but assumed given by domain experts, instead of being estimated. A naive way to estimate it is via counting labeled edges, but it has two limitations: 1) rare labels are neglected, and 2) it is noisy or biased due to few labeled nodes. The result is even more unreliable if the given labels are imbalanced. In Figure 3.3, we upsample the training labels 10 times for class 1 using the graph in Figure 3.1b. Edge counting in Figure 3.3b biases towards the upsampled class and clearly fails to estimate the correct compatibility matrix in Figure 3.3a, while our proposed NetEffect_Est succeeds in Figure 3.3c. This commonly occurs in practice, since we observe only limited labels, and becomes fatal if the observed distribution is different from the true one.

3.4.2 Closed-Form Formula

We begin the derivation by rewriting Equation (3.1) of BP. The main insight is reminiscent of 'leave-one-out' cross validation. That is, we find $\hat{\mathbf{H}}$ that would make the results of the propagation (RHS of Equation (3.2)) to the actual values (LHS of Equation (3.2)):

$$\underbrace{\hat{\mathbf{E}}}_{\text{reality}} \approx \underbrace{\mathbf{A}\hat{\mathbf{E}}\hat{\mathbf{H}}}_{\text{estimate}} \tag{3.2}$$

Formally, we want to minimize the difference between the reality and the estimate:

$$\min_{\hat{\mathbf{H}}} \sum_{i \in \mathcal{P}} \sum_{u=1}^{c} \|\hat{\mathbf{E}}_{iu} - \sum_{k=1}^{c} \sum_{j \in N(i) \cap \mathcal{P}} \hat{\mathbf{E}}_{jk}\hat{\mathbf{H}}_{ku}\|^2, \tag{3.3}$$

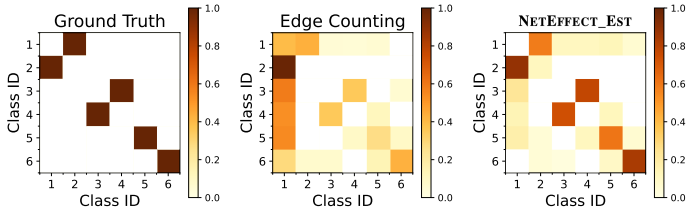
where $N(i)$ denotes the neighbors of node i. In other words, we aim to minimize the difference between initial belief $\hat{\mathbf{E}}$ of each node $i \in \mathcal{P}$ by the ones of its neighbors $N(i) \in \mathcal{P}$, i.e., $N(i) \cap \mathcal{P}$. To estimate the compatibility matrix $\hat{\mathbf{H}}$, we solve the optimization problem in Equation (3.3) with the proposed closed-form formula:

Lemma 3.1: Network Effect Formula

Given adjacency matrix $\mathbf{A}$ and initial beliefs $\hat{\mathbf{E}}$, the closed-form solution of vectorized compatibility matrix $\text{vec}(\hat{\mathbf{H}})$ is:

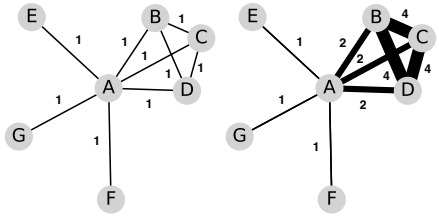
$$\boxed{\text{vec}(\hat{\mathbf{H}}) = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{y}} \tag{3.4}$$

where $\mathbf{X} = \mathbf{I}_{c\times c} \otimes (\mathbf{A}\hat{\mathbf{E}})$, $\mathbf{y} = \text{vec}(\hat{\mathbf{E}})$, and $\mathbf{I}_{c\times c}$ is a $c \times c$ identity matrix.



(a) Ground Truth (b) Edge Counting (c) NetEffect_Est

Figure 3.3: **NetEffect_Est handles imbalanced case well.** Labels of class 1 is upsampled.



(a) Adjacency Matrix (b) Emphasis Matrix

Figure 3.4: **Emphasis matrix at work:** it prefers well-connected neighbors.

Proof. We derive the solution using the closed linear regression formula, and the property of the mixed Kronecker matrix-vector product introduced before in Definition 2.8. By vectorizing Equation (3.2) as introduced in Definition 2.6, we have:

$$\begin{aligned} \text{vec}(\hat{\mathbf{E}}) &= \text{vec}(\mathbf{A}\hat{\mathbf{E}}\hat{\mathbf{H}}) \\ &= \text{vec}((\mathbf{A}\hat{\mathbf{E}})\hat{\mathbf{H}}\mathbf{I}_{c\times c}) \\ &= (\mathbf{I}_{c\times c} \otimes (\mathbf{A}\hat{\mathbf{E}}))\text{vec}(\hat{\mathbf{H}}), \end{aligned} \tag{3.5}$$

where $\otimes$ denotes the Kronecker product (Definition 2.7). Let $\mathbf{X} = \mathbf{I}_{c\times c} \otimes (\mathbf{A}\hat{\mathbf{E}})$ and $\mathbf{y} = \text{vec}(\hat{\mathbf{E}})$, the above equation can be solved by linear regression with the closed-form solution of $\text{vec}(\hat{\mathbf{H}})$, which completes the proof. ■

3.4.3 NetEffect_Est

The algorithm is presented in Algorithm 3.2. In practice, we can use any form of adjacency matrix for the estimation. Our proposed formula allows us to estimate the compatibility matrix by solving this optimization problem, but there still exists a practical challenge that need to be addressed. With few labels, it is difficult to properly separate them into training and validation sets for the regression, and the estimation can easily be interfered by the noisy observations. We thus use ridge regression with leave-one-out cross-validation (RidgeCV). This allows us to fully utilize the observations without having biases caused by random splits of training and validation sets. Moreover, the regularization effect of RidgeCV makes the compatibility matrix more robust to noisy observations. It is noteworthy that its computational cost is negligible.

3.5 Proposed GNE Exploitation

We propose NetEffect_Exp to exploit GNE for accurate and fast node classification with few labels. With few labels, it becomes crucial to better utilizing the graph structure. First, we address this by paying attention to influential neighbors by the proposed "emphasis" matrix; and then describe NetEffect_Exp with theoretical analysis.

Algorithm 3.2: NetEffect_Est

Data: Adjacency matrix $\mathbf{A}$, initial belief $\hat{\mathbf{E}}$, and priors $\mathcal{P}$
Result: Estimated compatibility matrix $\hat{\mathbf{H}}$
1 $\mathbf{X} \leftarrow \mathbf{I}_{c\times c} \otimes (\mathbf{A}\hat{\mathbf{E}})$; // Feature matrix
2 $\mathbf{y} \leftarrow \text{vec}(\hat{\mathbf{E}})$; // Target vector
3 Extract indices $\mathbf{i}$ with nodes in priors $\mathcal{P}$;
4 $\hat{\mathbf{H}} \leftarrow RidgeCV(\mathbf{X}[\mathbf{i}], \mathbf{y}[\mathbf{i}])$;
5 Return $\hat{\mathbf{H}}$;

3.5.1 "Emphasis" Matrix

Rationale and Overview With few priors, we propose to better utilize the graph structure, by paying attention to only the most important part of it. That is to say, not all neighbors are equally influential: In Figure 3.4, best practice shows that well-connected neighbors (i.e., nodes 'B', 'C', and 'D') have more influence on node 'A' than the rest. Thus, we propose "emphasis" matrix $\mathbf{A}^*$ to pay attention to such neighbors. NetEffect_Est can also benefit from it by replacing $\mathbf{A}$ with $\mathbf{A}^*$, where we denote the improved compatibility matrix as $\hat{\mathbf{H}}^*$. Algorithm 3.3 shows the details. In short, it has 3 steps:

1. **Favors influential neighbors** by quickly approximating the node-to-node proximity using (non-backtracking) random walks with restarts (lines 2-5);
2. **Touches-up** the new node-proximity by applying a series of transformations (including the best-practice element-wise logarithm) on the proximity matrix (line 6);
3. **Symmetrizes and weighs** the adjacency matrix with structural-aware embedding (lines 7-8), giving higher weights to neighbors with closer embeddings (line 9).

Proximity Matrix Approximation We propose to utilize random walks to approximate the proximity matrix. The approximated proximity matrix $\mathbf{W}'_{ij}$ records the times we visit node j if we start a random walk from node i. Only the well-connected neighbors will be visited more often. We theoretically show that it converges quickly:

Lemma 3.2: Convergence of Random Walks

With probability $1-\delta$, the error ϵ between the approximated and true distributions for a node walking to its 1-hop neighbor by random walks of length L with M trials is no greater than $\frac{\lceil (L-1)/2 \rceil}{L}\sqrt{\frac{\log(2/\delta)}{2LM}}$.

We further speed up the convergence by using non-backtracking random walks [ABLS07]. Given the start node s and walk length L, its function is defined as follows:

$$\mathcal{W}(s,L) = \left\{ (w_0 = s, ..., w_L) \quad \begin{matrix} w_l \in N(w_{l-1}), \forall l \in [1,L] \\ w_{l-1} \neq w_{l+1}, \forall l \in [1,L-1] \end{matrix} \right. . \tag{3.6}$$

Algorithm 3.3: "Emphasis" Matrix

Data: Adjacency matrix $\mathbf{A}$, number of trials M, number of steps L, and dimension d
Result: Emphasis matrix $\mathbf{A}^*$

$\mathbf{W}' \leftarrow \mathbf{O}_{n\times n}$;
/* Approximate proximity matrix by random walk */
for *node* i *in* G **do**
  for $m = 1, ..., M$ **do**
    for $j \in \mathcal{W}_m(i, L)$ **do**
      $\mathbf{W}'_{ij} \leftarrow \mathbf{W}'_{ij} + 1$;
    end
  end
end
/* Masking, degree normalization and logarithm */
$\mathbf{W}_{n\times n} \leftarrow \log\left(\mathbf{D}^{-1}(\mathbf{W}' \odot \mathbf{A})\right)$;
$\mathbf{U}_{n\times d}, \mathbf{\Sigma}_{d\times d}, \mathbf{V}^T_{d\times n} \leftarrow \text{SVD}(\mathbf{W}, d)$; // Embedding
$\mathbf{U} \leftarrow \sqrt{\mathbf{\Sigma}}\mathbf{U}$; // Scaling
/* Boost weights of close-embedded neighbors */
Weigh $\mathbf{A}^*_{n\times n}$, where $\mathbf{A}^*_{ij} = \mathcal{S}(\mathbf{U}_i, \mathbf{U}_j), \forall\{i, j|\mathbf{A}_{ij} = 1\}$;
Return $\mathbf{A}^*$;

Thanks to it, we improve Lemma 3.2 to have a tighter bound of error ϵ:

Lemma 3.3: Convergence of Non-Backtracking Random Walks

With the same condition as in Lemma 3.2, the error ϵ by non-backtracking random walks is no greater than $\frac{\lceil (L-1)/3 \rceil}{L}\sqrt{\frac{\log(2/\delta)}{2LM}}$.

Proof. For a L-steps random walk sequence S with M trials, its length $|S|$ is LM. A random variable X denotes the probability of node i will walk to its j-th neighbor is:

$$X = \mathbb{P}(\text{node } i \text{ walks to } N(i)_j) = \frac{\sum_{k=1}^{|S|} \mathbf{1}(N(i)_j = S_k)}{|S|}, \tag{3.7}$$

where $\mathbb{P}$ is the probability and $\mathbf{1}$ is the indicator. With regular random walk in the graph, X is upper-bounded by $\frac{\lceil (L-1)/2 \rceil}{L}$. By applying Hoeffding's inequality, we have:

$$\mathbb{P}(|\hat{\mu}_{|S|} - \mu| \geq \epsilon) \leq 2\exp\frac{-2L^3Mt^2}{\lceil (L-1)/2 \rceil^2}, \tag{3.8}$$

where $\hat{\mu}_{|S|}$ denotes the sampled mean of the given random variable, and μ denotes the expectation. Let $\delta = 2\exp\frac{-2L^3Mt^2}{\lceil (L-1)/2 \rceil^2}$, with probability $1-\delta$, we have the error:

$$\epsilon = |\hat{\mu}_{|S|} - \mu| \leq \frac{\lceil (L-1)/2 \rceil}{L}\sqrt{\frac{\log(2/\delta)}{2LM}} \tag{3.9}$$

Algorithm 3.4: NETEFFECT_EXP

Data: "Emphasis" matrix $\mathbf{A}^*$, estimated compatibility matrix $\hat{\mathbf{H}}^*$, and initial belief $\hat{\mathbf{E}}$
Result: Final belief $\mathbf{B}$

$\hat{\mathbf{B}}_{(0)} \leftarrow \mathbf{O}_{n\times c}, t \leftarrow 0$;
/* Propagation */
while $\|\hat{\mathbf{B}}_{(t+1)} - \hat{\mathbf{B}}_{(t)}\|_1 > 1$ **do**
  $\hat{\mathbf{B}}_{(t+1)} \leftarrow \hat{\mathbf{E}} + f\mathbf{A}^*\hat{\mathbf{B}}_{(t)}\hat{\mathbf{H}}^*$;
  $t \leftarrow t + 1$;
end
Return $\mathbf{B} \leftarrow \hat{\mathbf{B}}_{(t)} + 1/c$;

With non-backtracking random walk [ABLS07], the upper bound of X can be decreased to $\frac{\lceil (L-1)/3 \rceil}{L}$. Let $\delta = 2\exp\frac{-2L^3Mt^2}{\lceil (L-1)/3 \rceil^2}$, with probability $1-\delta$, we now have the error:

$$\epsilon = |\hat{\mu}_{|S|} - \mu| \leq \frac{\lceil (L-1)/3 \rceil}{L}\sqrt{\frac{\log(2/\delta)}{2LM}} \tag{3.10}$$

■

Structural-Aware Node Representation Based on $\mathbf{W}$, we apply a series of transformations to generate better and unbiased representations of nodes in a fast way. An element-wise multiplication by $\mathbf{A}$ (i.e., the Hadamard product in Definition 2.5) is done to keep the approximation of 1-hop neighbor for each node, which is sparse but supplies sufficient information. We use the inverse of the degree matrix $\mathbf{D}^{-1}$ to reduce the influence of nodes with large degrees. This prevents them from dominating the pairwise distance by containing more elements in their rows. The element-wise logarithm rescales the distribution in $\mathbf{W}$, in order to enlarge the difference between smaller structures. We use Singular Value Decomposition (SVD) for efficient rank-d decomposition of sparse $\mathbf{W}$, and multiply the left-singular vectors $\mathbf{U}$ by the squared eigenvalues $\sqrt{\mathbf{\Sigma}}$ to correct the scale.

"Emphasis" Matrix Construction Directly measuring the node similarity in the graph is not trivial, or may be time consuming (e.g., by counting motifs). Therefore, we propose to compute the node similarity via the structural-aware node representations, which capture the higher-order information, and construct the "emphasis" matrix $\mathbf{A}^*$ by weighing $\mathbf{A}$ with the node similarity. The intuition is that the nodes that are closer in the embedding space are better connected with higher-order structures. The similarity function is $\mathcal{S}(\mathbf{U}_i, \mathbf{U}_j) = e^{-\mathcal{D}(\mathbf{U}_{ik}, \mathbf{U}_{jk})}$, where e is the Euler's number. It is a universal law [She87], which turns the distance into similarity, and bounds it from 0 to 1. While $\mathcal{D}$ can be any distance metric, we use Euclidean as it works well empirically.

3.5.2 NetEffect_Exp

The algorithm of NetEffect_Exp is given in Algorithm 3.4. NetEffect_Exp takes as input the "emphasis" matrix $\mathbf{A}^*$, the compatibility matrix $\hat{\mathbf{H}}^*$ estimated by $\mathbf{A}^*$, and the initial beliefs $\hat{\mathbf{E}}$. It computes the beliefs $\hat{\mathbf{B}}$ iteratively by aggregating the beliefs of neighbors through $\mathbf{A}^*$ until they converge. This reusage of $\mathbf{A}^*$ aims to draw attention to the neighbors that are more structurally important. By exploiting GNE with $\hat{\mathbf{H}}^*$, NetEffect_Exp propagates properly in heterophily graphs.

Convergence Guarantee To ensure the convergence of NetEffect_Exp, we introduce a scaling factor f during the iterations. A smaller f leads to a faster convergence but distorts the results, thus we set f to $0.9/\rho(\mathbf{A}^*)$. Its exact convergence is as follows:

Lemma 3.4: Exact Convergence

The criterion for the exact convergence of NetEffect_Exp is $0 < f < 1/\rho(\mathbf{A}^*)$, where $\rho(\cdot)$ denotes the spectral radius of the matrix.

Proof. NetEffect_Exp exactly converges if and only if $\rho(\mathbf{A}^*)\rho(\hat{\mathbf{H}}^*) < 1$. $\mathbf{H}^*$ is normalized by row, where $\rho(\mathbf{H}^*) = 1$ is a constant and it is less than 1 after centering. The scaling factor f of propagation must be in the range of $(0, 1/\rho(\mathbf{A}^*))$ to meet the criterion of exact convergence. ■

Complexity Analysis NetEffect_Exp utilizes sparse matrix representations of graphs and scales linearly. Its complexity is as follows:

Lemma 3.5: Time and Space Complexity

The time complexity of NetEffect_Exp is approximately $O(m)$ and the space complexity is $O(\max(m, n \cdot L \cdot M) + n \cdot c^2)$.

Proof. For NetEffect_Est, since there are c sets of parameters are independent, we can separate the problem into c tasks, where each contains c features and $|\mathcal{P}|$ samples. The complexity can then be reduced to $O(|\mathcal{P}| \cdot c^3)$, and the efficient leave-one-out cross-validation only needs to be done once. For NetEffect_Exp, for each random walk, each node visits at most $L \cdot M$ unique nodes, so the maximum number of non-zero elements in $\mathbf{W}$ is either $n \cdot L \cdot M$ if we have not walked through all the edges, or m otherwise. SVD on $\mathbf{W}$ takes $O(d \cdot \max(m, n \cdot L \cdot M))$. It takes at most $O(m + n)$ for sparse matrix multiplication to run t iterations. Thus, the time complexity is $O(d \max(m, n \cdot L \cdot M) + |\mathcal{P}| \cdot c^3 + m)$. In practice, c, $|\mathcal{P}|$ and t are usually small constants which are negligible, and m is usually much larger. Keeping only the dominating terms, the time complexity is approximately $O(m)$. $\mathbf{W}$ contains at most $\max(m, n \cdot L \cdot M)$ non-zero elements. The Kronecker product at most contains $n \cdot c^2$ non-zero elements. The space complexity is $O(\max(m, n \cdot L \cdot M) + n \cdot c^2)$. ■

Table 3.4: **NetEffect wins on x-ophily and heterophily datasets.** Green (■, ■, ■) marks the top three (better is darker).

Dataset	Synthetic			Pokec-Gender			arXiv-Year			Patent-Year		
# of Nodes	1.2M			1.6M			169K			1.3M		
# of Edges	34.0M			22.3M			1.2M			4.3M		
# of Classes	6			2			5			5		
Label Fraction	4%			0.4%			4%			4%		
GNE Strength	Strong x-ophily			Strong Heterophily			Weak x-ophily			Weak Heterophily		
Method	Acc.	Time	Rel. Time	Acc.	Time	Rel. Time	Acc.	Time	Rel. Time	Acc.	Time	Rel. Time
GCN	16.7±0.0	3456	4.1×	51.8±0.1	2906	3.4×	35.3±0.1	132	2.5×	26.0±0.0	894	2.3×
APPNP	18.6±1.1	7705	9.2×	50.9±0.3	6770	7.8×	33.5±0.2	423	8.1×	27.5±0.2	2050	5.2×
MixHop	16.7±0.0	58391	70.0×	53.4±1.2	53871	62.1×	39.6±0.1	2983	57.4×	26.8±0.1	18787	47.6×
GPRGNN	18.9±1.2	7637	9.1×	50.7±0.2	6699	7.7×	30.1±1.4	400	7.7×	25.3±0.1	2034	5.1×
HOLS	46.1±0.1	1672	2.0×	54.4±0.1	8552	9.9×	34.1±0.3	566	10.9×	23.6±0.0	510	1.3×
NetEffect-Hom	45.6±0.1	835	1.0×	56.9±0.2	869	1.0×	37.0±0.3	52	1.0×	24.3±0.0	429	1.1×
NetEffect	80.4±0.0	841	1.0×	67.3±0.1	867	1.0×	38.9±0.1	52	1.0×	28.7±0.1	395	1.0×

3.6 Experiments

In this section, we aims to answer the following questions:
RQ1. **Accuracy**: How well does NetEffect work by estimating and exploiting GNE?
RQ2. **Scalability**: How does the running time of NetEffect scale w.r.t. graph size?
RQ3. **Explainability**: How does NetEffect explain the real-world graphs?

Datasets We focus on large graphs and include 8 graphs with at least 20K nodes. For each dataset, we sample only a few node labels for training for five times and report the average. "Synthetic" is the enlarged graph in Figure 3.1b, which exhibits x-ophily GNE.

Baselines We compare NetEffect with five baselines and separate them into four groups: *General GNNs:* GCN [KW17], APPNP [KBG19]. *Heterophily GNNs:* MixHop [APK+19], GPRGNN [CPLM21]. *BP-based methods:* HOLS [EKF20]. *Our proposed methods:* NetEffect-Hom and NetEffect. NetEffect-Hom is NetEffect using identity matrix as compatibility matrix, which assumes homophily and does not handle GNE.

Experimental Settings For GNNs, one-hot node degrees are used as the node features, as implemented by PyG [FL19]. Experiments are run on a server with 3.2 GHz Intel Xeon CPU.

3.6.1 RQ1 – Accuracy

In Table 3.4 and 3.5, we report the accuracy and running time. We highlight the top three from dark to light by ■, ■ and ■ denoting the first, second and third place. In summary, NetEffect wins on x-ophily, heterophily and homophily graphs.

x-ophily and Heterophily In Table 3.4, NetEffect outperforms all the competitors significantly by more than 34.3% and 12.9% accuracy on "Synthetic" and "Pokec-Gender", respectively.

Table 3.5: **NetEffect wins on homophily datasets.** Green (■, ■, ■) marks the top three (better is darker).

Dataset	Facebook			GitHub			arXiv-Category			Pokec-Locality		
# of Nodes	22.5K			37.7K			169K			1.6M		
# of Edges	171K			289K			1.2M			22.3M		
# of Classes	4			2			40			10		
Label Fraction	4%			4%			4%			0.4%		
Method	Acc.	Time	Rel. Time	Acc.	Time	Rel. Time	Acc.	Time	Rel. Time	Acc.	Time	Rel. Time
GCN	67.0±0.8	12	2.0×	81.0±0.6	28	2.2×	25.4±0.3	216	2.3×	17.3±0.4	4002	2.9×
APPNP	50.5±2.2	46	7.7×	74.2±0.0	73	5.6×	19.4±0.6	1176	12.3×	16.8±1.7	11885	8.6×
MixHop	69.2±0.7	296	49.3×	77.8±1.3	526	40.5×	33.0±0.6	3203	33.4×	16.9±0.3	52139	37.9×
GPRGNN	51.9±1.5	47	7.8×	74.1±0.1	75	5.8×	19.7±0.3	1174	12.2×	30.0±2.0	11959	8.7×
HOLS	86.0±0.4	934	155.7×	80.8±0.5	126	9.7×	61.4±0.2	627	6.5×	63.7±0.3	8139	5.9×
NetEffect-Hom	85.2±0.5	6	1.0×	81.3±0.5	13	1.0×	61.7±0.2	96	1.0×	66.0±0.2	1437	1.0×
NetEffect	85.2±0.5	6	1.0×	81.3±0.5	13	1.0×	58.8±0.6	108	1.1×	64.8±0.8	1377	1.0×

Table 3.6: **Ablation Study:** Estimating the compatibility matrix using our proposed "emphasis" matrix allows NetEffect to perform better. Green (■) marks the winner.

Dataset	GNE Strength	NetEffect-Hom	NetEffect-EC	NetEffect-A	NetEffect
Synthetic	Strong	77.7±0.0	68.0±0.1	77.4±0.0	80.5±0.0
Pokec-Gender		56.9±0.1	64.9±0.2	64.8±0.2	67.3±0.1
arXiv-Year (imba.)	Weak	37.0±0.3	36.5±1.0	35.7±0.6	38.4±0.0
Patent-Year (imba.)		24.1±0.0	24.0±0.9	28.7±0.1	28.7±0.0

These graphs exhibit strong GNE, thus NetEffect boosts the accuracy owing to precise estimations of compatibility matrix. Heterophily GNNs give results close to majority voting when the observed labels are not adequate. With homophily assumption, General GNNs and BP-based methods also not perform well. Both "arXiv-Year" and "Patent-Year" have weak GNE (Section 3.3.2). Even so, NetEffect still outperforms the competitors by estimating a reasonable compatibility matrix (Figure 3.6c).

Homophily In Table 3.5, NetEffect-Hom outperforms all the competitors on 3 out of 4 homophily graphs, namely "GitHub", "arXiv-Category" and "Pokec-Locality", and NetEffect performs similarly to NetEffect-Hom. In addition, NetEffect-Hom performs competitively with HOLS on "Facebook", while being 155.7× faster.

Ablation Study We evaluate different compatibility matrices – (i) NetEffect-EC uses edge counting on the labels of adjacent nodes in the priors, and (ii) NetEffect-A uses the adjacency matrix instead of "emphasis" matrix as the input of NetEffect_Est. To evaluate the cases when imbalanced labels are given, we upsample 5% labels to the class with the fewest labels in the datasets with weak GNE during the estimation. In Table 3.6, we find that NetEffect outperforms all its variants in all datasets. In the graphs with strong GNE, NetEffect shows its robustness to the structural noises and gives better results. In the imbalanced graphs, while

Table 3.7: **NetEffect is thrifty.** AWS dollar cost ($) is reported, by t3.small and p3.2xlarge. Green (■) marks the winner.

Dataset	NetEffect	GCN
Pokec-Gender	$0.33 (1.0×)	$12.61 (45.0×)
Pokec-Locality	$0.53 (1.0×)	$13.66 (29.1×)

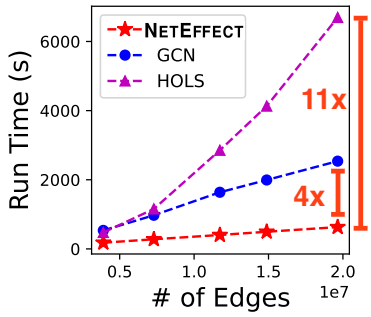


Figure 3.5: **NetEffect is scalable.** It is fast and scales linearly with the edge number.

NetEffect-EC brings its vulnerability to light, NetEffect stays with high accuracy. This study highlights the importance of a compatibility matrix estimation, as well as forming it into an optimization problem as shown in Lemma 3.1.

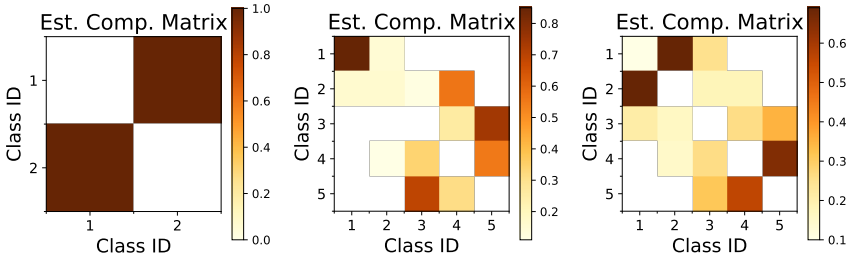
3.6.2 RQ2 – Scalability

NetEffect is scalable. We vary the edge number in "Pokec-Gender" and plot against the running time, including training and inference. In Figure 3.5, NetEffect scales linearly as expected, following Lemma 3.5.

NetEffect is also thrifty. Table 3.7 shows the estimated AWS dollar cost in "Pokec-Gender", assuming that we use an AWS CPU machine for NetEffect, and an AWS GPU machine for GCN. For CPU machine, we select t3.small with 3.3GHz CPU, which is faster than ours, and costs $0.023 per hour. For GPU machine, we select p3.2xlarge with a V100 GPU, which costs $3.06 per hour, which is about 0.89 slower than the RTX A6000 GPU when running PyTorch. The running time of GCN on "Pokec-Gender" and "Pokec-Locality" are 673 and 730 seconds, respectively. Using the information provided, the results in Table 3.7 can be computed.

3.6.3 RQ3 – Explainability

Figure 3.6 shows the compatibility matrices that NetEffect recovered, and the results agree well with intuition. For "Synthetic", NetEffect matches the answer used for graph generation (Figure 3.3c). For "Pokec-Gender", NetEffect report heterophily (Figure 3.6a), where people tend to interact more with the opposite gender [GMB+19]. For "arXiv-Year" and "Patent-Year", NetEffect find that papers and patents often cite nearby-year works (Figure 3.6b and 3.6c).



(a) "Pokec-Gender": Heterophily (b) "arXiv-Year": x-ophily (c) "Patent-Year": Heterophily

Figure 3.6: **NetEffect is explainable.** Our estimated compatibility matrices are much more robust to noises compared to edge counting (in Figure 3.2).

3.7 Conclusion

We analyze the *generalized network-effects* (GNE) in node classification in the presence of only few labels. Our proposed NetEffect has the following desirable properties:

1. **Principled**: NetEffect_Test to statistically identify the presence of GNE,
2. **General** and **Explainable**: NetEffect_Est to estimate GNE with derived closed-form solution, if there is any, and
3. **Accurate** and **Scalable**: NetEffect_Exp to efficiently exploit GNE for better performance on node classification.

Applied on a real-world graph with *22.3M* edges, NetEffect only requires *14 minutes*, and outperforms baselines on both accuracy and speed ($\geq 7\times$).

Chapter 4

SLIMG: Accurate, Robust, and Interpretable Graph Mining

Chapter based on work that appeared at KDD 2023 [YLSF23] [PDF].

How can we solve semi-supervised node classification in various graphs possibly with noisy features and structures? Graph neural networks (GNNs) have succeeded in many graph mining tasks, but their generalizability to various graph scenarios is limited due to the difficulty of training, hyperparameter tuning, and the selection of a model itself. Einstein said that we should "make everything as simple as possible, but not simpler." We rephrase it into the *careful simplicity* principle: a carefully designed simple model can surpass sophisticated ones in real-world graphs.

In this chapter, based on the principle, we propose SLIMG for semi-supervised node classification, which exhibits four desirable properties: It is (a) *Accurate*, winning or tying on *10 out of 13* real-world datasets; (b) *Robust*, being the only one that handles all scenarios of graph data (homophily, heterophily, random structure, noisy features, etc.); (c) *Fast and Scalable*, showing up to *18*× faster training in million-scale graphs; and (d) *Interpretable*, thanks to the linearity and sparsity.

We explain the success of SLIMG through a systematic study of the designs of existing GNNs, sanity checks, and comprehensive ablation studies.

Graph	Feature	SGC	S^2GC	SAGE	GCNII	GPR	SlimG
Noisy	Semantic		✓	✓			✓
Homophily	Noisy						✓
Heterophily	Noisy						✓
Homophily	Structural	✓	✓	✓	✓	✓	✓
Heterophily	Structural	✓		✓		✓	✓
Homophily	Semantic	✓	✓	✓	✓	✓	✓
Heterophily	Semantic	✓		✓	✓	✓	✓

Figure 4.1: **SlimG wins on all sanity checks.** Each row is a specific scenario of graph data that we propose for comprehensive evaluation in Section 4.5. The table is generated from the actual accuracy in Table 4.4: ✓ means the accuracy $\geq 80\%$.

4.1 Introduction

How can we solve semi-supervised node classification in various types of graphs possibly with noisy features and structures? Graph neural networks (GNNs) [KW17, HYL17, GSR+17, VCC+18] have succeeded in various graph mining tasks such as node classification, clustering, or link prediction. However, due to the difficulty of training, hyperparameter tuning, and even the selection of a model itself, many GNNs fail to show their best performance when applied to a large testbed that contains real-world graphs with various characteristics. Especially when a graph contains noisy observations in its features and/or its graphical structure, which is common in real-world data, existing models easily overfit their parameters to such noises.

In response to the question, we propose SlimG, our novel node classification model on graphs based on the *careful simplicity* principle: a simple carefully-designed model can be more accurate than complex ones thanks to better generalizability, robustness, and easier training. The four design decisions of SlimG (D1-4 in Section 4.4) are carefully made to follow this principle by observing and addressing the pain points of existing GNNs; we generate and combine various types of graph-based features (D1), design structure-only features (D2), remove redundancy in feature transformation (D3), and make the propagator function contain no hyperparameters (D4).

The resulting model, SlimG, is our main contribution (C1) which exhibits the following desirable properties:

- **C1.1 – Accurate** on both real-world and synthetic datasets, almost always winning or tying in the first place (see Figure 4.2, Table 4.4, and Table 4.5).
- **C1.2 – Robust**, being able to handle numerous real settings such as homophily, heterophily, no network effects, useless features (see Figure 4.1 and Table 4.4).
- **C1.3 – Fast and scalable**, using carefully chosen features, it takes only 32 seconds on million-scale real-world graphs (ogbn-Products) on a stock server (see Figure 4.2).
- **C1.4 – Interpretable**, learning the largest weights on informative features, ignoring noisy ones, based on the linear decision function (see Figure 4.5).

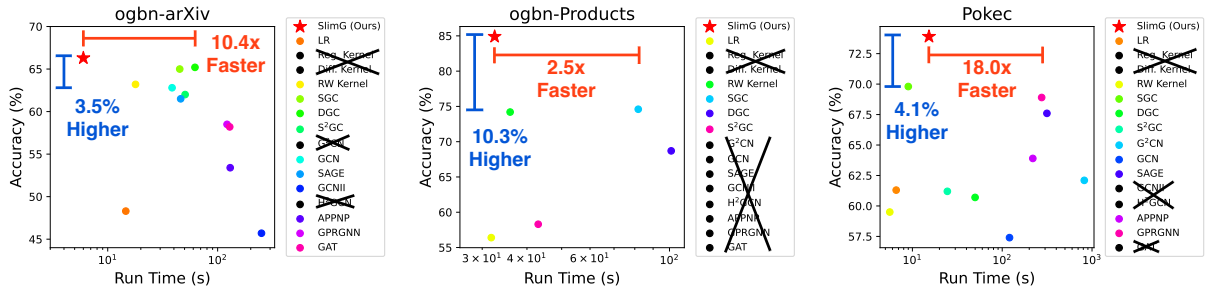


Figure 4.2: **SlimG wins both on accuracy and training time** on (left) ogbn-arXiv, (middle) ogbn-Products, and (right) Pokec, which are large real-world graphs (1.2M, 61.9M, and 30.6M edges, resp.). Several baselines run out of memory (crossed out).

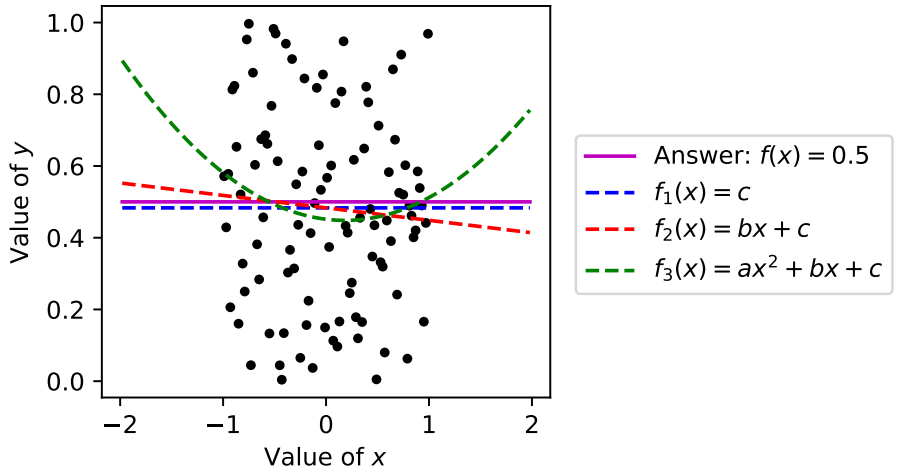


Figure 4.3: **Why 'less is more':** The simple model $f_0(x) = c$ (in blue) matches the reality (in purple), while richer models with more polynomial powers end up capturing noise in the given data: the tiny downward trend by $f_1(x)$ (in red) and the spurious curvature by $f_2(x)$ (in green).

Not only we propose a carefully designed, effective method (in Section 4.4), but we also explain the reasons for its success. This is thanks to our three additional contributions (C2-4):

- **C2 – Explanation (Section 4.3):** We propose GnnExp, a framework for the systematic linearization of a GNN. As shown in Table 4.3, GnnExp highlights the similarities, differences, and weaknesses of successful GNNs.
- **C3 – Sanity checks (Section 4.5):** We propose seven possible scenarios of graphs (homophily, heterophily, no network effects, etc.), which reveal the strong and weak points of each GNN; see Figure 4.1 with more details in Table 4.4.
- **C4 – Experiments (Section 4.6):** We conduct extensive experiments to better understand the success of SlimG even with its simplicity. Our results in Tables 4.6 to 4.10 show that SlimG effectively selects the most informative component in each dataset, fully exploiting its robustness and generality.

Less is more Our justification for the counter-intuitive success of simplicity is illustrated in Figure 4.3: A set of points are uniformly distributed in $x \in (-1, 1)$ and $y \in (0, 1)$, and the fitting polynomials $f_i(x)$ with degree $i = 0, 1, 2$ are given. Notice that the simplest model f_0 (blue line) matches the true generator $f(x) = 0.5$. Richer models use the 1st and the 2nd degree powers (*many cooks spoil the broth*) and end up modeling tiny artifacts, like the small downward slope of f_1 (red line), and the curvature of f_2 (green line). This 'many cooks' issue is more subtle and counter-intuitive than overfitting, as f_2 and f_3 have only 2 to 3 unknown parameters to fit to the hundred data points: even a *small* statistical can fail if it is not matched with the underlying data-generating mechanism.
Reproducibility: Our code, along with our datasets for *sanity checks*, is available at `https://github.com/mengchillee/SlimG`.

4.2 Background and Related Work

4.2.1 Background

We define semi-supervised node classification as follows:

- **Given** an undirected and attributed graph $G = (\mathbf{A}, \mathbf{X})$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is an adjacency matrix, $\mathbf{X} \in \mathbb{R}^{n \times d}$ is a node feature matrix, n is the number of nodes, and d is the number of features.
- **Given** the labels $\mathbf{y} \in \{1, \cdots, c\}^m$ of m nodes in G, where $m \ll n$, and c is the number of classes.
- **Predict** the unknown classes of $n - m$ testing nodes.

An overview of symbols and acronyms is provided in Table 4.1 and Table 4.2, respectively. We use the following symbols to represent adjacency matrices with various normalizations and/or self-loops. $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops. $\tilde{\mathbf{D}} = \mathrm{diag}(\tilde{\mathbf{A}}\mathbf{1}_{n\times 1})$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$, where $\mathbf{1}_{n\times 1}$ is the matrix of size $n \times 1$ filled with ones. $\tilde{\mathbf{A}}_{\text{sym}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ is the symmetrically normalized $\tilde{\mathbf{A}}$. Similarly, $\mathbf{A}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is also the symmetrically normalized $\mathbf{A}$ but without self-loops. We also use a different type of normalization $\mathbf{A}_{\text{row}} = \mathbf{D}^{-1}\mathbf{A}$ (and accordingly $\tilde{\mathbf{A}}_{\text{row}}$), which we call row normalization, based on the position of the matrix $\mathbf{D}$.

As background, we define logistic regression (LR) as a function to find the weight matrix $\mathbf{W}$ that best maps given features to labels with a linear function.

Table 4.1: Table of Symbols and Definitions.

Symbol	Definition
G	Undirected and attributed graph
n	Number of nodes
d	Number of features
c	Number of classes
m	Number of known node labels
$\mathbf{y}$	Set of node labels
$\mathbf{A} \in \mathbb{R}^{n \times n}$	Adjacency matrix
$\mathbf{X} \in \mathbb{R}^{n \times d}$	Node feature matrix
$\mathbf{I} \in \mathbb{R}^{n \times n}$	Identity matrix
$\mathbf{D} \in \mathbb{R}^{n \times n}$	Degree matrix with node degrees along diagonal
$\mathbf{W} \in \mathbb{R}^{d \times c}$	Weight matrix for classification
$\mathcal{P}$	Feature propagator function

Table 4.2: Table of Acronyms.

Acronym	Definition
LR	Logistic Regression
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
GNN	Graph Neural Network
GCN	Graph Convolutional Networks
SGC	Simple Graph Convolution

Definition 4.1: Logistic Regression

Given a feature $\mathbf{X} \in \mathbb{R}^{n \times d}$ and a label $\mathbf{y} \in \mathbb{R}^{m}$, where m is the number of observations such that $m \leq n$, let $\mathbf{Y} \in \mathbb{R}^{m \times c}$ be the one-hot representation of $\mathbf{y}$, and y_{ij} be the (i, j)-th element in $\mathbf{Y}$. Then, logistic regression (LR) is a function that finds an optimal weight matrix $\mathbf{W} \in \mathbb{R}^{d \times c}$ from $\mathbf{X}$ and $\mathbf{y}$ as follows:

$$\mathrm{LR}(\mathbf{X}, \mathbf{y}) = \arg\max_{\mathbf{W}} \sum_{i=1}^{m} \sum_{j=1}^{c} y_{ij} \log \mathrm{softmax}_j(\mathbf{W}^\top \mathbf{x}_i), \tag{4.1}$$

where $\mathrm{softmax}_j(\cdot)$ represents selecting the j-th element of the result of the softmax function. We omit the bias term for brevity.

4.2.2 Graph Neural Networks

We provided a brief review of GNNs in Section 3.2.4 and offer a more comprehensive review here. There exist many recent GNN variants; recent surveys [ZCH+20a, WPC+21] group them into spectral models [DBV16, KW17], sampling based models [HYL17, YHC+18, ZYZ+20], attention based models [VCC+18, KO21, BAY22], and deep models with residual connections [LMTG19, CWH+20]. Decoupled models [KBG19, KWG19, CPLM21] separate the two major functionalities of GNNs: the node-wise feature transformation and the propagation. GNNs are often fused with graphical inference [YJK19, HHS+21] to further improve the predicted results. These GNNs have shown great performance in many graph mining tasks, but suffer from limited robustness when applied to graphs with various characteristics possibly having noisy observations, especially in semi-supervised learning.

4.2.3 Linear Graph Neural Networks

[WJZ+19] proposed SGC by removing the nonlinear activation functions of GCN [KW17], reducing the propagator function to a simple matrix multiplication. [WWYL21] and [ZK21] improved SGC by manually adjusting the strength of self-loops with hyperparameters, increasing the number of propagation steps. [LGW+22] proposed G^{2}CN, which improves the accuracy of DGC [WWYL21] on heterophily graphs by combining multiple propagation settings (i.e. bandwidths). The main limitation of these models is the high complexity of propagator functions with many hyperparameters, which impairs both the robustness and interpretability of decisions even with linearity.

4.2.4 Graph Kernel Methods

Traditional works on graph kernel methods [SK03, IMN+18] are closely related to linear GNNs, which can be understood as applying a linear graph kernel to transform the raw features. A notable limitation of such kernel methods is that they are not capable of addressing various scenarios of real-world graphs, such as heterophily graphs, as their motivation is to aggregate all information in the local neighborhood of each node, rather than ignoring noisy and useless ones. We implement three popular kernel methods as additional baselines and show that our SlimG outperforms them in both synthetic and real graphs.

4.3 Proposed Framework: GnnExp

Why do GNNs work well when they do? In what cases will a GNN fail? We answer these questions with GnnExp, our proposed framework for revealing the essence of each GNN. The idea is to derive the essential *feature propagator* function on which each variant is based, ignoring nonlinearity, so that all models are comparable on the same ground. The observations from GnnExp motivate us to propose our method SlimG, which we describe in Section 4.4.

Definition 4.2: Linearization

Given a graph $G = (\mathbf{A}, \mathbf{X})$, let $f(\cdot; \theta)$ be a node classifier function to predict the labels of all nodes in G as $\hat{\mathbf{y}} = f(\mathbf{A}, \mathbf{X}; \theta)$, where θ is the set of parameters. Then, f is *linearized* if $\theta = \{\mathbf{W}\}$ and the optimal weight matrix $\mathbf{W}^* \in \mathbb{R}^{h \times c}$ is given as

$$\mathbf{W}^* = \mathrm{LR}(\mathcal{P}(\mathbf{A}, \mathbf{X}), \mathbf{y}), \tag{4.2}$$

where $\mathcal{P}$ is a feature propagator function that is linear with $\mathbf{X}$ and contains no learnable parameters, and $\mathcal{P}(\mathbf{A}, \mathbf{X}) \in \mathbb{R}^{n \times h}$. We ignore the bias term without loss of generality.

Definition 4.3: GnnExp

Given a GNN f, GnnExp is to represent f as a linearized GNN by replacing all (nonlinear) activation functions in f with the identity function and deriving a variant f' that is at least as expressive as f but contains no parameters in $\mathcal{P}$.

GnnExp represents the characteristic of a GNN as a linear feature propagator function $\mathcal{P}$, which transforms raw features $\mathbf{X}$ by utilizing the graph structure $\mathbf{A}$. Lemma 4.1 shows that GnnExp generalizes existing linear GNNs. Logistic regression is also represented by GnnExp with the identity propagator $\mathcal{P}(\mathbf{A}, \mathbf{X}) = \mathbf{X}$.

Lemma 4.1

GnnExp includes existing linear graph neural networks as its special cases: SGC, DGC, S^2GC, and G^2CN.

Proof. We prove the lemma for each of SGC, DGC, S^2GC, and G^2CN. The propagator function of SGC [WJZ+19] directly fits the definition of linearization. DGC [WWYL21] has variants DGC-Euler and DGC-DK. We focus on DGC-Euler, which is mainly used in their experiments. Then, DGC also fits the definition of linearization. S^2GC [ZK21] computes the summation of features propagated with different numbers of steps. The original formulation divides the added features by K, which is safely ignored as we multiply the weight matrix $\mathbf{W}$ to the transformed feature for classification.

G^2CN [LGW+22] does not provide an explicit formulation of the propagator function. The parameterized version $\mathcal{P}'$ of the propagator function is $\mathcal{P}'(\mathbf{A}, \mathbf{X}; \{\theta_i\}_{i=1}^N) = \sum_{i=1}^N \theta_i \mathbf{H}_{i,K}$, where θ_i is a parameter. The k-th feature representation $\mathbf{H}_{i,k}$ is recursively defined as $\mathbf{H}_{i,k} = [\mathbf{I} - \frac{T_i}{K}((b_i - 1)\mathbf{I} + \mathbf{A}_{\text{sym}})^2]\mathbf{H}_{i,k-1}$, where $\mathbf{L} = \mathbf{I} - \mathbf{A}_{\text{sym}}$ is the normalized Laplacian matrix, N, T_i, and b_i are hyperparameters, and $\mathbf{H}_{i,0} = \mathbf{X}$. Then, we make it contain no learnable parameters as $\mathcal{P}(\mathbf{A}, \mathbf{X}) = \|_{i=1}^N \mathbf{H}_{i,K}$. We prove the lemma from the four cases. ■

In Table 4.3, we conduct a comprehensive linearization of existing GNNs using GnnExp to understand the fundamental similarities and differences among GNN variants. The models are categorized into linear, decoupled, coupled, and attention models. We ignore bias terms for simplicity, without loss of generality.

4.3.1 Pain Points of Existing GNNs

Based on the comprehensive linearization in Table 4.3, we derive four pain points of existing GNNs which we address in Section 4.4.

Table 4.3: **GnnExp encompasses popular GNNs.** The * and ** superscripts mark fully and partially linearized models, respectively. We derive Pain Points (Section 4.3.1) and Distinguishing Factors (Section 4.3.2) of the variants through GnnExp.

Model	Type	Propagator function $\mathcal{P}(\mathbf{A}, \mathbf{X})$
LR	Linear	$\mathbf{X}$
SGC	Linear	$\tilde{\mathbf{A}}_{\text{sym}}^{K}\mathbf{X}$
DGC	Linear	$[(1-T/K)\mathbf{I} + (T/K)\tilde{\mathbf{A}}_{\text{sym}}]^{K}\mathbf{X}$
S^2GC	Linear	$\sum_{k=1}^{K}(\alpha\mathbf{I} + (1-\alpha)\tilde{\mathbf{A}}_{\text{sym}}^{k})\mathbf{X}$
G^2CN	Linear	$\|_{i=1}^{N}[\mathbf{I} - (T_i/K)((b_i-1)\mathbf{I} + \mathbf{A}_{\text{sym}})^2]^{K}\mathbf{X}$
PPNP*	Decoupled	$(\mathbf{I} - (1-\alpha)\tilde{\mathbf{A}}_{\text{sym}})^{-1}\mathbf{X}$
APPNP*	Decoupled	$[\sum_{k=0}^{K-1}\alpha(1-\alpha)^k\tilde{\mathbf{A}}_{\text{sym}}^{k} + (1-\alpha)^K\tilde{\mathbf{A}}_{\text{sym}}^{K}]\mathbf{X}$
GDC*	Decoupled	$\mathbf{S} = \text{sparse}_\epsilon(\sum_{k=0}^{\infty}(1-\alpha)^k\tilde{\mathbf{A}}_{\text{sym}}^{k})$ for $\tilde{\mathbf{S}}_{\text{sym}}\mathbf{X}$
GPR-GNN*	Decoupled	$\|_{k=0}^{K}\tilde{\mathbf{A}}_{\text{sym}}^{k}\mathbf{X}$
ChebNet*	Coupled	$\|_{k=0}^{K-1}\mathbf{A}_{\text{sym}}^{k}\mathbf{X}$
GCN*	Coupled	$\tilde{\mathbf{A}}_{\text{sym}}^{K}\mathbf{X}$
SAGE*	Coupled	$\|_{k=0}^{K}\mathbf{A}_{\text{row}}^{k}\mathbf{X}$
GCNII*	Coupled	$\|_{k=0}^{K-2}\tilde{\mathbf{A}}_{\text{sym}}^{k}\mathbf{X} \,\|\, ((1-\alpha)\tilde{\mathbf{A}}_{\text{sym}}^{K} + \alpha\tilde{\mathbf{A}}_{\text{sym}}^{K-1})\mathbf{X}$
H_2GCN*	Coupled	$\|_{k=0}^{2K}\mathbf{A}_{\text{sym}}^{k}\mathbf{X}$
GAT**	Attention	$\prod_{k=1}^{K}[\text{diag}(\mathbf{X}\mathbf{w}_{k,1})\tilde{\mathbf{A}} + \tilde{\mathbf{A}}\text{diag}(\mathbf{X}\mathbf{w}_{k,2})]\mathbf{X}$
DA-GNN**	Attention	$\sum_{k=0}^{K}\text{diag}(\tilde{\mathbf{A}}_{\text{sym}}^{k}\mathbf{X}\mathbf{w})\tilde{\mathbf{A}}_{\text{sym}}^{k}\mathbf{X}$

Pain Point 4.1: Lack of Robustness

All models in Table 4.3 fail to handle multiple graph scenarios at the same time, i.e., graphs with homophily, heterophily, no network effects, or useless features.

Most models in Table 4.3 make an implicit assumption on given graphs, such as homophily or heterophily, rather than being able to perform well in multiple scenarios at the same time. For example, all models except ChebNet, SAGE, and H_2GCN have self-loops in the new adjacency matrix, emphasizing the local neighborhood of each node even in graphs with heterophily or no network effects. This is the pain point that we also observe empirically from the sanity checks (in Table 4.4), where none of the existing models succeeds in making reasonable accuracy in all cases of synthetic graphs.

Pain Point 4.2: Vulnerability to Noisy Features

All models in Table 4.3 cannot fully exploit the graph structure if the features are noisy, since they depend on the node feature matrix $\mathbf{X}$.

If a graph does not have a feature matrix $\mathbf{X}$, a common solution is to introduce one-hot features [KW17], i.e., $\mathbf{X} = \mathbf{I}$, although it increases the running time of the model a lot. On the other hand, if $\mathbf{X}$ exists but some of its elements are meaningless with respect to the target classes, models whose propagator functions rely on $\mathbf{X}$ suffer from the noisy elements. In such cases, a desirable property for a model is to adaptively emphasize important features or disregard noisy ones to maximize its generalization performance, which is not satisfied by any of the existing models in Table 4.3.

Pain Point 4.3: Efficiency and Effectiveness

Concatenation-based models in Table 4.3 create spurious correlations between feature elements, requiring more parameters than in other models.

Existing models such as GPR-GNN and GCNII in Table 4.3 perform the concatenation of multiple feature matrices transformed in different ways. For example, GPR-GNN concatenates $\tilde{\mathbf{A}}_{\text{sym}}^{k}\mathbf{X}$ for different values of k from 0 to K, where K is a hyperparameter. Such a concatenation-based propagation limits the efficiency of a model in two ways. First, this increases the number of parameters K times, since the model needs to learn a separate weight matrix for each given feature matrix. Second, this creates spurious correlations in the resulting features, since the feature matrices like $\tilde{\mathbf{A}}_{\text{sym}}\mathbf{X}$ and $\tilde{\mathbf{A}}_{\text{sym}}^{2}\mathbf{X}$ have high correlations with each other.

Pain Point 4.4: Many Hyperparameters

Hyperparameters in $\mathcal{P}$ impair its interpretability and require extensive tuning.

Most models in Table 4.3, even the linear models such as DGC and G^{2}CN, contain many hyperparameters in the propagator function $\mathcal{P}$. Such hyperparameters lead to two limitations. First, the interpretability of the weight matrix $\mathbf{W}$ is impaired, since it is learned on top of the transformed feature $\mathcal{P}(\mathbf{A}, \mathbf{X})$ whose meaning changes arbitrarily by the choice of its hyperparameters. For example, DGC changes the number K of propagation steps between 250 and 900 in real-world datasets, making it hard to have consistent observations from the generated features. Second, $\mathcal{P}(\mathbf{A}, \mathbf{X})$ should be computed for every new choice of hyperparameters, while it can be cached and reused for searching hyperparameters outside $\mathcal{P}$.

4.3.2 Distinguishing Factors

What are the potential choices in designing a general approach that addresses the pain points? We analyze the fundamental similarities and differences among the GNN variants in Table 4.3.

Distinguishing Factor 4.1: Combination of Features

How should we combine node features, the immediate neighbors' features, and the K-step-away neighbors' features?

GNNs propagate information by multiplying the feature $\mathbf{X}$ with (a variant of) the adjacency matrix $\mathbf{A}$ multiple times. There are two main choices in Table 4.3: **(1)** the summation of transformed features (most models), and **(2)** the concatenation of features (GPR-GNN, GraphSAGE, GCNII, and H_2GCN). Simple approaches like SGC are categorized as the summation due to the self-loops in $\tilde{\mathbf{A}}_{\text{sym}}$.

Distinguishing Factor 4.2: Modification of $\mathbf{A}$

How should we normalize or modify the adjacency matrix $\mathbf{A}$?

The three prevailing choices are given as follows: **(1)** symmetric vs. row normalization, **(2)** the strength of self-loops, including making zero self-loops, and **(3)** static vs. dynamic adjustment based on the given features. Most models use the symmetric normalization $\tilde{\mathbf{A}}_{\text{sym}}$ with self-loops, but some variants avoid self-loops and use either row normalization $\mathbf{A}_{\text{row}}$ or symmetric one $\mathbf{A}_{\text{sym}}$. Recent models such as DGC, G^{2}CN, and GCNII determine the weight of self-loops with hyperparameters, since strong self-loops allow distant propagation with a large value of K. Finally, attention-based models learn the elements in $\mathbf{A}$ dynamically based on node features, making propagator functions quadratic with $\mathbf{X}$, not linear.

Distinguishing Factor 4.3: Heterophily

What to do if the direct neighbors differ in their features or labels?

In such cases, the simple aggregation of the features of immediate neighbors may hurt performance, and therefore, several GNNs do suffer under heterophily as shown in Table 4.4. GNNs that can handle heterophily adopt one or more of these ideas: **(1)** using the square of $\mathbf{A}$ as the base structure (G^{2}CN); **(2)** learning different weights for different steps (GPR-GNN, ChebNet, SAGE, and GCNII), and **(3)** making small or no self-loops in the modification of $\mathbf{A}$ (DGC, S^{2}GC, G^{2}CN, and H^{2}GCN). The idea is to avoid or downplay the effect of immediate (and odd-step-away) neighbors. Self-loops hurt under heterophily, as they force to have information of all intermediate neighbors by acting as the implicit summation of transformed features.

4.4 Proposed Method: SlimG

We propose SlimG, a novel method that addresses the limitations of existing models with strict adherence to the *careful simplicity* principle. We first derive four *design decisions* (D1-D4) that

directly address the pain points (PPs) of existing GNN models and propose the following propagator function of SLIMG:

$$\mathcal{P}(\mathbf{A}, \mathbf{X}) = \underbrace{\mathbf{U}}_{\text{Structure}} \parallel \underbrace{g(\mathbf{X})}_{\text{Features}} \parallel \underbrace{g(\mathbf{A}_{\text{row}}^2 \mathbf{X})}_{\text{2-step neighbors}} \parallel \underbrace{g(\tilde{\mathbf{A}}_{\text{sym}}^2 \mathbf{X})}_{\text{Neighbors}} \tag{4.3}$$

where $g(\cdot)$ is the principal component analysis (PCA) for the orthogonalization of each component, followed by an L2 normalization, and $\mathbf{U} \in \mathbb{R}^{n \times r}$ contains r-dimensional structural features independent of node features $\mathbf{X}$, derived by running the low-rank singular value decomposition (SVD) on the adjacency matrix $\mathbf{A}$.

D1: Concatenating winning normalizations (for PP 4.1 – robustness) The main principle of SLIMG to acquire robustness and generalizability, in response to Pain Point 4.1, is to transform the raw features into various forms and then combine them through concatenation. In this way, SLIMG is able to emphasize essential features or ignore useless ones by learning separate weights for different components. The four components of SLIMG in Equation (4.3) show their strength in different cases: structural features $\mathbf{U}$ for graphs with noisy features, self-features $\mathbf{X}$ for a noisy structure, two-step aggregation $\mathbf{A}_{\text{row}}^2$ for heterophily graphs, and smoothed two-hop aggregation $\tilde{\mathbf{A}}_{\text{sym}}^2$ of the local neighborhood for homophily.

Specifically, we use the row-normalized matrix $\mathbf{A}_{\text{row}}$ with no self-loops due to the limitations of the symmetric normalization $\tilde{\mathbf{A}}_{\text{sym}}$: First, the self-loops force one to combine all intermediate neighbors of each node until the K-hop distance, even in heterophily graphs where the direct neighbors should be avoided. Second, neighboring features are rescaled based on the node degrees during an aggregation, even when we want simple aggregation of K-hop neighbors preserving the original scale of features. We thus use $\mathbf{A}_{\text{row}}^2$ along with the popular transformation $\tilde{\mathbf{A}}_{\text{sym}}^2$ in SLIMG.

D2: Structural features (for PP 4.2 – noisy features) In response to Pain Point 4.2, we have to resort to the structure $\mathbf{A}$ ignoring $\mathbf{X}$ when features are missing, noisy, or useless for classification. Thanks to our design decision (D1) for concatenating different components, we can safely add to $\mathcal{P}$ structure-based features which the model can utilize adaptively based on the amount of information $\mathbf{X}$ provides. However, it is not effective to use raw $\mathbf{A}$, which requires the model to learn a separate weight vector for each node, severely limiting its generalizability. We thus adopt low-rank SVD with rank r to extract structural features $\mathbf{U}$. The value of r is selected to keep 90% of the energy of $\mathbf{A}$, where the sum of the largest r squared singular values divided by the squared Frobenius norm of $\mathbf{A}$ is smaller than 0.9. If the chosen value of r is larger than d in large graphs, we set r to d for the size consistency between different components.

D3: Orthogonalization and sparsification (for PP 4.3 – collinearity) To improve the efficiency and effectiveness of SLIMG in response to Pain Point 4.3, we use two reliable methods to further transform the features: dimensionality reduction by PCA and regularization by group LASSO. First, we run PCA on each of the four components to orthogonalize them and to improve the consistency of learned weights. Second, we use group LASSO to learn sparse weights

on the component level, preserving the relative magnitude of each element and suppressing noisy features. To assure the consistency between components (especially with the structural features $\mathbf{U}$), we force all components to have the same dimensionality by selecting r features from each component when adopting PCA.

D4: Multi-level neighborhood aggregation (for PP 4.4 – hyperparameters) Our propagator function $\mathcal{P}$ considers multiple levels of neighborhoods through the concatenation of different components. This allows us to remove all hyperparameters from $\mathcal{P}$ to tune for each dataset, in response to Pain Point 4.4, gaining in both interpretability and efficiency. Specifically, $\mathbf{X}$, $\tilde{\mathbf{A}}^2_{\text{sym}}$, and $\mathbf{A}^2_{\text{row}}$ aggregate the zero-, one-, and two-hop neighborhood of each node, respectively, considering the self-loops included in $\tilde{\mathbf{A}}^2_{\text{sym}}$. Then, $\mathbf{U}$ considers the global topology information of each node, which is in effect the same as considering the distant neighborhood in the graph. As a result, SLIMG performs well in different real-world datasets without tuning any hyperparameters in $\mathcal{P}$, unlike existing GNNs that are often required to increase the value of K up to hundreds [WWYL21].

Time Complexity The time complexity of SLIMG, including all its components SVD, PCA, and the training of LR, is linear with a graph size in most real-world graphs where the number of edges is much larger than the numbers of nodes and features.

Lemma 4.2: Time Complexity

Given a graph, let n and e be the numbers of nodes and edges, respectively, and d be the number of features. Then, the time complexity of the training of SLIMG is $O(de+d^2n+d^3)$.

Proof. The training of SLIMG consists of three parts: SVD, PCA, and LR. The time complexity of SVD is $O(de+d^2n)$ as SLIMG runs the sparse truncated SVD for generating structural features. The complexity of PCA, which is applied to each of the four components in Equation (4.3), is $O(d^2n+d^3)$. The complexity of the gradient-based optimization of LR is $O(dnt)$, where t is the number of epochs. We safely assume $t < n$, since $t < 100$ in all our experiments. We prove the lemma by combining the three complexity terms. ■

4.5 Proposed Sanity Checks

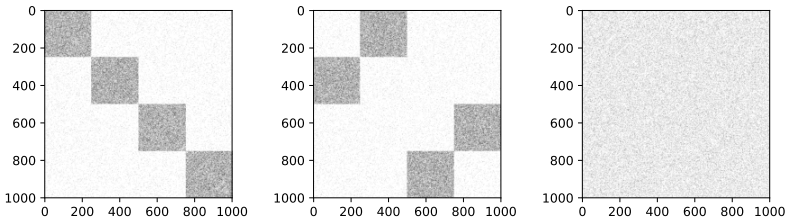
We propose a set of *sanity checks* to directly evaluate the robustness of GNNs to various scenarios of node classification.

4.5.1 Design of Sanity Checks

We categorize possible scenarios of node classification based on the characteristics of node features $\mathbf{X}$, a graph structure $\mathbf{A}$, and node labels $\mathbf{y}$. We denote by A_{ij} and Y_i the random

Structure	Feature		
	Semantic $\mathbf{X}$	Structural $\mathbf{X}$	Random $\mathbf{X}$
Homophily $\mathbf{A}$	Both help	Both help	$\mathbf{A}$ helps
Heterophily $\mathbf{A}$	Both help	Both help	$\mathbf{A}$ helps
Uniform $\mathbf{A}$	$\mathbf{X}$ helps	None helps	None helps

(a) Sanity Check Matrix



(b) Homophily $\mathbf{A}$ (c) Heterophily $\mathbf{A}$ (d) Uniform $\mathbf{A}$

Figure 4.4: **Illustration of our sanity checks.** (a) We consider 3 possibilities for each of $\mathbf{A}$ and $\mathbf{X}$, creating a total of 9 cases. (b - d) We visualize the adjacency matrices from the three cases of $\mathbf{A}$, which exhibit different structural patterns.

variables for edge (i, j) between nodes i and j and label y_i of node i, respectively, and a_{ij} refers to a connection in $\mathbf{A}$. We summarize the nine possible cases in Figure 4.4a.

Structure We consider three cases of the structure $\mathbf{A}$: *uniform*, *homophily*, and *heterophily*, which are defined as follows:

- **Uniform:** $P(Y_i = y \mid A_{ij} = 1, Y_j = y) = P(Y_i = y)$
- **Homophily:** $P(Y_i = y \mid A_{ij} = 1, Y_j = y) > P(Y_i = y)$
- **Heterophily:** $P(Y_i = y \mid A_{ij} = 1, Y_j = y) < P(Y_i = y)$

The uniform case means that the label of a node is independent of the labels of its neighbors. This is the case when the graph structure provides no information for classification. In the homophily case, adjacent nodes are likely to have the same label, which is the most common assumption in graph data. In the heterophily case, adjacent nodes are likely to have different labels, which is not as common as homophily but often observed in real-world graphs.

We assume the one-to-one correspondence between the structural property and the labels: *uniform* with uniformly random $\mathbf{A}$, *homophily* with block-diagonal $\mathbf{A}$, and *heterophily* with non-block-diagonal $\mathbf{A}$. Note that the other combinations, e.g., homophily with non-block-diagonal $\mathbf{A}$, or uniform with block-diagonal $\mathbf{A}$, are not feasible by definition. Figure 4.4 illustrates the three cases of $\mathbf{A}$. The number of node clusters in the graph, which is four in the figure, is the same as the number of node labels in experiments.

Table 4.4: **SLIMG wins on all sanity checks.** Each value denotes the average and the standard deviation of accuracy from five runs. There are three groups of scenarios: (left) only features $\mathbf{X}$ help; (middle) only connectivity $\mathbf{A}$ helps; (right) both help. Green (, ,) marks the top three (higher is darker); red ( ) marks the ones that are too low (2σ below the third place). SLIMG is the only method without red cells and achieves the best average accuracy and average rank (variance in the parentheses).

Model	Only $\mathbf{X}$ helps	Only $\mathbf{A}$ helps		Both $\mathbf{X}$ and $\mathbf{A}$ help				**Avg. Acc.**	**Avg. Rank**
	Semantic $\mathbf{X}$ Uniform $\mathbf{A}$	Random $\mathbf{X}$ Homophily	Random $\mathbf{X}$ Heterophily	Structural $\mathbf{X}$ Homophily	Structural $\mathbf{X}$ Heterophily	Semantic $\mathbf{X}$ Homophily	Semantic $\mathbf{X}$ Heterophily		
LR	83.7±0.6	24.2±0.7	24.2±0.7	71.4±0.9	66.8±2.2	83.4±0.6	83.4±0.6	62.4 (26.9)	10.7 (5.4)
Reg. Kernel	82.7±0.5	27.9±0.4	24.3±1.0	75.7±0.2	65.3±1.6	91.5±0.5	79.5±0.3	63.8 (27.0)	10.4 (4.3)
Diff. Kernel	26.8±1.7	38.0±8.7	37.6±7.5	79.5±0.3	73.5±0.6	70.9±23.	56.1±27.	54.6 (20.7)	10.6 (4.0)
RW Kernel	72.2±0.7	37.0±0.4	24.5±1.3	81.3±1.2	51.0±1.1	94.5±0.9	57.8±0.7	59.8 (24.7)	10.4 (3.6)
SGC	44.6±9.8	64.3±0.7	50.2±14.	87.1±0.6	84.3±0.5	93.9±0.9	91.5±0.5	73.7 (20.4)	5.7 (3.1)
DGC	63.8±1.0	50.5±13.	26.0±0.9	88.6±1.0	45.3±1.3	96.2±0.4	54.0±0.6	60.6 (24.6)	8.3 (5.9)
S^2GC	79.9±0.6	38.5±12.	25.4±0.9	88.4±1.0	67.9±1.5	95.9±0.6	78.0±0.5	67.7 (26.2)	7.4 (3.4)
G^2CN	25.2±0.3	24.2±1.1	25.0±0.1	88.5±1.0	88.6±1.2	24.3±1.1	50.7±31.	46.6 (30.2)	11.6 (6.3)
GCN	36.3±3.5	46.7±8.0	43.7±1.9	83.3±1.3	72.2±1.7	91.2±1.2	80.3±3.9	64.8 (22.1)	8.1 (3.0)
SAGE	80.3±1.1	31.1±0.7	34.6±2.1	83.9±0.8	81.3±0.7	94.4±0.5	94.4±0.9	71.4 (27.0)	5.7 (2.9)
GCNII	73.5±1.2	30.7±0.7	27.1±1.3	84.2±0.8	69.0±1.4	90.6±0.9	80.4±1.2	65.1 (25.7)	8.7 (1.8)
H^2GCN	80.2±1.5	27.0±1.0	27.5±0.8	78.0±0.9	74.6±1.3	91.9±0.7	92.2±0.9	67.3 (28.2)	8.0 (3.9)
APPNP	66.0±2.6	30.3±1.2	25.2±0.7	71.2±4.9	43.8±2.0	83.2±3.8	58.7±4.5	54.1 (21.6)	12.9 (2.0)
GPR-GNN	73.4±0.4	74.6±0.7	65.9±2.1	89.9±0.6	87.6±1.2	95.0±1.1	91.9±1.1	82.6 (11.2)	3.3 (2.1)
GAT	32.7±5.5	42.6±4.8	36.8±5.7	64.0±5.7	55.6±6.8	68.5±7.1	67.0±12.	52.5 (15.0)	11.6 (4.1)
SLIMG (Ours)	81.0±1.1	87.1±1.4	89.2±1.2	88.1±0.5	88.9±0.7	94.4±0.6	93.9±0.5	88.9 (4.5)	2.6 (1.8)

Features We consider three cases of node features $\mathbf{X}$: *random*, *semantic*, and *structural*. The three cases are defined in relation to $\mathbf{A}$ and $\mathbf{y}$. We use the notation $p(\cdot)$ since the features are typically modeled as continuous variables:

- **Random:** $p(\mathbf{x}_i, \mathbf{x}_j \mid y_i, y_j, a_{ij}) = p(\mathbf{x}_i, \mathbf{x}_j)$
- **Structural:** $p(\mathbf{x}_i, \mathbf{x}_j \mid y_i, y_j, a_{ij}) \neq p(\mathbf{x}_i, \mathbf{x}_j \mid y_i, y_j)$
- **Semantic:** $p(\mathbf{x}_i, \mathbf{x}_j \mid y_i, y_j, a_{ij}) \neq p(\mathbf{x}_i, \mathbf{x}_j \mid a_{ij})$

The random case means that each feature element x_{ij} is determined independently of all other variables in the graph, providing no useful information. The semantic case represents a typical graph where $\mathbf{X}$ provides useful information of $\mathbf{y}$. In this case, the feature $\mathbf{x}_i$ of each node i is directly correlated with the label y_i. In the structural case, $\mathbf{X}$ provides information of the graph structure, rather than the labels. Thus, $\mathbf{X}$ is meaningful for classification if $\mathbf{A}$ is not *uniform*, as it gives only indirect information of $\mathbf{y}$.

4.5.2 Observations from Sanity Checks

Table 4.4 shows the results of sanity checks for our SLIMG and all baseline models whose details are given in Section 4.6. We assume 4 target classes of nodes, making the accuracy of random guessing 25%. Among the nine cases in Table 4.4a, we do not report the results on two cases where the graph does not give any information (i.e., "none helps"), since all methods produce similar accuracy.

SLIMG wins on all sanity checks, thanks to its careful design for the robustness to various graph scenarios. Although homophily $\mathbf{A}$ and useful (i.e., not random) $\mathbf{X}$ is a common assump-

tion in many datasets, many nonlinear GNNs show failure (i.e., red cells) in such cases. This implies that the *theoretical expressiveness* of a model is often not aligned with its actual performance even in controlled testbeds, as we also show in our intuitive example in Figure 4.3. Only a few baselines succeed in other scenarios with different $\mathbf{A}$ and $\mathbf{X}$, as we present in the observations below.

Observation 4.1: Summary

SLIMG wins on all sanity checks without failures (i.e., no red cells). SLIMG shows the best average accuracy and the average rank compared to 15 competitors.

Observation 4.2: No Network Effects

Only a few models including S^2GC and GraphSAGE perform well in uniform $\mathbf{A}$, where the graph structure provides no useful information, since they have the raw $\mathbf{X}$ (not multiplied with $\mathbf{A}$) in their propagator functions $\mathcal{P}$.

Observation 4.3: Useless Features

None of the existing models in Table 4.4 succeeds with useless (i.e., random) features $\mathbf{X}$, since their propagator functions $\mathcal{P}$ rely on $\mathbf{X}$ in all cases.

Observation 4.4: Heterophily Graphs

Models that can utilize even-hop neighbors, such as G^2CN, GraphSAGE, and GPR-GNN, succeed in heterophily graphs either with semantic or structural $\mathbf{X}$.

4.6 Experiments

We conduct experiments on 13 real-world datasets to answer following research questions (RQ):

RQ1. **Accuracy:** How well does SLIMG work for semi-supervised node classification on real-world graphs?

RQ2. **Success of Simplicity:** How does SLIMG succeed even with its simplicity? What if we add nonlinearity to SLIMG?

RQ3. **Speed and Scalability:** How fast and scalable is SLIMG to large real-world graphs?

Table 4.5: **SLIMG wins** most of the times on 13 real-world datasets (7 homophily and 6 heterophily graphs) against 15 competitors. We color the best and worst results as green and red, respectively, as in Table 4.4. SLIMG is the only approach that exhibits no failures (i.e., no red cells) in all datasets. Most competitors cause out-of-memory (OOM) errors on large graphs.

Model	Cora	CiteSeer	PubMed	Comp.	Photo	ArXiv	Products	Cham.	Squirrel	Actor	Penn94	Twitch	Pokec	Avg. Rank
LR	51.5±1.2	52.9±4.5	79.9±0.5	73.9±1.2	79.3±1.5	48.3±1.9	56.4±0.5	24.9±1.7	26.7±1.9	27.8±0.8	63.5±0.5	53.0±0.1	61.3±0.0	11.7 (4.2)
Reg. Kernel	67.8±2.5	62.1±4.4	83.4±1.4	80.3±1.4	87.1±1.2	O.O.M.	O.O.M.	29.4±2.6	24.3±2.3	29.6±1.4	O.O.M.	O.O.M.	O.O.M.	12.2 (3.8)
Diff. Kernel	70.6±1.5	62.7±3.8	82.1±0.4	83.1±1.0	89.8±0.6	O.O.M.	O.O.M.	34.5±7.9	28.3±1.5	24.7±0.9	53.5±0.8	O.O.M.	O.O.M.	11.8 (2.5)
RW Kernel	72.7±1.7	64.1±3.9	83.1±0.7	84.2±0.7	90.6±0.7	63.2±0.2	74.2±0.0	34.9±3.5	25.0±1.6	26.4±1.1	63.1±0.7	57.6±0.1	59.5±0.0	8.3 (3.3)
SGC	76.2±1.1	65.8±3.9	84.1±0.8	83.7±1.6	90.1±0.9	65.0±3.4	74.6±5.1	38.1±4.5	33.1±1.0	24.6±0.8	64.0±1.1	56.5±0.1	69.8±0.0	6.6 (4.2)
DGC	77.8±1.4	66.1±4.2	84.3±0.6	83.9±0.7	90.4±0.2	65.2±4.0	68.7±13.	37.2±3.7	29.2±1.2	25.2±2.1	62.5±0.4	58.2±0.2	60.7±0.1	6.6 (3.2)
S^2GC	78.3±1.5	66.9±4.4	84.3±0.3	83.1±0.8	90.1±0.8	62.0±7.4	58.3±18.	34.9±4.9	27.6±1.8	26.7±1.8	63.1±0.5	58.7±0.1	61.2±0.0	6.6 (2.7)
G^2CN	76.6±1.5	64.2±3.3	81.4±0.6	82.8±1.6	88.8±0.5	O.O.M.	O.O.M.	40.7±2.9	32.1±1.5	24.3±0.5	O.O.M.	O.O.M.	O.O.M.	10.5 (4.5)
GCN	76.0±1.2	65.0±2.9	84.3±0.5	85.1±0.9	91.6±0.5	62.8±0.6	O.O.M.	38.5±3.0	31.4±1.8	26.8±0.4	62.9±0.7	57.0±0.1	63.9±0.4	6.3 (2.4)
SAGE	74.6±1.3	63.7±3.6	82.9±0.4	83.8±0.5	90.6±0.5	61.5±0.6	O.O.M.	39.8±4.3	27.0±1.3	27.8±0.9	O.O.M.	56.6±0.4	68.9±0.1	8.5 (3.5)
GCNII	77.8±1.7	63.4±3.0	84.9±0.8	82.3±1.8	90.8±0.6	45.7±0.5	O.O.M.	30.5±2.5	21.9±3.0	29.0±1.3	64.5±0.5	56.9±0.6	62.1±0.3	8.4 (4.6)
H^2GCN	77.6±0.9	64.7±3.8	85.4±0.4	49.5±16.	75.8±11.	O.O.M.	O.O.M.	31.9±2.6	25.0±0.5	28.9±0.6	63.9±0.4	58.7±0.0	O.O.M.	8.9 (4.9)
APPNP	80.0±0.6	67.1±2.8	84.6±0.5	84.2±1.7	92.5±0.3	53.4±1.3	O.O.M.	30.9±4.7	23.9±3.2	26.1±1.0	63.7±0.9	47.3±0.3	57.4±0.4	7.6 (4.8)
GPR-GNN	78.8±1.3	64.2±4.0	85.1±0.7	85.0±1.0	92.6±0.3	58.5±0.8	O.O.M.	31.7±4.7	26.2±1.6	29.5±1.1	64.5±0.4	57.6±0.2	67.6±0.1	5.4 (3.7)
GAT	78.2±1.2	65.8±4.0	83.6±0.2	85.4±1.4	91.7±0.5	58.2±1.0	O.O.M.	39.1±4.1	28.6±0.6	26.4±0.4	60.5±0.8	O.O.M.	O.O.M.	7.5 (3.7)
SLIMG	77.8±1.1	67.1±2.3	84.6±0.5	86.3±0.7	91.8±0.5	66.3±0.3	84.9±0.0	40.8±3.2	31.1±0.7	30.9±0.6	68.2±0.6	59.7±0.1	73.9±0.1	1.9 (1.5)

Table 4.6: **SLIMG effectively combines different components.** SLIMG-Ci represents that we use only the i-th component shown in Equation (4.3). Node classification is done accurately even we use a single component at each time, and SLIMG outperforms the best accuracy of a single component in 11 out of the 13 datasets. Green (■, ■) marks the top two.

Model	Cora	CiteSeer	PubMed	Comp.	Photo	ArXiv	Products	Cham.	Squirrel	Actor	Penn94	Twitch	Pokec
SLIMG-C1	46.3±3.0	29.2±2.5	64.5±1.0	77.6±1.0	78.5±0.9	51.4±0.2	73.6±2.9	41.9±2.0	29.1±1.0	21.6±1.2	60.9±0.6	59.3±0.1	66.7±0.0
SLIMG-C2	53.5±1.5	53.6±3.6	79.3±0.3	74.5±1.1	81.4±0.9	49.8±0.2	57.4±0.1	25.1±1.5	21.8±0.9	29.9±2.1	62.3±0.5	53.0±0.1	61.1±0.0
SLIMG-C3	77.6±0.7	62.7±4.3	77.4±0.8	86.0±1.0	90.3±0.8	66.2±0.2	82.5±0.0	40.6±1.0	27.6±3.2	24.1±1.4	64.7±0.6	53.1±0.1	73.2±0.1
SLIMG-C4	76.8±0.9	64.7±3.6	82.1±0.7	85.3±1.2	90.9±0.7	65.3±0.3	78.3±0.1	40.4±2.2	31.6±1.4	23.7±1.4	64.3±0.7	56.3±0.1	68.4±0.2
SLIMG	77.8±1.1	67.1±2.3	84.6±0.5	86.3±0.7	91.8±0.5	66.3±0.3	84.9±0.0	40.8±3.2	31.1±0.7	30.9±0.6	68.2±0.6	59.7±0.1	73.9±0.1

RQ4. **Interpretability:** How to explain the importance of graph signals through the learned weights of SLIMG?

RQ5. **Ablation Study:** Are all the design decisions of SLIMG, such as two-hop aggregation, effective in real-world graphs?

Flexibility Table 4.6 illustrates the accuracy of SLIMG when only one of its four components in Equation (4.3) is used at each time. The accuracy of SLIMG with only a single component is higher than those of most baselines in Table 4.5 when an appropriate component is picked for each dataset, e.g., C1 for Twitch, C2 for Actor, C3 for Cora, and C4 for CiteSeer. This shows that high accuracy in semi-supervised node classification can be achieved by a well-designed simple model even without high *expressivity*. SLIMG focuses on the best component in each dataset effectively, improving the accuracy of individual components in 11 out of 13 datasets.

Datasets We use 7 homophily and 6 heterophily graph datasets in experiments, which were commonly used in previous works on node classification [CPLM21, PWC+20, LHL+21]. Cora, CiteSeer, and PubMed [SNB+08, YCS16] are homophily citation graphs between research ar-

Table 4.7: **Linearity is enough for SLIMG:** it outperforms its own variants with nonlinearity that replace the linear classifier or the PCA function g with a nonlinear neural network. Green (■, ■) marks the top two.

Model	Cora	CiteSeer	PubMed	Comp.	Photo	ArXiv	Products	Cham.	Squirrel	Actor	Penn94	Twitch	Pokec
w/ MLP-2	65.9±1.1	54.2±5.3	83.3±0.2	84.8±0.5	90.1±1.9	65.8±0.1	85.7±0.0	40.0±2.5	30.5±0.5	28.8±1.0	66.7±1.7	60.3±0.3	76.5±0.1
w/ MLP-3	66.1±2.0	50.3±3.6	80.9±0.9	85.2±0.8	90.0±0.8	63.0±0.1	84.9±0.9	38.5±5.5	30.9±0.7	28.9±1.2	65.1±0.6	60.3±0.2	76.4±0.2
w/ NL Trans.	70.7±2.3	57.5±5.1	81.0±0.4	71.4±10.	77.9±2.2	57.0±0.6	O.O.M.	41.3±3.2	30.0±1.6	27.6±2.4	61.8±1.6	61.5±0.3	75.7±0.5
SLIMG	77.8±1.1	67.1±2.3	84.6±0.5	86.3±0.7	91.8±0.5	66.3±0.3	84.9±0.0	40.8±3.2	31.1±0.7	30.9±0.6	68.2±0.6	59.7±0.1	73.9±0.1

ticles. Computers and Photo [SMBG18] are homophily Amazon co-purchase graphs between items. ogbn-arXiv and ogbn-Products are large homophily graphs from Open Graph Benchmark [HFZ+20]. Since we use only 2.5% of all labels as training data, we omit the classes with instances fewer than 100. Chameleon and Squirrel [RAS21] are heterophily Wikipedia web graphs. Actor [TSWY09] is a heterophily graph connected by the co-occurrence of actors on Wikipedia pages. Penn94 [TMP11, LHL+21] is a heterophily graph of gender relations in a social network. Twitch [RS21] and Pokec [LK14] are large graphs, which have been relabeled by [LHL+21] to be heterophily. We make the heterophily graphs undirected as done in [CPLM21].

Evaluation We perform semi-supervised node classification by randomly dividing all nodes in a graph by the 2.5%/2.5%/95% ratio into training, validation, and test sets. In this setting [CPLM21, DWCL22], which is common in real-world data where labels are often scarce and expensive, we can properly evaluate the performance of each method in semi-supervised learning. We perform five runs of each experiment with different random seeds and report the average and standard deviation. All hyperparameter searches and early stopping are done based on validation accuracy for each run.

Competitors and Hyperparameters We include various types of competitors: linear models (LR, SGC, DGC, S^2GC, and G^2CN), coupled nonlinear models (GCN, GraphSAGE, GCNII, and H_2GCN), decoupled models (APPNP and GPR-GNN), and attention-based models (GAT). We perform row-normalization on the node features as done in most studies on GNNs. We search their hyperparameters for every data split through a grid search. The hidden dimension size is set to 64, and the dropout probability is set to 0.5. For the linear models, we use L-BFGS to train 100 epochs with the patience of 5. For the nonlinear ones, we use ADAM and update them for 1000 epochs with the patience of 200.

We also include three graph kernel methods [SK03], namely Regularized Laplacian, Diffusion Process, and the K-step Random Walk. They focus on feature transformation and are used with the LR classifier as SLIMG is. Thus, they perform as the direct competitors of SLIMG that apply different feature transformations. The propagator functions of graph kernel methods [SK03] are given as follows:

$$\text{(Reg. Kernel) } \mathcal{P}(\mathbf{A}, \mathbf{X}) = (\mathbf{I}_n + \sigma^2 \tilde{\mathbf{L}})^{-1}\mathbf{X} \tag{4.4}$$

$$\text{(Diff. Kernel) } \mathcal{P}(\mathbf{A}, \mathbf{X}) = \exp(-\sigma^2/2\tilde{\mathbf{L}})\mathbf{X} \tag{4.5}$$

$$\text{(RW Kernel) } \mathcal{P}(\mathbf{A}, \mathbf{X}) = (a\mathbf{I}_n - \tilde{\mathbf{L}})^p\mathbf{X}, \tag{4.6}$$

Table 4.8: **SLIMG is scalable.** We measure the runtime of SLIMG by varying the numbers of features and edges in a graph.

Number of Features	20	40	60	80	100
Time (s)	13.0	16.2	18.6	22.9	27.8
Number of Edges	12M	25M	60M	80M	100M
Time (s)	12.3	19.9	21.9	24.0	27.8

Table 4.9: **Ablation Study: SLIMG works best with the current design.** Each design decision of SLIMG leads to an improvement of accuracy in real-world graphs; SLIMG is among the top two in all datasets, marked as green (,).

Model	**Cora**	**CiteSeer**	**PubMed**	**Comp.**	**Photo**	**ArXiv**	**Products**	**Cham.**	**Squirrel**	**Actor**	**Penn94**	**Twitch**	**Pokec**
w/o Sp. Reg.	77.8±0.6	65.0±3.5	83.8±0.5	85.9±0.8	91.7±0.7	65.2±0.2	83.4±1.9	40.1±3.8	30.7±1.0	30.1±0.6	67.4±0.6	59.8±0.1	74.2±0.0
w/o PCA	74.8±1.5	66.0±3.1	84.7±0.5	84.4±1.1	90.3±0.7	60.8±0.2	84.5±0.0	41.3±2.0	31.8±1.1	27.3±1.1	67.7±0.7	59.1±0.2	72.8±0.1
w/o Struct. U	78.1±1.0	67.4±2.5	84.4±0.3	86.0±0.5	92.1±0.4	66.1±0.2	82.5±0.6	37.5±4.2	29.8±0.4	31.3±0.5	65.8±0.6	56.8±0.0	72.8±0.1
SLIMG	77.8±1.1	67.1±2.3	84.6±0.5	86.3±0.7	91.8±0.5	66.3±0.3	84.9±0.0	40.8±3.2	31.1±0.7	30.9±0.6	68.2±0.6	59.7±0.1	73.9±0.1

where $\tilde{\mathbf{L}} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2}$ is the normalized Laplacian matrix, and $\sigma = 1$, $a = 1$, and $p = 2$ are their hyperparameters.

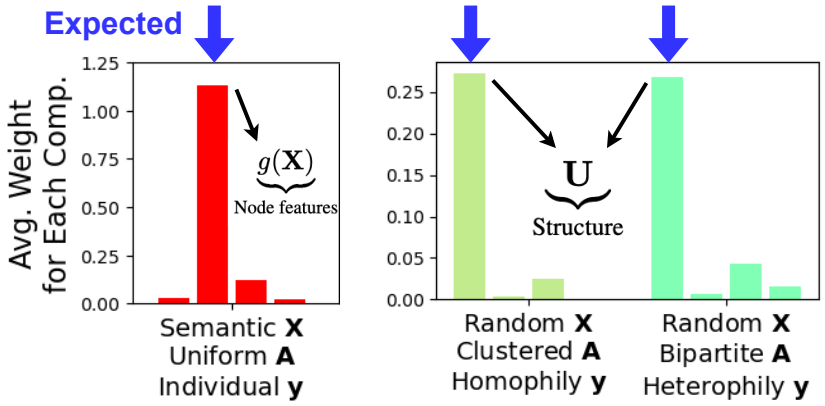
SLIMG contains two hyperparameters, which are the weight of LASSO and the weight of group LASSO. It is worth noting that SLIMG does not need to recompute the features when searching the hyperparameters, while most of the linear methods need to do so because of including one or more hyperparameters in $\mathcal{P}$.

4.6.1 RQ1 – Accuracy

In Table 4.5, SLIMG is compared against 15 competitors on 13 real-world datasets (7 homophily and 6 heterophily graphs). We color the best and worst results as green and red, respectively, as in Table 4.4. We report the accuracy in Table 4.5 where , , represent the top three methods (higher is darker), SLIMG outperforms all competitors in 4 homophily and 5 heterophily graphs, and achieves competitive accuracy in the rest; SLIMG is among the top three in 10 out of 13 times. Moreover, SLIMG is the only model that exhibits no failures (i.e., no red cells), and shows the best average rank with a significant difference from the second-best. It is notable that many competitors, even linear models such as the kernel methods and G^2CN, run out of memory in large graphs. This shows that linearity is not a sufficient condition for efficiency and scalability, and thus a careful design of the propagator function $\mathcal{P}$ is needed as in SLIMG.

4.6.2 RQ2 – Success of Simplicity

We conduct studies to better understand how SLIMG exhibits the superior performance on real-world graphs even with its simplicity. Tables 4.6 and 4.7 demonstrate the strength of simplicity for semi-supervised node classification in real-world graphs.



(a) No Network Effects (b) Useless Features

Figure 4.5: **SLIMG is interpretable:** it suppresses useless information and focuses on informative ones for each scenario: (a) self-features $g(\mathbf{X})$ and (b) structural features $\mathbf{U}$. The learned weights are directly matched with the expectations.

Table 4.10: **Ablation Study: Two-step aggregation is good enough for SLIMG.** The values k_{row} and k_{sym} represent the numbers of propagation steps for the $\mathbf{A}_{\text{row}}$ and $\tilde{\mathbf{A}}_{\text{sym}}$ components in Equation (4.3), respectively. We observe no consistent improvement of accuracy by increasing k_{row} and k_{sym}, supporting the current design of SLIMG. Green (■, ■) marks the top two.

k_{row}	k_{sym}	Cora	CiteSeer	PubMed	Comp.	Photo	ArXiv	Products	Cham.	Squirrel	Actor	Penn94	Twitch	Pokec
$\{2,4,6\}$	$\{2,3,4\}$	79.4±1.1	66.0±4.4	84.4±0.4	85.9±0.5	91.3±0.5	67.9±0.2	84.8±2.1	41.0±3.9	31.4±0.6	29.8±0.6	68.2±0.6	60.7±0.1	76.6±0.2
$\{2,4\}$	$\{2,3\}$	78.9±0.9	66.9±2.5	84.0±0.5	86.2±0.7	91.5±0.5	67.4±0.1	73.9±23.	41.4±4.0	31.0±0.7	30.4±0.4	68.3±0.5	60.8±0.1	76.0±0.1
6	4	79.2±0.7	66.2±3.4	84.0±0.3	85.2±0.7	91.0±0.8	67.3±0.2	84.7±1.7	38.2±6.3	29.2±1.5	31.2±0.8	67.7±0.7	59.8±0.1	74.2±0.2
4	3	79.2±0.8	66.1±3.5	84.2±0.5	85.8±0.6	91.3±0.4	67.2±0.1	85.4±0.0	38.6±7.1	29.5±1.8	30.4±0.7	67.5±0.6	60.0±0.1	74.6±0.1
2 (ours)	2 (ours)	77.8±1.1	67.1±2.3	84.6±0.5	86.3±0.7	91.8±0.5	66.3±0.3	84.9±0.0	40.8±3.2	31.1±0.7	30.9±0.6	68.2±0.6	59.7±0.1	73.9±0.1

Nonlinearity Many recent works on linear GNNs have shown that nonlinearity is not an essential component in semi-supervised node classification [ZK21, WWYL21, LGW+22]. To support the success of SLIMG, we design three nonlinear variants of it:

- **w/ MLP-2:** We replace LR with a 2-layer MLP.
- **w/ MLP-3:** We replace LR with a 3-layer MLP.
- **w/ Nonlinear (NL) Transformation:** We replace the PCA function $g(\cdot)$ as a nonlinear function. Specifically, we adopt a 2-layer MLP for the first two components and a 2-layer GCN for the last two components. The transformed features are concatenated and given to another 2-layer MLP.

We use dropout with a probability of 0.5 to prevent overfitting in both MLP and GCN. The nonlinear models are trained with the same setting as GCN reported in Table 4.5. We report the result in Table 4.7, showing that adding nonlinearity does not necessarily improve the accuracy while sacrificing both scalability and interpretability.

4.6.3 RQ3 – Speed and Scalability

We plot the training time versus the accuracy of each model on the ogbn-arXiv, ogbn-Products, and Pokec graphs, which are the largest in our benchmark, in Figure 4.2. We report the training time of each model with the hyperparameters that show the highest validation accuracy. SLIMG achieves the highest accuracy in the ogbn-arXiv and ogbn-Products datasets while being 10.4× and 2.5× faster than the second-best model, respectively. SLIMG also shows the highest accuracy in the Pokec dataset, while being 18.0× faster than the best-performing deep model. It is worth noting that SLIMG is even faster than LR in ogbn-arXiv, taking fewer iterations than in LR during the optimization. Its fast convergence is owing to the orthogonalization of each component of the features.

Table 4.8 shows the training time of SLIMG in the ogbn-Products graph with varying numbers of features and edges. Smaller graphs are created by removing edges uniformly at random. SLIMG scales well with both variables even in large graphs containing up to 61M edges, showing linear complexity as we claim in Lemma 4.2.

4.6.4 RQ4 – Interpretability

Figure 4.5 illustrates the weights learned by the classifier in SLIMG for the sanity checks, where the ground truths are known. SLIMG assigns large weights to the correct factors in graphs with different mutual information between variables. When there are no network effects in Figure 4.5a, it successfully assigns the largest weights to the self-features $g(\mathbf{X})$, ignoring all other components. When the features are useless in Figure 4.5b, it puts most of the attention on the structural features $\mathbf{U}$, which does not rely on $\mathbf{X}$.

4.6.5 RQ5 – Ablation Studies

Design Decisions In Table 4.9, we show the accuracy of SLIMG when each of its core ideas is disabled: sparse regularization, PCA, and the structural features $\mathbf{U}$. SLIMG performs best with all of its ideas enabled; it is always included in the top two in each dataset. This shows that SLIMG is designed effectively with ideas that help improve its accuracy on real-world datasets. Note that the sparse regularization and PCA improve the efficiency of SLIMG, by reducing the number of parameters, as well as its accuracy.

Receptive Fields To analyze the effect of changing the receptive field of SLIMG, we vary the distance of aggregation (i.e., the value of K) in Table 4.10. The values of k_{row} or k_{sym} given as sets, e.g., $\{2, 4, 6\}$, represent that we include more than one component to SLIMG with different K, increasing the overall complexity of decisions. Since $\mathbf{A}_{\text{row}}$ is designed to consider heterophily relations, we use only the even values of k_{row}. Table 4.10 shows no significant gain in accuracy by increasing the values of k_{row} and k_{sym}, or even including more components to SLIMG; the accuracies of all variants are similar to each other in all cases. That is, SLIMG works sufficiently well even with the 2-step aggregation for both $\mathbf{A}_{\text{row}}$ and $\tilde{\mathbf{A}}_{\text{sym}}$.

4.7 Conclusion

We propose SlimG, a simple but effective model for semi-supervised node classification, which is designed by the *careful simplicity* principle. We summarize our contributions as follows:

- **C1 – Method:** SlimG outperforms state-of-the-art GNNs in synthetic and real datasets, showing the best robustness; SlimG succeeds in all types of graphs with homophily, heterophily, noisy features, etc. SlimG is scalable to million-scale graphs, even when other baselines run out of memory, making interpretable decisions from its linearity.
- **C2 – Explanation:** Our GnnExp framework illuminates the fundamental similarities and differences of popular GNN variants (see Table 4.3), revealing their pain points.
- **C3 – Sanity checks:** Our sanity checks immediately highlight the strengths and weaknesses of each GNN method before it is sent to production (see Table 4.4).
- **C4 – Experiments:** Our extensive experiments explain the success of SlimG in various aspects: linearity, robustness, receptive fields, and ablation studies (see Tables 4.5 to 4.10).

Chapter 5

NetInfoF: Measuring and Exploiting Network Usable Information

Chapter based on work that appeared at ICLR 2024 [LYZ+24] [PDF].

Given an attributed graph, and a graph task (link prediction or node classification), can we tell if a graph neural network (GNN) will perform well? More specifically, do the graph structure and node features carry enough usable information for the task? Our goals are: (1) to develop a fast tool to measure how much information is in the graph structure and in the node features, and (2) to exploit the information to solve the task, if there is enough.

In this chapter, we propose NetInfoF, a framework including NetInfoF_Probe and NetInfoF_Act, for the measurement and the exploitation of network usable information (NUI), respectively. Given a graph data, NetInfoF_Probe measures NUI without any model training, and NetInfoF_Act solves link prediction and node classification, while two modules share the same backbone. In summary, NetInfoF has following notable advantages: (a) *General*, handling both link prediction and node classification; (b) *Principled*, with theoretical guarantee and closed-form solution; (c) *Effective*, thanks to the proposed adjustment to node similarity; (d) *Scalable*, scaling linearly with the input size.

In our carefully designed synthetic datasets, NetInfoF correctly identifies the ground truth of NUI and is the only method being robust to all graph scenarios. Applied on real-world datasets, NetInfoF wins in *11 out of 12* times on link prediction compared to general GNN baselines.

5.1 Introduction

Given a graph with node features, how to tell if a graph neural network (GNN) can perform well on graph tasks or not? How can we know what information (if any) is usable to the tasks, namely, link prediction and node classification? GNNs [KW17, HYL17, VCC+18, LZX+21] are commonly adopted on graph data to generate low-dimensional representations that are versatile for performing different graph tasks. However, sometimes there are no network effects, and training a GNN will be a waste of computation time. That is to say, we want a measurement of how informative the graph structure and node features are for the task at hand, which we call *network usable information (NUI)*.

We propose NetInfoF, a framework to measure and exploit NUI in a given graph. First, NetInfoF_Probe measures NUI of the given graph with NetInfoF_Score (Equation (5.10)), which we proved is lower-bound the accuracy (Theorem 5.2). Next, our NetInfoF_Act solves both the link prediction and node classification by sharing the same backbone with NetInfoF_Probe. To save training effort, we propose to compute NetInfoF_Score by representing different components of the graph with carefully derived node embeddings. For link prediction, we propose the adjustment to node similarity with a closed-form formula to address the limitations when the embeddings are static. We demonstrate that our derived embeddings contain enough usable information, by showing the superior performance on both tasks. In Figure 5.1, NetInfoF_Act outperforms the GNN baselines most times on link prediction; in Figure 5.2, NetInfoF_Score measured by NetInfoF_Probe highly correlates to the test performance in real-world datasets.

In summary, our proposed NetInfoF has following advantages:

1. **General**, handling both node classification and link prediction (Lemma 5.1 and 5.2);
2. **Principled**, with theoretical guarantee (Theorem 5.1 and 5.2) and closed-form solution (Lemma 5.1 and 5.2);
3. **Effective**, thanks to the proposed adjustment of node similarity (Figure 5.1);
4. **Scalable**, scaling linearly with the input size (Figure 5.6).

In synthetic datasets, NetInfoF correctly identifies the ground truth of NUI and is the only method being robust to all possible graph scenarios; in real-world datasets, NetInfoF wins in *11 out of 12* times on link prediction compared to general GNN baselines.

Reproducibility: Code is at `https://github.com/amazon-science/Network-Usable-Info-Framework`.

5.2 Related Work

We introduce the related work in two groups: information theory, and GNNs. In a nutshell, NetInfoF is the only one fulfills all the properties as shown in Table 5.1.

5.2.1 Information Theory

The typical measure of the dependence between the random variables is the mutual information [KSG04]. It is powerful and widely used in sequential feature selection [LCW+18], but its exact computation is difficult [Pan03, BBR+18] especially on continuous random variables [Ros14,

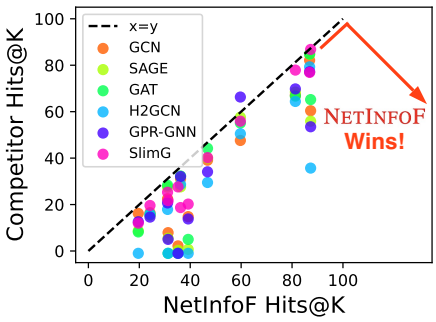
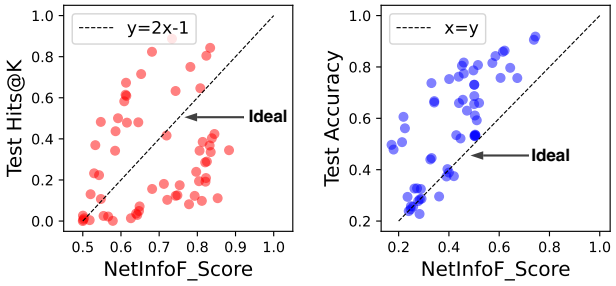


Figure 5.1: **NetInfoF wins** in real-world datasets on link prediction (most points are below or on line $x = y$).



(a) Link Prediction (b) Node Classification

Figure 5.2: **NetInfoF_Score highly correlates to test performance** in real-world datasets. Each point denotes the result of a component from each dataset.

MS21] and high-dimensional data [FWV06, MT19]. Recently [XZS+20, ECS22] proposed the concept of $\mathcal{V}$-information. However, the definition needs a trained model, which is expensive to obtain and is dependent on the quality of training.

Only a few works study the usable information in the graphs, but are not feasible in our problem settings because of three challenges, i.e., our desired method has to: (1) work without training any models, where [ALB23] requires model training; (2) identify which components of the graph are usable, where [DK23] ignores the individual components; and (3) generalize to different graph tasks, where [LSYF24] focuses on node classification only.

5.2.2 Graph Neural Networks

Our review of GNNs in Section 4.2.2 focuses on models for node classification; here, we also include those designed for link prediction. Although most GNNs learn node embeddings assuming homophily, some GNNs [APK+19, ZYZ+20, CPLM21, LWJ22] break this assumption by handling k-step-away neighbors differently for every k, and some systematically study the heterophily graphs on node classification [PKBP23, LHL+22, LHX+23, MCJ+23, CCG+24, MLST22]. Subgraph GNNs [ZC18, YZW+22] are designed only for link prediction and are expensive in inference. Linear GNNs [WJZ+19, WWYL21, ZK21, LGW+22, YLSF23] target interpretable models. Such approaches remove the non-linear functions and maintain good performance. As the only method being robust to all graph scenarios, SlimG [YLSF23] works well on node classification. However, it is unclear how well it works for link prediction.

5.3 NetInfoF_Score: Would a GNN work?

How to tell whether a GNN will perform well on the given graph task? A graph data is composed of more than one component, such as graph structure and node features. In this section, we define our problem, and answer two important questions: (1) How to measure the predictive information of each component in the graph? (2) How to connect the graph information with

Table 5.1: **NetInfoF matches all properties**, while baselines miss more than one property.

Property		General GNNs	Subgraph GNNs	SlimG	NetInfoF
1. General w.r.t. Graph Task	1.1. Node Classification	✔		✔	✔
	1.2. Link Prediction	✔	✔		✔
2. Principled	2.1. Theoretical Guarantee				✔
	2.2. Closed-Form Solution			✔	✔
3. Scalable		✔		✔	✔
4. Robust w.r.t. Input Scenario	4.1. Node Classification			✔	✔
	4.2. Link Prediction				✔

the performance metric on the task? We identify that a GNN is able to perform well on the task when its propagated representation is more informative than graph structure or node features.

5.3.1 Problem Definition

Given an undirected graph $G = (\mathcal{V}, \mathcal{E})$ with node features $\mathbf{X}_{n\times f}$, where n is the number of nodes and f is the number of features, the problem is defined as follows:

- **Measure** the network usable information (NUI), and
- **Exploit** NUI, if there is enough, to solve the graph task.

We consider two common graph tasks, link prediction and node classification. In link prediction, $\mathcal{E}$ is split into $\mathcal{E}_{\text{train}}$ and $\mathcal{E}_{\text{pos}}$. The negative edge set $\mathcal{E}_{\text{neg}}$ is randomly sampled with the same size of $\mathcal{E}_{\text{pos}}$. The goal is to predict the existence of the edges, 1 for the edges in $\mathcal{E}_{\text{pos}}$, and 0 for the ones in $\mathcal{E}_{\text{neg}}$. In node classification, $|\mathcal{V}_{\text{train}}|$ node labels $\mathbf{y} \in \{1, ..., c\}^{|\mathcal{V}_{\text{train}}|}$ are given, where c is the number of classes. The goal is to predict the rest $|\mathcal{V}| - |\mathcal{V}_{\text{train}}|$ unlabeled nodes' classes. An overview of symbols and acronyms is provided in Table 5.2 and Table 5.3, respectively.

5.3.2 Proposed Derived Node Embeddings

To tell whether a GNN will perform well, we can analyze its node embeddings, but they are only available after training. For this reason, we propose to analyze the derived node embeddings in linear GNNs. More specifically, similar to SlimG in Chapter 4, we derive five components of node embeddings that represent the information of graph structure, node features, and features propagated through structure.

C1: Structure Embedding. The structure embedding $\mathbf{U}$ is the left singular vector of the adjacency matrix $\mathbf{A}$, which is extracted by the singular value decomposition (SVD). This aims to capture the community information of the graph.

Table 5.2: Table of Symbols and Definitions.

Symbol	Definition
G	Undirected and attributed graph
$\mathcal{V}$	Set of nodes
$\mathcal{E}$	Set of edges
n	Number of nodes
f	Number of features
c	Number of classes
d	Number of dimensions
$\mathbf{y}$	Set of node labels
$\mathbf{A} \in \mathbb{R}^{n \times n}$	Adjacency matrix
$\mathbf{X} \in \mathbb{R}^{n \times f}$	Node feature matrix
$\mathbf{I} \in \mathbb{R}^{n \times n}$	Identity matrix
$\mathbf{D} \in \mathbb{R}^{n \times n}$	Degree matrix with node degrees along diagonal
$\mathbf{H} \in \mathbb{R}^{d \times d}$	Compatibility matrix for adjusting node similarity

Table 5.3: Table of Acronyms.

Acronym	Definition
NUI	Network Usable Information
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
PPR	Personalized PageRank
LR	Logistic Regression
GNN	Graph Neural Network
GCN	Graph Convolutional Networks
SGC	Simple Graph Convolution

C2: Neighborhood Embedding. The neighborhood embedding $\mathbf{R}$ aims to capture the local higher-order neighborhood information of nodes. This component is designed for link prediction and therefore not used by SlimG. Similarly to our random walk approach in Section 3.5.1, to mimic Personalized PageRank (PPR), we construct a random walk matrix $\mathbf{A}_{\text{PPR}}$, where each element is the number of times a node visits another node in T trials of the k_{PPR}-step random walks. Random walks highlight local higher-order structures in the vicinity of nodes. To make $\mathbf{A}_{\text{PPR}}$ sparser and to speed up embedding extraction, we eliminate noisy elements with only one visit time. We extract the left singular vectors of $\mathbf{A}_{\text{PPR}}$ by SVD as neighborhood embeddings $\mathbf{R}$.
C3: Feature Embedding. Given the raw node features $\mathbf{X}$, we represent the feature embedding with the preprocessed node features $\mathbf{F} = g(\mathbf{X})$, where g is the preprocessed function.
C4: Propagation Embedding without Self-loop. We row-normalize the adjacency matrix into $\mathbf{A}_{\text{row}} = \mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix. The features are propagated without self-loop to capture the information of k_{row}-step neighbors, where k_{row} is an even number. This is useful to capture the information of similar neighbors when the structure exhibits heterophily (e.g., in a bipartite graph). Therefore, we have node embedding $\mathbf{P} = g(l(\mathbf{A}_{\text{row}}^2\mathbf{X}))$, where l is the column-wise L2-normalization, ensuring every dimension has a similar scale.
C5: Propagation Embedding with Self-loop. The adjacency matrix with self-loop is useful to propagate features in graphs that exhibit homophily. Following the most common strategy, we symmetrically normalize the adjacency matrix into $\tilde{\mathbf{A}}_{\text{sym}} = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}$, where $\mathbf{I}$ is the identity matrix. Similar to C4, we have node embeddings $\mathbf{S} = g(l(\tilde{\mathbf{A}}_{\text{sym}}^{k_{\text{sym}}}\mathbf{X}))$.

While C1-2 aim to capture the information with only the graph structure, C4-5 aim to capture the information of propagation, which is similar to the one that a trained GNN can capture. To ensure that the embeddings have intuitive meanings, we set all the number of steps k_{PPR}, k_{row} and k_{sym} as 2, which works sufficiently well in most cases. As C1-2 adopted SVD as their last step, the embedding dimensions are orthogonalized. For C3-5, we use principal component analysis (PCA) as g to reduce and orthogonalize the embedding dimensions, leading to faster convergence and better performance when training the model. Each component has the same number of dimensions d.

5.3.3 NetInfoF_Score: Definition and Theorems

Next, we want to find a formula that connects the metrics of graph information and task performance. To begin, we derive the inequality between entropy and accuracy:

Theorem 5.1: Entropy and Accuracy

Given a discrete random variable Y, we have:

$$2^{-H(Y)} \leq \text{accuracy}(Y) = \max_{y \in Y} p_y \tag{5.1}$$

where $H(Y) = -\sum_{y \in Y} p_y \log p_y$ denotes the Shannon entropy.

Proof. Let Y be a discrete random variable with n outcomes $(y_1, \ldots, y_n)$, and with probabilities $(p_1, \ldots, p_n)$, where:

$$1 = p_1 + \cdots + p_n \tag{5.2}$$

Let p_{max} be the highest probability (break ties arbitrarily), that is:

$$p_{\text{max}} = \max_i p_i \tag{5.3}$$

For ease of presentation, and without loss of generality, assume that the most likely outcome is the first one, y_1, and thus $p_{max} = p_1$. Given no other information, the best classifier for Y is the one that always guesses outcome y_1, and it has accuracy:

$$\text{accuracy}(Y) = p_1 \equiv p_{\text{max}} \tag{5.4}$$

The entropy $H(Y)$ is:

$$H(Y) = -(p_1 \log p_1 + \cdots + p_n \log p_n) \tag{5.5}$$

Thus we have:

$$\begin{aligned}
2^{-H(Y)} &= p_1^{p_1} * p_2^{p_2} \cdots * p_n^{p_n} && (5.6)\\
&\leq p_{\text{max}}^{p_1} * p_{\text{max}}^{p_2} * \cdots * p_{\text{max}}^{p_n} \quad // \ \because p_{\text{max}} \geq p_i && (5.7)\\
&\leq p_{\text{max}}^{p_1 + p_2 + \cdots + p_n} && (5.8)\\
&\leq p_{\text{max}} && (5.9)
\end{aligned}$$

which completes the proof. ■

Before extending Theorem 5.1 to the case with two random variables, we need a definition:

Definition 5.1: NetInfoF_Score of Y given X

Given two discrete random variables X and Y, NetInfoF_Score of Y given X is defined as:

$$\text{NetInfoF\_Score} = 2^{-H(Y|X)} \tag{5.10}$$

where $H(\cdot|\cdot)$ denotes the conditional entropy.

We prove that NetInfoF_Score low-bounds the accuracy:

Theorem 5.2: NetInfoF_Score

Given two discrete random variables X and Y, NetInfoF_Score of Y given X low-bounds the accuracy:

$$\text{NetInfoF\_Score} = 2^{-H(Y|X)} \leq \text{accuracy}(Y|X) = \sum_{x \in X} \max_{y \in Y} p_{x,y} \tag{5.11}$$

where $p_{x,y}$ is the joint probability of x and y.

Proof. Let Y and X be two discrete random variables with n outcomes $(y_1, \ldots, y_n)$ and m outcomes $(x_1, \ldots, x_m)$, respectively, then their joint probabilities are $(p_{1,1}, \ldots, p_{m,n})$, where:

$$1 = p_1 + \cdots + p_m = \sum_{j=1}^{n} p_{1,j} + \cdots + \sum_{j=1}^{n} p_{m,j} \tag{5.12}$$

The accuracy is:

$$\text{accuracy}(Y|X) = \sum_{i=1}^{m} \max_{j} p_{i,j} \tag{5.13}$$

The conditional entropy $H(Y|X)$ is:

$$H(Y|X) = p_1 * (-\sum_{j=1}^{n} \frac{p_{1,j}}{p_1} * \log_2 \frac{p_{1,j}}{p_1}) + \cdots + p_m * (-\sum_{j=1}^{n} \frac{p_{m,j}}{p_m} * \log_2 \frac{p_{m,j}}{p_m}) \tag{5.14}$$

Thus we have:

$$\Rightarrow -H(Y|X) = p_1 * (\sum_{j=1}^{n} \frac{p_{1,j}}{p_1} * \log_2 \frac{p_{1,j}}{p_1}) + \cdots + p_m * (\sum_{j=1}^{n} \frac{p_{m,j}}{p_m} * \log_2 \frac{p_{m,j}}{p_m}) \tag{5.15}$$

$$\leq p_1 * \log_2 (\sum_{j=1}^{n} (\frac{p_{1,j}}{p_1})^2) + \cdots + p_m * \log_2 (\sum_{j=1}^{n} (\frac{p_{m,j}}{p_m})^2) \quad // \ \because \text{Jensen's Inequality} \tag{5.16}$$

$$\leq p_1 * \log_2 (p_1^{-2} * \sum_{j=1}^{n} {p_{1,j}}^2) + \cdots + p_m * \log_2 (p_m^{-2} * \sum_{j=1}^{n} p_{m,j}^2) \tag{5.17}$$

$$\leq p_1 * \log_2 (p_1^{-1} * \max_{j} p_{1,j}) + \cdots + p_m * \log_2 (p_m^{-1} * \max_{j} p_{m,j}) \tag{5.18}$$

This is because $\forall i = 1, \ldots, m$:

$$\sum_{j=1}^{n} p_{i,j}^2 = p_{i,1} * p_{i,1} + \cdots + p_{i,n} * p_{i,n} \tag{5.19}$$

$$\leq p_{i,1} * \max_j p_{i,j} + \cdots + p_{i,n} * \max_j p_{i,j} \tag{5.20}$$

$$\leq (p_{i,1} + \cdots + p_{i,n}) * \max_j p_{i,j} \tag{5.21}$$

$$= p_i * \max_j p_{i,j} \tag{5.22}$$

To continue, we have:

$$\Rightarrow 2^{-H(Y|X)} \leq 2^{p_1 * \log_2 (p_1^{-1} * \max_j p_{1,j}) + \cdots + p_m * \log_2 (p_m^{-1} * \max_j p_{m,j})} \tag{5.23}$$

$$\leq (p_1^{-1} * \max_j p_{1,j})^{p_1} * \cdots * (p_m^{-1} * \max_j p_{m,j})^{p_m} \tag{5.24}$$

$$\leq \max_j p_{1,j} + \cdots + \max_j p_{m,j} \quad // \ \because \text{Weighted AM-GM Inequality} \tag{5.25}$$

$$= \text{accuracy}(Y|X) \tag{5.26}$$

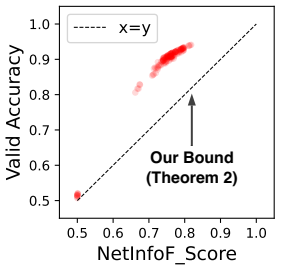
which completes the proof. ■

Theorem 5.2 provides an advantage to NetInfoF_Score by giving it an intuitive interpretation, which is the lower-bound of the accuracy. When there is little usable information to the task, the value of NetInfoF_Score is close to random guessing. To empirically verify it, we run the experiments on the synthetic datasets with five splits, and report NetInfoF_Score and accuracy for all components of the derived embeddings. In Figure 5.3, we find that even for the validation set, NetInfoF_Score is always less than or equal to the accuracy, strictly following Theorem 5.2. In the next sections, we show how NetInfoF_Score can be effectively and efficiently computed with our proposed NetInfoF_Probe.

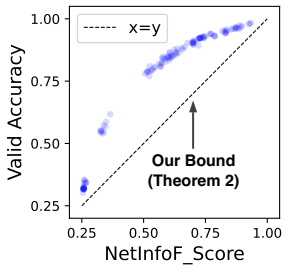
Pinsker's inequality [Pin64], like our Theorem 5.2, also links information-theoretic quantities with probabilistic bounds. However, they differ in that Pinsker's inequality provides an upper bound on the total variation distance between two probability distributions, while NetInfoF_Score offers a lower bound on accuracy. Moreover, in classification tasks, accuracy is generally more intuitive and interpretable for users than the total variation distance.

5.4 NetInfoF for Link Prediction

With the derived node embeddings, how can we measure NUI in link prediction as well as solve the task? Compared to general GNNs, the node embeddings of linear GNNs are given by closed-form formula. They are thus rarely applied on link prediction because of following two reasons: (1) Predicting links by GNNs relies on measuring node similarity, which is incorrect if the neighbors are expected to have dissimilar embeddings; for example, in a bipartite graph, while a source node is connected to a target node, their structural embeddings are expected to be very different, resulting in low node similarity by linear GNNs; (2) To perform well on Hits@K, it is crucial to suppress the similarity of the nodes of negative edges, i.e. the unexisting

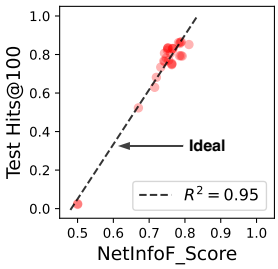


(a) Link Prediction

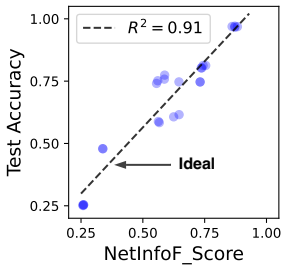


(b) Node Classification

Figure 5.3: **Theorem 5.2 holds**. The value of NetInfoF_Score is always less than or equal to validation accuracy.



(a) Link Prediction



(b) Node Classification

Figure 5.4: **NetInfoF_Score predicts right.** It is correlated to the test performance in the synthetic datasets.

connections in the graph. Hits@K is the ratio of positive edges that are ranked at K-th place or above among both the positive and negative edges, which is preferred in link prediction where most real-world applications are recommendations. Since the embeddings of linear GNNs are static, they can not learn to separate the embeddings of nodes on each side of the negative edges. Therefore, how to generalize linear GNNs to solve link prediction remains a challenge.

For these reasons, we propose an adjustment to the similarity of the nodes, which generalizes NetInfoF to link prediction, including NetInfoF_Probe to measure NUI and NetInfoF_Act to solve the task.

5.4.1 Proposed Adjustment to Node Similarity

To solve the limitations of linear GNNs on link prediction, it is crucial to properly measure the similarity between nodes. We consider cosine similarity as the measurement, whose value is normalized between 0 and 1. By L2-normalizing each node embedding $\mathbf{z}_{1\times d}$, the cosine similarity reduces to a simple dot product $\mathbf{z}_i \cdot \mathbf{z}_j$. However, even if node i and node j are connected by an edge, it may result in low value if they are expected to have dissimilar embeddings (e.g. structure embeddings in a bipartite graph). Therefore, before the dot product, we propose using the compatibility matrix $\mathbf{H}_{d\times d}$ to transform one of the embeddings, and rewrite the node similarity function into $\mathbf{z}_i\mathbf{H}\mathbf{z}_j^\intercal$.

The compatibility matrix $\mathbf{H}$ represents the characteristics of the graph: if the graph exhibits homophily, $\mathbf{H}$ is nearly diagonal; if it exhibits heterophily, $\mathbf{H}$ is off-diagonal. It is used in belief propagation (BP) to handle the interrelations between node classes. In our case, $\mathbf{H}$ represents the interrelations between the dimensions of node embeddings. By maximizing the similarity of nodes connected by edges, $\mathbf{H}$ can be estimated by the following lemma:

Lemma 5.1: Compatibility Matrix

The compatibility matrix $\mathbf{H}$ has the closed-form solution and can be solved by the following optimization problem:

$$\min_{\mathbf{H}} \sum_{(i,j)\in\mathcal{E}} \|\mathbf{z}_i\mathbf{H} - \mathbf{z}_j\|_2^2, \tag{5.27}$$

where $\mathcal{E}$ denotes the set of (positive) edges in the given graph.

Proof. The goal is to maximize the similarity of nodes connected by edges. If we start from cosine similarity and L2-normalize the node embeddings $\mathbf{z}$, we have:

$$\begin{aligned} &\frac{\mathbf{z}_i \boldsymbol{H} \mathbf{z}_j^\intercal}{\|\mathbf{z}_i\|\|\mathbf{z}_j\|} = 1, \forall (i,j) \in \mathcal{E} \\ \Rightarrow &\mathbf{z}_i \boldsymbol{H} \mathbf{z}_j^\intercal = \|\mathbf{z}_i\|\|\mathbf{z}_j\| = 1 \\ \Rightarrow &\mathbf{z}_i \boldsymbol{H} = \mathbf{z}_j \end{aligned} \tag{5.28}$$

Setting $\mathbf{z}_i$ as the input data and $\mathbf{z}_j$ as the target data, this equation can be solved by d-target linear regression with d coefficients, which has the closed-form solution. ■

This optimization problem can be efficiently solved by multi-target linear regression. Nevertheless, this estimation of $\mathbf{H}$ does not take into account negative edges, which may accidentally increase the similarity of negative edges in some complicated cases. This hurts the performance especially when evaluating with Hits@K. Therefore, based on Lemma 5.1, we propose an improved estimation $\mathbf{H}^*$, which further minimizes the similarity of nodes connected by the sampled negative edges:

Lemma 5.2: Compatibility Matrix with Negative Edges

The compatibility matrix with negative edges $\mathbf{H}^*$ has the closed-form solution and can be solved by the following optimization problem:

$$\min_{\mathbf{H}^*} \sum_{(i,j)\in\mathcal{E}} (1 - \mathbf{z}_i\mathbf{H}^*\mathbf{z}_j^\intercal) - \sum_{(i,j)\in\mathcal{E}_{\text{neg}}} (\mathbf{z}_i\mathbf{H}^*\mathbf{z}_j^\intercal), \tag{5.29}$$

where $\mathcal{E}_{\text{neg}}$ denotes the set of negative edges.

Algorithm 5.1: Compatibility Matrix with Negative Edges

Input: Preprocessed node embedding $\hat{\mathbf{Z}}$, edge set $\mathcal{E}$, and sample size S

Extract 2-core edge set $\mathcal{E}_{pos}$ from $\mathcal{E}$;
Estimate $\mathbf{H}$ with $\hat{\mathbf{Z}}$ and $\mathcal{E}_{pos}$ by Lemma 5.1;
if $|\mathcal{E}_{pos}| > S$ **then**
    Sample S edges from $\mathcal{E}_{pos}$;
end
Initialize $\mathbf{H}^*$ with $\mathbf{H}$;
Keep top coefficients in upper triangle of $\mathbf{H}^*$ with 95% energy;
Sample $2|\mathcal{E}_{pos}|$ negative edges as $\mathcal{E}_{neg}$;
Estimate $\mathbf{H}^*$ with $\hat{\mathbf{Z}}$, $\mathcal{E}_{pos}$ and $\mathcal{E}_{neg}$ by Lemma 5.2;
Return $\mathbf{H}^*$;

Proof. We rewrite the adjusted node similarity s from matrix form into a simple computation:

$$\begin{aligned} s(\mathbf{z}_i, \mathbf{z}_j, \boldsymbol{H}) &= \mathbf{z}_i \boldsymbol{H} \mathbf{z}_j^\intercal \\ &= \begin{bmatrix} \mathbf{z}_{i,1} & \cdots & \mathbf{z}_{i,d} \end{bmatrix} \begin{bmatrix} \boldsymbol{H}_{1,1} & \cdots & \boldsymbol{H}_{1,d} \\ \vdots & \ddots & \vdots \\ \boldsymbol{H}_{d,1} & \cdots & \boldsymbol{H}_{d,d} \end{bmatrix} \begin{bmatrix} \mathbf{z}_{j,1} \\ \vdots \\ \mathbf{z}_{j,d} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{z}_{i,1}\mathbf{z}_{j,1} & \cdots & \mathbf{z}_{i,1}\mathbf{z}_{j,d} \\ \vdots & \ddots & \vdots \\ \mathbf{z}_{i,d}\mathbf{z}_{j,1} & \cdots & \mathbf{z}_{i,d}\mathbf{z}_{j,d} \end{bmatrix} \odot \begin{bmatrix} \boldsymbol{H}_{1,1} & \cdots & \boldsymbol{H}_{1,d} \\ \vdots & \ddots & \vdots \\ \boldsymbol{H}_{d,1} & \cdots & \boldsymbol{H}_{d,d} \end{bmatrix} \\ &= \mathbf{z}_{i,1}\mathbf{z}_{j,1}\boldsymbol{H}_{1,1} + \mathbf{z}_{i,1}\mathbf{z}_{j,2}\boldsymbol{H}_{1,2} + \cdots + \mathbf{z}_{i,1}\mathbf{z}_{j,d}\boldsymbol{H}_{1,d} + \cdots + \mathbf{z}_{i,d}\mathbf{z}_{j,d}\boldsymbol{H}_{d,d}, \end{aligned} \tag{5.30}$$

where $\odot$ is the Hadamard product introduced in Definition 2.5. Next, to maximize the similarity of nodes connected by positive edges, and to minimize the similarity of nodes connected by negative edges, we have:

$$s(\mathbf{z}_i, \mathbf{z}_j, \boldsymbol{H}) = \begin{cases} 1 & (i,j) \in \mathcal{E} \\ 0 & (i,j) \in \mathcal{E}_{\text{neg}} \end{cases} \tag{5.31}$$

Therefore, this equation can be solved by linear regression with d^2 coefficients, which has the closed-form solution. ■

With great power comes great responsibility, estimating $\mathbf{H}^*$ has a higher computational cost than estimating $\mathbf{H}$. Thus, we provide three techniques to speed up the computation of $\mathbf{H}^*$ with the help of $\mathbf{H}$, and the details are in Algorithm 5.1.

T1: Warm Start. We approximate the solution by LSQR iteratively and warm up the approximation process with $\mathbf{H}$. Since $\mathbf{H}$ is similar to $\mathbf{H}^*$ and cheap to compute, this step largely speeds up the approximation process and reduces the number of iterations needed for convergence.

T2: Coefficient Selection. We reduce the number of coefficients by only estimating the upper triangle of $\mathbf{H}^*$, and keep the ones with 95% energy in $\mathbf{H}$. This is because the similarity function

Algorithm 5.2: NetInfoF_Probe for Link Prediction

Input: Node embedding $\mathbf{Z}$, train edge set $\mathcal{E}_{train}$, valid edge set $\mathcal{E}_{valid}$, valid edge labels $\mathbf{y}_{valid}$, sample size S, and bin size k

Preprocess $\mathbf{Z}$ into $\hat{\mathbf{Z}}$ by column-wise standardization and row-wise L2-normalization;
$\mathbf{H}^* =$ Compatibility-Matrix-with-Negative-Edges$(\hat{\mathbf{Z}}, \mathcal{E}_{train}, S)$;
if $|\mathcal{E}_{pos}| > S$ **then**
    Sample S edges from $\mathcal{E}_{pos}$;
end
else
    $\mathcal{E}_{pos} \leftarrow \mathcal{E}$;
end
Sample $2|\mathcal{E}_{pos}|$ negative edges as $\mathcal{E}_{neg}$;
Fit k-bins discretizer with $\hat{\mathbf{z}}_i\mathbf{H}^*\hat{\mathbf{z}}_j^\intercal, \forall(i,j) \in \mathcal{E}_{pos} \cup \mathcal{E}_{neg}$;
Discretize $\hat{\mathbf{z}}_i\mathbf{H}^*\hat{\mathbf{z}}_j^\intercal, \forall(i,j) \in \mathcal{E}_{valid}$ into k bins as $\mathbf{s}_{valid}$;
Return NetInfoF_Score, computed with $\mathbf{s}_{valid}$ and $\mathbf{y}_{valid}$ by Equation (5.10);

is symmetric, and the unimportant coefficients with small absolute values in $\mathbf{H}$ remain unimportant in $\mathbf{H}^*$. The absolute sum of the kept coefficients divided by $\sum_{i=1}^{d}\sum_{j=i+1}^{d}|\mathbf{H}_{ij}|$ is 95% and the rest are zeroed out. This helps us to reduce the number of coefficients from d^2 to less than $(d+1)d/2$.

T3: Edge Reduction. We sample S positive edges from the 2-core graph, and $2S$ negative edges, where the sample size S depends on d. Since in large graphs $|\mathcal{E}|$ is usually much larger than d^2, it is not necessary to estimate fewer than $(d+1)d/2$ coefficients with all $|\mathcal{E}|$ edges. Moreover, the 2-core graph remains the edges with stronger connections, where each node in it has at least degree 2. Sampling from the 2-core graph avoids interference from noisy edges and leads to better estimation.

5.4.2 NetInfoF_Probe for NUI Measurement

Based on Theorem 5.2, we propose NetInfoF_Probe that computes NetInfoF_Score, without exactly computing the conditional entropy of the high-dimensional variables. By sampling negative edges, the link prediction can be seen as a binary classification problem. For each component of embeddings, NetInfoF_Probe estimates its corresponding $\mathbf{H}^*$ and discretizes the adjusted node similarity of positive and negative edges. To avoid overfitting, we fit the k-bins discretizer with the similarity of training edges, and discretize the one of validation edges into k bins. NetInfoF_Score can then be easily computed between two categorical variables. For instance, the node similarity between node i and j with embedding $\mathbf{U}$ is $(\hat{\mathbf{U}}_i\mathbf{H}^*_{\hat{\mathbf{U}}}) \cdot \hat{\mathbf{U}}_j$, where $\hat{\cdot}$ denotes the embedding preprocessed by column-wise standardization and row-wise L2-normalization. The details are in Algorithm 5.2.

5.4.3 NetInfoF_Act for NUI Exploitation

To solve link prediction, NetInfoF_Act shares the same derived node embeddings with NetInfoF_Probe, and uses a link predictor following by the Hadamard product of the embeddings. We transform the embeddings on one side of the edge with $\mathbf{H}^*$, which handles the heterophily embeddings and better separates the nodes in the negative edges. By concatenating all components, the input to the predictor is as follows:

$$\underbrace{\hat{\mathbf{U}}_i\mathbf{H}^*_{\hat{\mathbf{U}}} \odot \hat{\mathbf{U}}_j}_{\text{Structure}} \| \underbrace{\hat{\mathbf{R}}_i\mathbf{H}^*_{\hat{\mathbf{R}}} \odot \hat{\mathbf{R}}_j}_{\text{PPR}} \| \underbrace{\hat{\mathbf{F}}_i\mathbf{H}^*_{\hat{\mathbf{F}}} \odot \hat{\mathbf{F}}_j}_{\text{Features}} \| \underbrace{\hat{\mathbf{P}}_i\mathbf{H}^*_{\hat{\mathbf{P}}} \odot \hat{\mathbf{P}}_j}_{\substack{\text{Features of} \\ \text{2-Step Neighbors}}} \| \underbrace{\hat{\mathbf{S}}_i\mathbf{H}^*_{\hat{\mathbf{S}}} \odot \hat{\mathbf{S}}_j}_{\substack{\text{Features of} \\ \text{Grand Neighbors}}} \tag{5.32}$$

where $(i, j) \in \mathcal{E} \cup \mathcal{E}_{\text{neg}}$ and where $\odot$ is the Hadamard product (Definition 2.5). Among all the choices, we use logistic regression as the predictor for its scalability and interpretability. We suppress the weights of useless components, if there is any, by adopting sparse-group LASSO for the feature selection. The time complexity of NetInfoF_Act is:

Lemma 5.3: Time Complexity

The time complexity of NetInfoF_Act for link prediction is linear on the input size $|\mathcal{E}|$:

$$O(f^2|\mathcal{V}| + f^3 + d^4|\mathcal{E}|) \tag{5.33}$$

where f and d are the number of features and embedding dimensions, respectively.

Proof. NetInfoF_Act includes four parts: SVD, PCA, compatibility matrix estimation, and logistic regression. The time complexity of truncated SVD is $O(d^2|\mathcal{V}|)$, and the one of PCA is $f^2|\mathcal{V}| + f^3$. Compatibility matrix estimation is optimized by Ridge regression with regularized least-squares routine, whose time complexity is $d^4|\mathcal{E}|$. The time complexity of training logistic regression is $dt|\mathcal{E}|$, where t is the number of epochs. In our experiments, t is no greater than 100, and $|\mathcal{V}|$ is much less than $|\mathcal{E}|$ in most datasets. By combining all the terms and keeping only the dominant ones, we have time complexity of NetInfoF_Act for link prediction $O(f^2|\mathcal{V}| + f^3 + d^4|\mathcal{E}|)$. ■

5.5 NetInfoF for Node Classification

In this section, we show how we can generalize NetInfoF to node classification. Since node classification does not rely on the node similarity, it does not need compatibility matrix.

5.5.1 NetInfoF_Probe for NUI Measurement

To effectively and efficiently compute NetInfoF_Score, we propose to assign labels to the nodes by clustering. This idea is based on the intuition that good embeddings for node classification can be easily split by clustering. Among clustering methods, we use k-means as it

Algorithm 5.3: NetInfoF_Probe for Node Classification

Input: Train, valid, and test node embedding $\mathbf{Z}_{train}$, $\mathbf{Z}_{valid}$, and $\mathbf{Z}_{test}$, train and valid node labels $\mathbf{y}_{train}$ and $\mathbf{y}_{valid}$, and cluster number k

1 Preprocess $\mathbf{Z}$ into $\hat{\mathbf{Z}}$ row-wise L2-normalization;

2 Fit clustering model with $\hat{\mathbf{Z}}_{test}$;

3 Assign cluster labels $\mathbf{s}_{train}$ and $\mathbf{s}_{valid}$ to $\hat{\mathbf{Z}}_{train}$ and $\hat{\mathbf{Z}}_{valid}$, respectively;

4 Return NetInfoF_Score, computed with $\mathbf{s}_{train} \cup \mathbf{s}_{valid}$ and $\mathbf{y}_{train} \cup \mathbf{y}_{valid}$ by Equation (5.10);

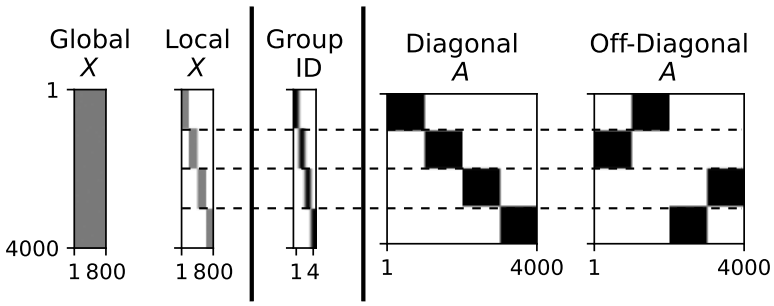


Figure 5.5: Scenarios of node features $\mathbf{X}$ and graph structure $\mathbf{A}$ in synthetic datasets.

is fast. We cluster each component of the embeddings and compute NetInfoF_Score, where $k \geq c$. To ensure that the clustering is done stably, a row-wise L2-normalization is done on the embedding. The details are in Algorithm 5.3.

5.5.2 NetInfoF_Act for NUI Exploitation

To solve node classification, we again concatenate the embeddings of different components, and the input of classifier is as follows:

$$\underbrace{l(\mathbf{U})}_{\text{Structure}} \| \underbrace{l(\mathbf{R})}_{\text{PPR}} \| \underbrace{l(\mathbf{F})}_{\text{Features}} \| \underbrace{l(\mathbf{P})}_{\substack{\text{Features of} \\ \text{2-Step Neighbors}}} \| \underbrace{l(\mathbf{S})}_{\substack{\text{Features of} \\ \text{Grand Neighbors}}} \tag{5.34}$$

where l is the column-wise L2-normalization. Similar to NetInfoF_Act in link prediction, we use logistic regression as the classifier and adopt sparse-group LASSO for the regularization.

5.6 Synthetic Datasets for Sanity Checks

To ensure that NetInfoF is robust to all graph scenarios, we carefully design the synthetic datasets for sanity checks. We include all possible graph scenarios, where the ground truth of NUI is available. For the experiments of node classification in the synthetic datasets, please refer to the paper [LYZ+24].

Table 5.4: **NetInfoF wins** on link prediction in the synthetic datasets. Hits@100 is reported. Green (■) marks the winner.

Model	Rand. X Diag. A	Rand. X Off-Diag. A	Global X Diag. A	Global X Off-Diag. A	Local X Diag. A	Local X Off-Diag.	Avg. Rank
GCN	82.7±1.1	70.9±1.2	87.2±0.5	85.1±1.2	17.4±1.7	19.2±1.8	4.3 (0.9)
SAGE	77.4±1.1	66.5±4.1	86.2±1.0	85.2±2.2	11.0±1.2	09.5±1.0	5.1 (1.1)
GAT	86.3±0.9	83.1±0.3	87.3±0.9	85.2±0.6	16.0±2.0	16.9±2.1	3.3 (1.5)
H^2GCN	24.5±4.3	58.9±3.7	75.3±1.1	85.8±2.5	19.8±2.0	19.2±1.5	5.0 (2.1)
GPR-GNN	75.1±0.8	52.3±1.6	83.4±1.3	79.5±1.6	19.3±1.7	17.1±2.0	6.0 (1.1)
SlimG	85.7±0.8	67.8±2.8	87.9±1.0	85.1±1.3	82.5±1.6	31.1±1.1	3.3 (1.3)
NetInfoF	87.3±0.7	86.7±0.6	89.8±0.3	89.8±1.0	89.6±0.2	90.8±0.7	1.0 (0.0)

5.6.1 Designs

We separate the nodes into c groups to simulate that there are usually multiple communities in a graph. To cover all the possibilities in the real-world, the scenarios are the cross-product of different scenarios on the node features $\mathbf{X}$ and the graph structure $\mathbf{A}$, as shown in Figure 5.5. We ignore the scenario that $\mathbf{X}$ is useful but $\mathbf{A}$ is useless, since this is impractical in the real-world. There are 3 scenarios of node features $\mathbf{X}$:

1. **Random**: the node features are random, with no correlation with the existence of edges.
2. **Global**: all dimensions of the node features are correlated with the existence of edges.
3. **Local**: only a subset of dimensions of the node features are correlated with the existence of edges, where there is no overlapping between the subsets of node groups.

There are 2 scenarios of graph structure $\mathbf{A}$:

1. **Diagonal**: the nodes and their neighbors are in the same group.
2. **Off-Diagonal**: the nodes and their neighbors are in two different groups.

5.6.2 Observations

In Table 5.4, NetInfoF receives the highest average rank among all GNN baselines, and is the only method that can handle all scenarios. While GNNs have worse performance when $\mathbf{X}$ is either random or local, SlimG, a linear GNN, cannot handle cases with off-diagonal $\mathbf{A}$.

5.6.3 Would a GNN work?

Figure 5.4a shows that NetInfoF_Score is highly correlated with test Hits@100, with high R^2 values, where each point denotes a component of embeddings from each split of synthetic datasets. Table 5.5 reports NetInfoF_Score and test performance of each component. By measuring their NetInfoF_Score, NetInfoF_Probe tells when propagating features through structure contains less information than using features or structure itself. For example, in scenarios where node features are useless (the first two scenarios in Table 5.5), NetInfoF_Probe spots that $\mathbf{F}$ (i.e., $g(\mathbf{X})$) provides little NUI to the task, and thus the propagated embeddings

Table 5.5: NetInfoF_Score and test Hits@100 of each derived node embeddings on link prediction. Green (■) marks the winner and red (■) marks the results close to random guessing.

Metric	Feature Component	Random X Diagonal A	Random X Off-Diag. A	Global X Diagonal A	Global X Off-Diag. A	Local X Diagonal A	Local X Off-Diag. A
NetInfoF_Score	$C1:U$	75.1±0.9	74.3±0.5	75.3±0.6	74.1±0.7	75.2±0.8	74.8±0.5
	$C2:R$	76.5±0.9	76.3±0.5	76.7±0.8	76.3±0.2	76.6±0.8	76.3±0.4
	$C3:F$	50.0±0.0	50.0±0.0	67.0±0.5	71.5±0.6	75.9±1.1	75.5±0.6
	$C4:P$	75.1±0.7	73.2±0.6	78.2±0.7	78.5±0.5	78.5±1.2	78.9±0.5
	$C5:S$	74.3±1.1	72.0±0.5	78.7±1.0	79.2±0.5	79.0±0.9	81.1±0.6
Test Hits@100	$C1:U$	83.3±1.2	76.4±2.1	83.5±1.2	76.7±0.5	83.2±1.1	78.2±0.9
	$C2:R$	83.0±1.5	74.7±1.4	83.0±1.3	75.0±1.5	83.2±1.0	75.3±0.7
	$C3:F$	02.2±0.2	02.5±0.2	52.3±1.9	63.0±2.2	80.7±1.1	77.1±1.0
	$C4:P$	82.3±1.2	73.4±0.4	86.2±1.1	79.3±0.7	85.7±1.5	79.4±1.3
	$C5:S$	80.7±0.9	68.1±1.8	86.4±0.9	79.2±1.1	86.5±1.0	85.2±1.1

$\mathbf{P}$ and $\mathbf{S}$ have less NUI than the structural embeddings $\mathbf{U}$ and $\mathbf{R}$. This indicates that training GNNs is less likely to have a better performance than only utilizing the information from the graph structure, which correctly matches the test performance in Table 5.5.

5.7 Experiments

We conduct experiments by real-world graphs to answer the following research questions (RQ):
RQ1. **Effectiveness:** How well does NetInfoF perform in real-world graphs?
RQ2. **Scalability:** Does NetInfoF scales linearly with the input size?
RQ3. **Ablation Study:** Are all the design choices in NetInfoF necessary?
For the experiments of node classification in the real-world datasets, please refer to the paper [LYZ+24]. The experiments are conducted on an AWS EC2 G4dn instance with 192GB RAM.

5.7.1 RQ1 – Effectiveness

Real-World Datasets We evaluate NetInfoF on 7 homophily and 5 heterophily real-world graphs. We randomly split the edges into training, validation, and testing sets by the ratio 70%/10%/20% five times and report the average for fair comparison. Since our goal is to propose a general GNN method, we focus on comparing NetInfoF with 6 GNN baselines, which are general GNNs (GCN, SAGE, GAT), heterophily GNNs (H^2GCN, GPR-GNN), and a linear GNN (SlimG). While Hits@100 is used for evaluating on most graphs, Hits@1000 is used on the larger ones, namely, Products, Twitch, and Pokec, which have much more negative edges in the testing sets.

In Table 5.6, NetInfoF outperforms GNN baselines in 11 out of 12 datasets, and has the highest average rank, as our derived embeddings include comprehensive graph information, that is, structure, features, and features propagated through structure. Compared to non-linear GNNs, SlimG performs worse in most heterophily graphs, showing that it cannot properly measure the node similarity of heterophily embeddings in link prediction. By addressing the

Table 5.6: **NetInfoF wins** on link prediction in most real-world datasets. Hits@100 is reported for most datasets, and Hits@1000 for the large datasets (Products, Twitch, and Pokec). Green (■) marks the winner.

Model	Cora	CiteSeer	PubMed	Comp.	Photo	ArXiv	Products	Cham.	Squirrel	Actor	Twitch	Pokec	Avg. Rank
GCN	67.1±1.8	60.4±10.	47.6±13.	22.5±3.1	39.1±1.6	14.8±0.6	02.2±0.1	82.1±4.5	16.5±1.0	31.1±1.7	16.2±0.3	07.9±1.7	4.1 (1.3)
SAGE	68.4±2.8	55.9±2.5	57.6±1.1	27.5±2.1	40.0±1.9	00.7±0.1	00.3±0.2	84.7±3.6	15.5±1.5	27.6±1.4	08.7±0.6	05.5±0.5	4.5 (1.2)
GAT	66.7±3.6	65.2±2.6	55.1±2.4	28.3±1.6	44.2±3.5	05.0±0.8	O.O.M.	84.8±4.5	15.6±0.8	32.3±2.4	08.2±0.3	O.O.M.	4.0 (1.7)
H^2GCN	64.4±3.4	35.7±5.4	50.5±0.9	17.9±0.7	29.5±2.4	O.O.M.	O.O.M.	79.3±4.5	16.0±2.6	28.7±2.1	O.O.M.	O.O.M.	6.2 (1.0)
GPR-GNN	69.8±1.9	53.5±8.1	66.3±3.3	20.7±1.8	34.1±1.1	13.8±0.8	O.O.M.	77.2±5.6	14.6±2.7	32.1±1.3	12.6±0.2	05.0±0.2	4.6 (1.8)
SlimG	77.9±1.3	86.8±1.0	55.9±2.8	25.3±0.9	40.2±2.5	20.2±1.0	27.6±0.6	76.9±2.8	19.6±1.5	18.7±1.0	12.0±0.3	21.7±0.2	3.5 (1.8)
NetInfoF	81.3±0.6	87.3±1.3	59.7±1.1	31.1±1.9	46.8±2.2	39.2±1.8	35.2±1.1	86.9±2.3	24.2±2.0	36.2±1.2	19.6±0.7	31.3±0.5	1.1 (1.3)

limitations of linear GNNs, NetInfoF is able to consistently outperform both SlimG and non-linear GNNs in both homophily and heterophily graphs. Note that the results in Pokec are similar to the ones in homophily graphs, since it can be labeled as either homophily (by locality) or heterophily (by gender).

OGB Link Prediction Datasets We evaluate NetInfoF on OGB datasets. Table 5.7 shows that NetInfoF outperforms other baselines, while using a model with much fewer parameters. NetInfoF has only 1280 learnable parameters for all datasets, while GCN and SAGE have at least 279K and 424K, respectively, which is 218× more than the ones that NetInfoF has.

5.7.2 RQ2 – Scalability

We plot the number of edges versus the run time of link prediction in seconds on the real-world datasets. In Fig. 5.6, we find that NetInfoF scales linearly with the number of edges, thanks to our speed-up techniques in estimating compatibility matrix $\mathbf{H}^*$. To give a concrete example, the numbers of coefficients of $\mathbf{H}^*_{\mathbf{U}}$ are reduced from $d(d+1)/2 = 8256$ to 3208, 5373, and 4293, for Products, Twitch, and Pokec, respectively. Moreover, those numbers are very reasonable: Products is a homophily graph, its $\mathbf{H}^*_{\mathbf{U}}$ has the fewest coefficients, which are mostly on the diagonal; Twitch is a heterophily graph, its $\mathbf{H}^*_{\mathbf{U}}$ has the most coefficients, which are mostly on the off-diagonal; Pokec can be seen as either homophily or heterophily, its $\mathbf{H}^*_{\mathbf{U}}$ has the number of coefficients in between.

5.7.3 RQ3 – Ablation Study

To demonstrate the necessity of addressing the limitations of linear GNNs in link prediction with our design choices, we study NetInfoF (1) without compatibility matrix (w/o CM), and (2) with only compatibility matrix $\mathbf{H}$ (w/ only $\mathbf{H}$), which is not optimized with negative edges. Table 5.8 shows that NetInfoF works best with both design choices. In heterophily graphs, merely using $\mathbf{H}$ leads to better performance because of properly handling heterophily embeddings; while in homophily graphs, it accidentally increases the similarity between nodes in negative edges and hurts the performance. Taking into account both heterophily embeddings and negative edges, using $\mathbf{H}^*$ as the compatibility matrix has the best performance in both heterophily and homophily graphs.

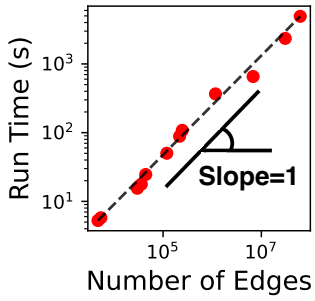


Figure 5.6: **NetInfoF_Act is scalable** on link prediction, being linear with number of edges.

Table 5.7: **NetInfoF wins** on link prediction in OGB datasets. Hits@20, Hits@50, and Hits@100 are reported for ddi, collab, and ppa, respectively. Green (■) marks the winner.

Model	ogbl-ddi	ogbl-collab	ogbl-ppa
GCN	37.1±5.1	44.8±1.1	18.7±1.3
SAGE	53.9±4.7	48.1±0.8	16.6±2.4
SLIMG	35.9±0.6	45.1±0.2	21.1±0.6
NETINFOF	56.8±3.4	53.7±0.2	24.2±0.1

Table 5.8: **Ablation Study: All design choices in NetInfoF are necessary** on link prediction. CM stands for compatibility matrix, and **H** is not optimized with negative edges. Green (■) marks the winner.

Model	Cora	CiteSeer	PubMed	Comp.	Photo	ArXiv	Products	Cham.	Squirrel	Actor	Twitch	Pokec
w/o CM	79.8±0.9	86.5±1.6	58.5±1.2	27.9±0.2	44.7±3.2	35.9±1.8	34.6±0.4	74.6±1.5	14.3±0.6	29.5±1.8	08.9±0.8	30.5±0.3
w/ only **H**	80.9±0.5	87.0±1.4	58.8±1.4	26.5±1.2	43.4±2.1	32.5±1.6	30.6±0.4	74.3±3.2	19.3±1.2	32.2±1.6	10.3±3.1	29.8±0.4
NETINFOF	81.3±0.6	87.3±1.3	59.7±1.1	31.1±1.9	46.8±2.2	39.2±1.8	35.2±1.1	86.9±2.3	24.2±2.0	36.2±1.2	19.6±0.7	31.3±0.5

5.8 Conclusion

We propose the NETINFOF framework to measure and exploit the network usable information (NUI). In summary, NETINFOF has the following advantages:

1. **General**, handling both link prediction and node classification (Lemma 5.1 and 5.2);
2. **Principled**, with theoretical guarantee (Theorem 5.1 and 5.2) and closed-form solution (Lemma 5.1 and 5.2);
3. **Effective**, thanks to the proposed adjustment of node similarity (Figure 5.1);
4. **Scalable**, scaling linearly with the input size (Figure 5.6).

Applied on our carefully designed synthetic datasets, NETINFOF correctly identifies the ground truth of NUI and is the only method that is robust to all graph scenarios. Applied on real-world graphs, NETINFOF wins in *11 out of 12* times on link prediction.

Chapter 6

HybGRAG: Hybrid Retrieval-Augmented Generation on Textual and Relational Knowledge Bases

Chapter based on work that appeared at ACL 2025 [LZM+25] [PDF].

Given a semi-structured knowledge base (SKB), where text documents are interconnected by relations, how can we effectively retrieve relevant information to answer user questions? Retrieval-Augmented Generation (RAG) retrieves documents to assist large language models (LLMs) in question answering; while Graph RAG (GRAG) uses structured knowledge bases as its knowledge source. However, many questions require both textual and relational information from SKB – referred to as "hybrid" questions – which complicates the retrieval process and underscores the need for a hybrid retrieval method that leverages both information.

In this chapter, through our empirical analysis, we identify key insights that show why existing methods may struggle with hybrid question answering (HQA) over SKB. Based on these insights, we propose HybGRAG for HQA, consisting of a retriever bank and a critic module, with the following advantages: (1) *Agentic*, it automatically refines the output by incorporating feedback from the critic module, (2) *Adaptive*, it solves hybrid questions requiring both textual and relational information with the retriever bank, (3) *Interpretable*, it justifies decision making with intuitive refinement path, and (4) *Effective*, it surpasses all baselines on HQA benchmarks.

In experiments on the STaRK benchmark, HybGRAG achieves significant performance gains, with an average relative improvement in Hit@1 of *51%*.

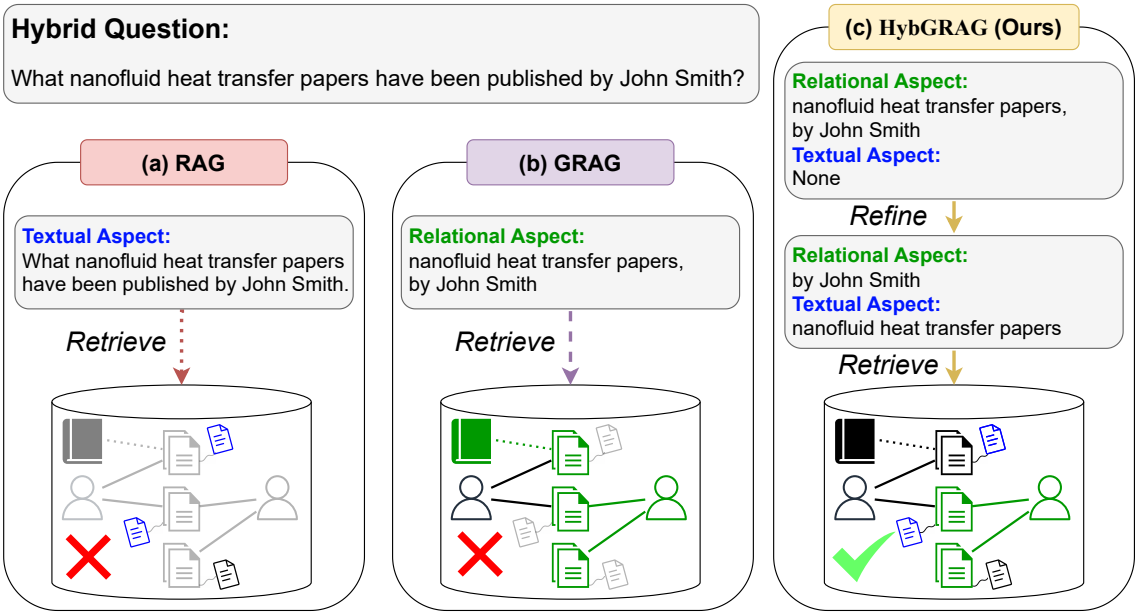


Figure 6.1: **HybGRAG solves hybrid questions in SKB**, which are semi-structured, involving textual and relational aspects. (a) RAG overlooks the interconnections between documents and does not meet the requirements specified by the relational aspect. (b) GRAG relies solely on the relational aspect and misidentifies the textual aspect as part of the relational one. (c) HybGRAG refines the question routing through self-reflection and successfully retrieves the target document in SKB, addressing both textual and relational aspects.

6.1 Introduction

Retrieval-augmented generation (RAG) [LPP+20, GLT+20] enables large language model (LLM) to access information from an unstructured document database. This allows LLMs to address unknown facts and solve Open-Domain Question Answering (ODQA) with additional textual information. Building on this, Graph RAG (GRAG) has extended this concept by retrieving information from structured knowledge bases, where documents are interconnected by relationships. The existing GRAG methods can be categorized into two primary directions. The first focuses on leveraging the capability of LLMs for Knowledge Base Question Answering (KBQA) [YRB+21, SXT+24, JXZ+24, MK24], extracting and using relational information from knowledge graphs (KGs). The second aims to build relationships between documents in the database to improve ODQA performance [LHG+24, DFP+24, ETC+24].

Recently, an emerging problem concentrates on "*hybrid*" question answering (HQA), where a question requires both relational and textual information to be answered correctly, given a semi-structured knowledge base (SKB) [WZY+24]. SKB consists of a structured knowledge base, i.e., knowledge graph (KG), and unstructured text documents, where the text documents are associated with entities of KG. In Figure 6.1 top, an example of hybrid questions is given, which involves both the textual aspect (paper topic) and the relational aspect (paper author), and SKBs are the cylinders.

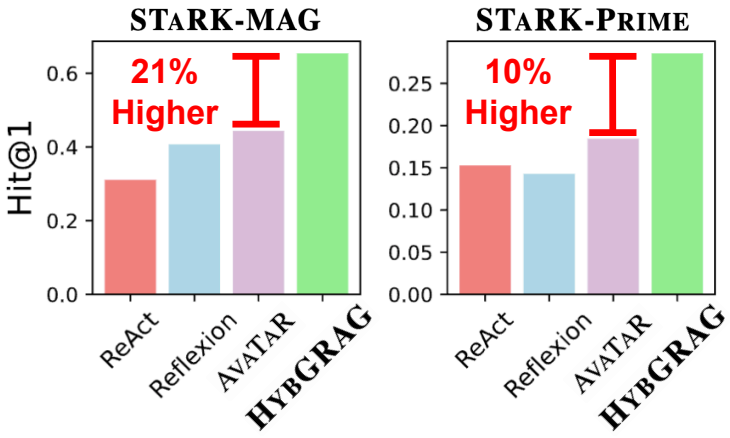


Figure 6.2: **HybGRAG wins** in STARK, outperforming baselines by up to 21% in Hit@1.

Nevertheless, through our empirical analysis, we uncover two critical insights showing that existing methods that perform RAG or GRAG cannot effectively tackle HQA, which requires a synergy between the two retrieval methods. First, they focus solely on retrieving either textual or relational information. As shown in Figure 6.1(a) and (b), this limitation reduces their applicability when the synergy between the two modalities is required. Second, in hybrid questions, the aspects required to retrieve different types of information may not be easily distinguishable. In Figure 6.1(c), question routing [LLZ+24] is performed to identify the aspects of the question. However, in an unsuccessful routing, confusion between the textual aspect "nanofluid heat transfer papers" and the relational aspect "by John Smith", leads to incorrect retrieval.

To solve HQA in SKB, we propose HybGRAG. HybGRAG handles hybrid questions with a retriever bank, which leverages both textual and relational information simultaneously. To improve the accuracy of the retrieval, HybGRAG performs self-reflection [RG24], which iteratively improves its question routing based on feedback from a carefully designed critic module, as shown in Figure 6.1(c). Similarly to chain-of-thought (CoT) [WWS+22], which is widely regarded as interpretable, HybGRAG's refinement path provides intuitive explanations for the performance improvement. Last but not least, the framework of HybGRAG is designed to be flexible, and can easily be adapted to different problems. We summarize the contributions of HybGRAG as follows:

1. **Agentic**: it automatically refines the question routing with self-reflection;
2. **Adaptive**: it solves textual, relational and hybrid questions with a unified framework;
3. **Interpretable**: it justifies the decision making with intuitive refinement path; and
4. **Effective**: it outperforms all baselines on real-world HQA benchmarks.

In Table 6.1, HybGRAG is the only work that satisfies all the desired properties and solves HQA. In Figure 6.2, evaluating on a HQA benchmark STARK, HybGRAG outperforms the second-best baseline significantly with relative improvements in Hit@1 *47%* in STARK-MAG, and *55%* in STARK-Prime, respectively.

Table 6.1: **HybGRAG matches all properties**, while baselines miss more than one property.

Property		Regular RAG	Think-on-Graph	AvaTaR	**HybGRAG**
1. Agentic				✔	✔
2. Adaptive	2.1. Questions in ODQA	✔		✔	✔
	2.2. Questions in KBQA		✔		✔
	2.3. Questions in HQA				✔
3. Interpretable		?	✔	✔	✔

Table 6.2: Table of Symbols and Definitions.

Symbol	Definition
G	Knowledge graph
$\mathcal{N}$	Set of entities in KG
$\mathcal{R}$	Set of relations in KG
$\mathcal{T}_{\mathcal{N}}$	Set of entity types
$\mathcal{T}_{\mathcal{R}}$	Set of relation types
$\mathcal{D}$	Set of text documents
q	Question
$\mathcal{X}$	Set of retrieved documents

Table 6.3: Table of Acronyms.

Acronym	Definition
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
GRAG	Graph Retrieval-Augmented Generation
SKB	Semi-Structured Knowledge Base
KG	Knowledge Graph
HQA	Hybrid Question Answering
ODQA	Open-Domain Question Answering
KBQA	Knowledge Base Question Answering
VSS	Vector Similarity Search
PPR	Personalized PageRank
ICL	In-Context Learning

6.2 Proposed Insights: Challenges in HQA

What new challenges in HQA over SKB remain unsolved by existing methods? In this section, we first define the problem HQA, and then conduct experiments to uncover two critical insights, laying the foundation for designing our method for HQA.

6.2.1 Problem Definition

An overview of symbols and acronyms is provided in Table 6.2 and Table 6.3, respectively. A semi-structured knowledge base (SKB) consists of a KG $G = (\mathcal{N}, \mathcal{R})$, where $\mathcal{N}$ and $\mathcal{R}$ represent the sets of entities and relations. It also includes a set of text documents $\mathcal{D} = \bigcup_{i \in \mathcal{N}} D_i$, where D_i is the document associated with entity i. Entity and relation types are denoted by $\mathcal{T}_{\mathcal{N}}$ and $\mathcal{T}_{\mathcal{R}}$, respectively. Each hybrid question q in SKB involves semi-structured information, namely, textual and relational information. We define hybrid question answering (HQA) as follows:

Table 6.4: Textual and relational information are both useful to answer hybrid question in STARK-MAG.

Method	Hit@1	Hit@5
Text Retriever: VSS	0.2908	0.4961
Graph Retriever: PPR	0.2533	0.5523
Optimal Routing	0.4522	0.7463

Table 6.5: LLMs frequently extracts a subgraph from KG in SKB without target entities in STARK-MAG.

# of Iterations	Feedback Type	Hit Rate
1	N/A	0.6769
2	Simple	0.7914
2	Corrective	0.9231

- **Given** a SKB consisting of a KG $G = (\mathcal{N}, \mathcal{R})$ and a set of documents $\mathcal{D}$, and a hybrid question q.
- **Retrieve** a set of documents $\mathcal{X} \subseteq \mathcal{N}$, where each document satisfies the requirements specified by the relational and textual aspects of q.

6.2.2 Insight 1: Hybrid-Sourcing Question

To investigate if it is necessary to leverage both textual and relational information to answer hybrid questions, we conduct an experiment to show that text documents and KG contain useful but non-overlapping information. As a retriever that uses only textual information, vector similarity search (VSS) [KOM+20] performs retrieval and ranking by comparing the question and documents in the embedding space ("ada-002"); as a retriever that uses only relational information, Personalized PageRank (PPR) [ACL06] performs random walks from the topic entities identified by an LLM (Claude 3 Sonnet) and ranks neighboring entities based on their connectivity in KG.

In Table 6.4, the text and graph retrievers have competitive performance. Interestingly, if an optimal routing always picks the retriever that gives the correct result, the performance is significantly higher, indicating little overlap between the strengths of the text and graph retrievers. This highlights the importance of a solution to leverage both textual and relational information simultaneously by synergizing these two retrievers. In Figure 6.1(c), we show a hybrid question that requires both textual and relational information to be answered. Based on this observation, we uncover the first challenge:

Challenge 6.1: Hybrid-Sourcing Question

In HQA, there are questions that require both textual and relational information to be answered.

6.2.3 Insight 2: Refinement-Required Question

The success of KBQA often relies on the assumption that the target entities are within an extracted subgraph from KG [LHJ+23]. Similarly, answering a question in HQA requires extracting a subgraph from KG in SKB. As hybrid questions involve both textual and relational aspects,

they can be challenging for an LLM to comprehend. To study this, we test if an LLM can extract a subgraph from KG that contains the target entities (hit). More specifically, an LLM (Claude 3 Sonnet) is prompted to identify the relational aspect in the question, i.e. topic entities and useful relations used to extract the subgraph. An oracle is used to instruct LLM to perform an extra iteration with feedback if the target entities are not included in the subgraph.

In Table 6.5, if the result is incorrect, simply prompting LLM to redo the extraction gives a much better hit ratio. Moreover, if the LLM receives feedback that points out the erroneous part of the extraction (e.g., extracted topic entity is wrong), it significantly improves the result. This is because in hybrid questions that contain both textual and relational aspects, LLM can falsely identify the textual aspect as the relational one. In Figure 6.1(c), there is an error in retrieving the correct reference from LLM, as it confuses the textual aspect as an entity of type "field of study" on the first attempt. Based on this observation, we uncover the second challenge:

Challenge 6.2: Refinement-Required Question

In HQA, LLM struggles to distinguish between the textual and relational aspects of the question on the first attempt, necessitating further refinements.

6.3 Proposed Method: HybGRAG

To solve HQA, we propose HybGRAG, consisting of the *retriever bank* and the *critic module*, to address Challenge 6.1 and Challenge 6.2, respectively.

6.3.1 Retriever Bank

To solve Challenge 6.1, we propose the retriever bank, composed of multiple retrieval modules and a router. Given a question q, the router determines the selection and usage of the retrieval module, a process known as question routing. The selected retrieval module then retrieves the top-K references $\mathcal{X}$, as elaborated in the next paragraph.

Retrieval Modules We design two retrieval modules, namely text and hybrid retrieval modules, to retrieve information from text documents and SKB, respectively. Each retrieval module includes a retriever and a ranker, which offers the flexibility to cover a wide range of questions.

The text retrieval module retrieves documents using similarity search for a question q, such as dense retrieval, which is designed to directly find answers within text documents. We use VSS between question q and documents $\mathcal{D}$ in the embedding space as both the retriever and the ranker. This is typically used when nothing can be extracted from the hybrid retrieval module.

The hybrid retrieval module takes the identified topic entities $\hat{\mathcal{N}}$ and useful relations $\hat{\mathcal{R}}$ as input. It uses a graph retriever to extract entities in the ego-graph of $\hat{\mathcal{N}}$, connected by $\hat{\mathcal{R}}$. For example, in Figure 6.1, $\{\hat{\mathcal{N}} = \{\text{John Smith}\}, \hat{\mathcal{R}} = \{\text{author writes paper}\}\}$ and the graph retriever extracts the entities/papers connected by the path "John Smith -> author writes paper -> {paper(s)}". If more than one ego-graph is extracted, their intersection is used as the final

Table 6.6: **Corrective feedback** of the critic module in HybGRAG for STaRK.

Error Source	Error Type	Feedback
Identification	Incorrect Entity/Relation	Entity/relation {name} is incorrect. Please remove or substitute this entity/relation.
	Missing Entity	There is only one entity but there may be more. Please extract one more entity and relation.
	No Entity	There is no entity extracted. Please extract at least one entity and one relation.
	No Intersection	There is no intersection between the entities. Please remove or substitute one entity and relation.
	Incorrect Intersection	There is an intersection between the entities, but the answer is not within it. Please remove or substitute one entity and relation.
Selection	Incorrect Retrieval Module	The retrieved document is incorrect. The current retrieval module may not be helpful to narrow down the search space.

result. Finally, to solve hybrid questions, we propose ranking the documents associated with the extracted entities using a VSS ranker between question q and documents $\mathcal{D}$. This ensures the synergy between the relational and textual information.

Router Given a question q, the LLM router performs question routing to determine the selection and usage of the retrieval module. More specifically, the router first identifies the relational aspect, i.e., topic entities $\hat{\mathcal{N}}$ and useful relations $\hat{\mathcal{R}}$ based on the types of entities $\mathcal{T}_{\mathcal{N}}$ and the types of relation $\mathcal{T}_{\mathcal{R}}$ using few shot examples [BMR+20]. The router then makes the selection s_t, determining whether to use a text or a hybrid retrieval module. Identifying $\hat{\mathcal{N}}$ and $\hat{\mathcal{R}}$ before determining s_t improves the quality of s_t. For example, if there is no entity extracted $\hat{\mathcal{N}} = \emptyset$, a text retrieval module is a better option.

6.3.2 Critic Module

Given a hybrid question q, the router is asked to perform question routing, including identifying topic entities $\hat{\mathcal{N}}$ and useful relations $\hat{\mathcal{R}}$. However, as mentioned in Challenge 6.2, they may be incorrectly identified in the first iteration.

To solve this, we propose the critic module, which provides feedback to help the router perform better question routing. Instead of using a single LLM to complete this complicated task, we divide the critic into two parts, an LLM validator C_{val} to validate the correctness of the retrieval $\mathcal{X}$, and an LLM commenter C_{com} to provide feedback f_t if the retrieval is incorrect. This divide-and-conquer step, similar to previous works [GDP+23, AWW+24], is crucial to our critic module, offering two key advantages: (1) By breaking a difficult task into two easier ones, we can now leverage pre-trained LLMs to solve them while maintaining good performance. This avoids issues arising from fine-tuning LLMs, such as the unavailability of labels or catastrophic forgetting [TSdC+19]. (2) Since the tasks of C_{val} and C_{com} are independent, they can each have

Algorithm 6.1: HybGRAG

Input: Question q, a SKB with G and $\mathcal{D}$, Entity Types $\mathcal{T}_{\mathcal{N}}$, Relation Types $\mathcal{T}_{\mathcal{R}}$, and Maximum Iterations T

$f_1 =$ " ";
for $t = 1, \ldots, T$ **do**
  /* Retriever Bank */
  $s_t, \hat{\mathcal{N}}_t, \hat{\mathcal{R}}_t = Router(q, \mathcal{T}_{\mathcal{N}}, \mathcal{T}_{\mathcal{R}}, f_t)$;
  if s_t *is hybrid retrieval module* **then**
    $\mathcal{X}_t = HybridRM(q, G, \mathcal{D}, \hat{\mathcal{N}}_t, \hat{\mathcal{R}}_t)$;
  else
    $\mathcal{X}_t = TextRM(q, \mathcal{D})$;
  end
  /* Validator */
  if $C_{val}(q, \mathcal{X}_t) = True$ **then**
    Return $\mathcal{X}_t$;
  else
    /* Commenter */
    $f_{t+1} = C_{com}(q, s_t, \hat{\mathcal{N}}_t, \hat{\mathcal{R}}_t)$;
  end
end
Return $\mathcal{X}_t$;

their own exclusive contexts, preventing the inclusion of irrelevant information and avoiding the "lost in the middle" phenomenon [SCM+23, LLH+24].

Validator The LLM validator C_{val} aims to validate if the top references retrieved $\mathcal{X}$ meet the requirements specified by the question q, which is a binary classification task. To improve accuracy, we provide an additional validation context for the validator. We use the reasoning paths between topic entities and entities in the extracted ego-graph as the validation context, which are used to verify whether the output satisfies certain requirements in the question. The reasoning paths are verbalized as "{topic entity} → {useful relation} →... → {useful relation} → {neighboring entity}". For example, if a hybrid question asks for a paper (i.e. a document) from a specific author, then the context including the reasoning paths "{author} → {writes} → {paper}" is essential for verification.

Commenter The LLM commenter C_{com} aims to provide feedback f to help the router refine question routing. To effectively guide the router, we construct *corrective* feedback that it can easily understand. In more detail, it points out the error(s) in each action, such as incorrect identification of topic entities, as shown in Table 6.6. Unlike natural language feedback, which may cause uncertainty or inconsistency depending on the LLM used, our corrective feedback provides clear guidance on how to refine the question routing. Furthermore, it leverages in-context learning (ICL) to provide sophisticated feedback. We collect a small number of success-

ful experiences (≈ 30) in the training set as examples, with each experience $\{s_t, \hat{\mathcal{N}}_t, \hat{\mathcal{R}}_t, f_{t+1}\}$ comprising a pair of router action and feedback, which is verified by ground truth. During inference, the commenter gives high-quality feedback based on multiple pre-collected examples.

6.3.3 Overall Algorithm

The algorithm of HybGRAG is in Algorithm 6.1. Given a question q, in iteration t, the router determines s_t, $\hat{\mathcal{N}}_t$ and $\hat{\mathcal{R}}_t$ to retrieve the references $\mathcal{X}_t$ from both G and $\mathcal{D}$ in SKB, or only $\mathcal{D}$, with the selected retrieval module. The validator C_{val} in the critic module then decides whether to accept $\mathcal{X}_t$ as the final answer or reject it. If $\mathcal{X}_t$ is rejected, the commenter C_{com} generates feedback f_{t+1} for the router to assist in refining its action in iteration $t+1$.

6.4 Experiments

We conduct experiments to answer the following research questions (RQ):
RQ1. **Effectiveness:** How well does HybGRAG perform in the real-world HQA benchmark?
RQ2. **Ablation Study:** Are all the design choices in HybGRAG necessary?
RQ3. **Interpretability:** How does HybGRAG refine its question routing based on feedback?
RQ4. **Adaptive:** How well does HybGRAG perform in an end-to-end RAG setting?
RQ5. **Cost Analysis:** How much does HybGRAG cost to run?

Benchmarks We conduct experiments on two QA benchmarks: STaRK [WZY+24], which serves as the primary evaluation benchmark and focuses on retrieval, and CRAG [YSX+24], a complementary benchmark to evaluate end-to-end RAG performance. While STaRK focuses on HQA, CRAG encompasses both ODQA and KBQA as sub-problems.

6.4.1 RQ1 – Effectiveness

We use the default evaluation metrics provided by STaRK, which are Hit@1, Hit@5, Recall@20 and mean reciprocal rank (MRR), to evaluate the performance of the retrieval task. We compare HybGRAG with various baselines, including recent methods (QAGNN [YRB+21] and Think-on-Graph [SXT+24]); traditional RAG approaches; and self-reflective LLMs (ReAct [YZY+23], Reflexion [SCG+23], and AvaTaR [WZH+24]).

In Table 6.7, HybGRAG outperforms all baselines significantly in both datasets in STaRK. Most baselines are designed to handle ODQA and KBQA, and the results have shown that they cannot handle HQA effectively (Challenge 6.1). Our hybrid retrieval module is the second-best performing method, highlighting the importance of designing a synergized retrieval module that uses both textual and relational information simultaneously. In addition, HybGRAG performs significantly better than the hybrid retrieval module, indicating that the extracted entity and relation are frequently incorrect in the first iteration (Challenge 6.2). By tackling Challenge 6.1 and 6.2 with our retriever bank and critic module respectively, HybGRAG has a significant improvement in performance.

Table 6.7: **Retrieval Evaluation on STaRK: HybGRAG wins.** '*' denotes that only 10% of the testing questions are evaluated due to the high latency and cost of the methods. Green (■, ■) marks the top two.

Method	STaRK-MAG				STaRK-Prime			
	Hit@1	Hit@5	Recall@20	MRR	Hit@1	Hit@5	Recall@20	MRR
QAGNN	0.1288	0.3901	0.4697	0.2912	0.0885	0.2123	0.2963	0.1473
Think-on-Graph*	0.1316	0.1617	0.1130	0.1418	0.0607	0.1571	0.1307	0.1017
Dense Retriever	0.1051	0.3523	0.4211	0.2134	0.0446	0.2185	0.3013	0.1238
VSS (Text Retrieval Module)	0.2908	0.4961	0.4836	0.3862	0.1263	0.3149	0.3600	0.2141
Multi-VSS	0.2592	0.5043	0.5080	0.3694	0.1510	0.3356	0.3805	0.2349
VSS w/ LLM Reranker*	0.3654	0.5317	0.4836	0.4415	0.1779	0.3690	0.3557	0.2627
ReAct	0.3107	0.4949	0.4703	0.3925	0.1528	0.3195	0.3363	0.2276
Reflexion	0.4071	0.5444	0.4955	0.4706	0.1428	0.3499	0.3852	0.2482
AvaTaR	0.4436	0.5966	0.5063	0.5115	0.1844	0.3673	0.3931	0.2673
Hybrid Retrieval Module (Ours)	0.5028	0.5820	0.5031	0.5373	0.2492	0.3274	0.3366	0.2842
HybGRAG (Ours)	0.6540	0.7531	0.6570	0.6980	0.2856	0.4138	0.4358	0.3449
Relative Improvement	47.4%	26.2%	29.3%	36.5%	54.9%	12.7%	10.9%	29.0%

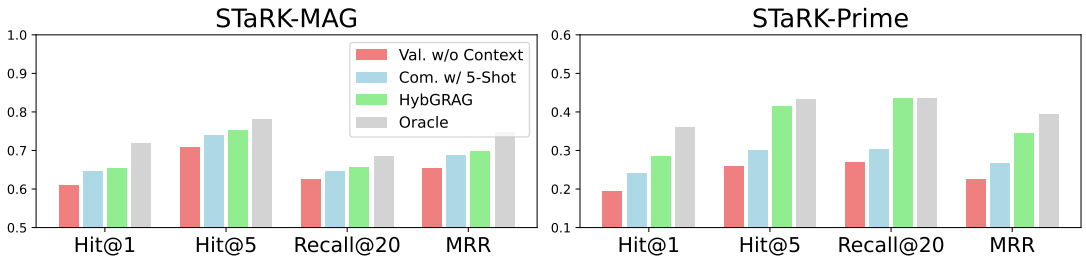


Figure 6.3: **Design choices in HybGRAG are necessary** in STaRK. We compare HybGRAG with two variants: a validator without validation context, and a commenter with only 5-shot. Oracle uses ground truth during inference.

6.4.2 RQ2 – Ablation Study

Critic Module We compare HybGRAG variants with a validator without validation context, a commenter with only five shots, and those with oracles. The oracle has access to the ground truth, which gives the optimal judgement on the correctness of the output and the error type of question routing, if there is any. In Figure 6.3, we show that HybGRAG performs the best with all our design choices, approaching the performance of an oracle.

Self-Reflection In Figure 6.4, we demonstrate that with more self-reflection iterations, the performance of HybGRAG improves further. Performance improves significantly when increasing the number of iterations from 1 to 2, where no self-reflection is performed in iteration 1. It is also shown that a few iterations are sufficient, as the improvement diminishes over iterations.

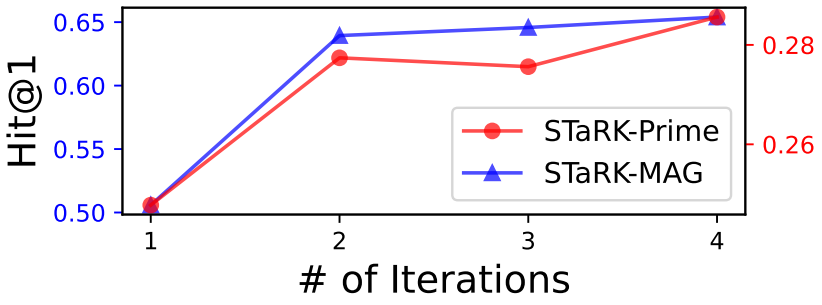


Figure 6.4: **HybGRAG improves its question routing** thanks to the critic module.

Table 6.8: **HybGRAG maintains strong performance** with a less powerful LLM model in STaRK-MAG. Green (■) marks the winner.

Base Model	Hit@1	Hit@5	Recall@20	MRR	Speedup
Claude 3 Haiku	0.6019	0.7084	0.6067	0.6483	1.96×
Claude 3 Sonnet	0.6540	0.7531	0.6570	0.6980	1.00×

Model Size Although we do not have access to Claude 3 Opus, we conduct experiments with Claude 3 Haiku, a more cost-efficient but less powerful alternative to Claude 3 Sonnet*. In Table 6.8, HybGRAG maintains strong performance even with Claude 3 Haiku. The results also follow the scaling law of LLMs [KMH+20].

Multi-Agent Perspective Since HybGRAG can be interpreted as a multi-agent system, we add a single-agent baseline, which relies on the router to make decisions and provide feedback for self-reflection. In Table 6.9, HybGRAG outperforms both single-agent and no-agent baselines. This highlights that self-reflection is essential for achieving strong performance in HQA, as pointed out in Challenge 6.2. Moreover, unlike the plain text feedback generated by the single-agent baseline, the feedback generated by HybGRAG more effectively guides the router in refining its decision, thanks to our carefully designed critic module.

6.4.3 RQ3 – Interpretability

Figure 6.5 illustrates examples of the interaction between the router in the retriever bank and the critic module in STaRK-MAG. In the first iteration of Figure 6.5(a), the router misidentifies a "optical TALU implementations in electronic circuits" as a topic entity representing the field of study (relational aspect). Since the ego-graph extracted based on this entity has no intersection with the ego-graph extracted based on "Netaji Subhash Engineering College", the critic module recognizes that the former entity has a higher chance of being a textual aspect. Thus, it gives the feedback to the router, and the router addresses it accordingly. This refinement path of HybGRAG is similar to CoT, making it interpretable and easy for the user to understand.

*https://www.anthropic.com/news/claude-3-family

Table 6.9: **HybGRAG performs best with multi-agent design** in STaRK-MAG. "Router for SR" baseline performs self-reflection using a single LLM router. Green (■) marks the winner.

Method	Setting	Hit@1	Hit@5	Recall@20	MRR
Hybrid RM	No-Agent	0.5028	0.5820	0.5031	0.5373
Router for SR	Single-Agent	0.6206	0.7069	0.6187	0.6587
HybGRAG	Multi-Agent	0.6540	0.7531	0.6570	0.6980

(a) Error Type: Incorrect Entity/Relation

Q: Any 2012 publications from Netaji Subhash Engineering College on optical TALU implementations in electronic circuits?

Action 1:
Selection: Hybrid retrieval module
Entity: Netaji Subhash Engineering College (institution), optical TALU implementations in electronic circuits (field of study)
Relation: author affiliated with institution, author writes paper, paper has topic field of study

Feedback 1: The retrieved document is incorrect.
Entity "optical TALU implementations in electronic circuits" and relation "paper has topic field of study" are incorrect. Please remove or substitute one entity and relation.

Action 2:
Selection: Hybrid retrieval module
Entity: Netaji Subhash Engineering College (institution)
Relational: author affiliated with institution, author writes paper

Feedback 2: Accept. ✅

(b) Error Type: Missing Entity

Q: Are there any 2016 publications by co-authors of "A Low Abundance of 135Cs in the Early Solar System from Barium Isotopic Signatures" which discuss the comparison of Earth's chemical composition with that of chondrites?

Action 1:
Selection: Hybrid retrieval module
Entity: A Low Abundance of 135Cs in the Early Solar System from Barium Isotopic Signatures (paper)
Relation: author writes paper

Feedback 1: The retrieved document is incorrect.
There is only one entity but there may be more. Please extract one more entity and relation.

Action 2:
Selection: Hybrid retrieval module
Entity: A Low Abundance of 135Cs in the Early Solar System from Barium Isotopic Signatures (paper), chondrites (field of study)
Relational: author writes paper, paper has topic field of study

Feedback 2: Accept. ✅

Figure 6.5: **HybGRAG is interpretable.** In examples from STaRK-MAG, HybGRAG successfully refines its entity and relation extraction based on corrective feedback from critic module.

6.4.4 RQ4 – End-to-End RAG Evaluation

We modify HybGRAG to adapt to CRAG. We use default evaluation metrics, where an LLM evaluator is used to determine if the predicted answers are accurate, incorrect (hallucination), or missing, and Score_a with 1, -1, and 0 for these respective categories. We compare HybGRAG with CoT LLM, text-only RAG, graph-only RAG, and RAG that concatenates text and graph references. We include two self-reflective LLMs (ReAct, Corrective RAG) that share the same retriever bank but use different critics.

In Table 6.10, HybGRAG outperforms all baselines in CRAG. RAGs with a single retrieval module cannot handle both types of questions. RAG with a concatenated reference also distracts by irrelevant content in the long reference. Although the same retriever bank is provided, self-reflective baselines still find it difficult to refine their action. Since ReAct relies on the LLM's ability to think and provide natural language feedback, it often lacks clear guidance for improving its actions. Without a fine-tuned retrieval evaluator, Corrective RAG cannot effectively identify the usefulness of a reference.

Table 6.10: **End-to-End RAG Evaluation on CRAG: HybGRAG wins.** All baselines (except CoT LLM) share our retriever bank, but use different critics to provide feedback. Green (■, ■) marks the top two.

Method	Llama 3.1 70B				Claude 3 Sonnet			
	Accuracy ↑	Halluc. ↓	Missing	Score_a ↑	Accuracy ↑	Halluc. ↓	Missing	Score_a ↑
CoT LLM	0.4607	0.5026	0.0367	-0.0419	0.3910	0.4052	0.2038	-0.0142
Text-Only RAG	0.4105	0.3685	0.2210	0.0420	0.5034	0.3955	0.1011	0.1079
Graph-Only RAG	0.4861	0.4442	0.0697	0.0419	0.5303	0.2974	0.1723	0.2329
Text & Graph RAG	0.4120	0.3790	0.2090	0.0330	0.5820	0.3416	0.0764	0.2404
ReAct	0.1745	0.2360	0.5895	-0.0615	0.4352	0.4075	0.1573	0.0277
Corrective RAG	0.4509	0.4652	0.0839	-0.0143	0.4674	0.3333	0.1993	0.1341
HybGRAG (Ours)	0.5206	0.3588	0.1206	0.1618	0.6322	0.2959	0.0719	0.3363

Table 6.11: Number of API calls and tokens used by HybGRAG for STaRK.

HybGRAG Component	API Call # per Iteration	Token # for for Prompts	Token # for Examples in MAG	Token # for Examples in Prime
Router	2	159	2709	3018
Validator	1	39	1383	2107
Commenter	1	52	1215	1583

6.4.5 RQ5 – Model Cost Analysis

We report the number of API calls and token consumption (excluding references) for each step in an iteration of HybGRAG in Table 6.11 for STaRK. While most token consumption arises from the examples used for ICL, the prompts themselves require very few tokens. Moreover, since HybGRAG uses chat LLM as the router, the examples for ICL only need to be given once. Compared to the state-of-the-art baseline AvaTaR, which requires at least 500 API calls during training, our hybrid retrieval module achieves a relative improvement 24% in Hit@1 with only 2 API calls, while HybGRAG achieves 51% with at most 14 API calls, both without training.

6.5 Related Work

6.5.1 Graph Retrieval-Augmented Generation

Various settings have been explored for GRAG [PZL+24], and can be roughly divided into three directions. The first focuses on Knowledge Base Question Answering (KBQA), taking advantage of the LLM capability [YRB+21, SXT+24, JXZ+24, MK24]. The second focuses on Open-Domain Question Answering (ODQA), building relationships between documents to improve retrieval [LHG+24, DFP+24, ETC+24]. The last assumes that a subgraph is given when answering a question [HTS+24, HLZ+24]. In contrast, we focus on solving Hybrid Question Answering (HQA) in SKB, and previous GRAG methods are not easily generalized to HQA.

6.5.2 Agentic and Self-Reflective LLMs

LLM agents [YZY+23, WZH+24] facilitate planning in complex reasoning tasks. Among them, AvaTaR is the most recent, proposing iterative prompt optimization via contrastive reasoning. However, they may still struggle to generate the correct output on the first attempt. Self-reflection addresses this limitation by iteratively optimizing the output based on feedback, typically provided by a critic implemented using various approaches: pre-trained LLMs [SCG+23, MTG+23], external tools [GSG+24, QZF+24], or fine-tuned LLMs [PIP+24, AWW+24, YGZL24]. Nevertheless, they do not generalize to HQA for two reasons. First, they lack appropriate retrieval tools and guidance on how to refine retrieval effectively. Second, in the absence of external tools or labels for fine-tuning, using pre-trained LLMs as critics without careful design results in suboptimal self-evaluation and overly implicit feedback.

6.6 Conclusion

To solve hybrid question answering (HQA), we propose HybGRAG, driven by insights from our empirical analysis, which has following advantages:

1. **Agentic**: it refines question routing with self-reflection by our critic module;
2. **Adaptive**: it solves textual, relational and hybrid questions by our retriever bank;
3. **Interpretable**: it justifies the decision making with intuitive refinement path; and
4. **Effective**: it significantly outperforms all the baselines on HQA benchmarks.

Applied on STaRK, HybGRAG achieves an average relative improvement *51%* in Hit@1.

Part II

Graph-Level Graph Mining

Overview

Given a graph database, how can we find substructures frequently shared by graphs?
How can we leverage these substructures to solve the given graph-level tasks?

In a graph database, graphs share substructure patterns, some of which occur more frequently than others. These frequent substructures provide clear explanations and useful information for identifying graph properties or detecting anomalies. In this chapter, we aim to detect frequent substructures and leverage them to solve downstream graph-level tasks, such as graph anomaly detection and graph regression.

We address this problem in two different settings of increasing difficulty and propose two corresponding **algorithms**:

- § 7: Graph anomaly detection – GAWD identifies frequent substructures that best compress the graphs and assigns higher anomaly scores to those that are less compressible.
- § 8: Various graph-level tasks – RWK$^+$ is an improved random walk graph kernel used to learn frequent substructures and extract features across graphs for downstream tasks.

We further extend this problem to a real-world graph **application**:

- § 9: In human trafficking detection, DeltaShield represents millions of escort advertisements as graphs and identifies common templates shared among them.

Chapter 7

GAWD: Graph Anomaly Detection in Weighted Graph Databases

Chapter based on work that appeared at ASONAM 2021 [LNB+21] [PDF].

Given a set of node-labeled directed weighted graphs, how to find the most anomalous ones? How can we summarize the normal behavior in the database without losing information? The idea of the existing work is to (1) iteratively identify the "best" substructure (i.e., subgraph or motif) that yields the largest compression when each of its occurrences is replaced by a super-node, and (2) score each graph by how much it compresses over iterations – the more the compression, the lower the anomaly score.

In this chapter, we propose GAWD, for detecting anomalous graphs in directed weighted graph databases. Different from existing work, our GAWD exhibits: (i) a *lossless* graph encoding scheme, (ii) ability to handle numeric edge weights, (iii) interpretability by common patterns, and (iv) scalability with running time linear in input size.

Experiments on four datasets injected with anomalies show that GAWD achieves significantly better results than state-of-the-art baselines.

Figure 7.1: **GAWD wins on both effectiveness and scalability:** We evaluate four datasets and show the big gap between it and competitors w.r.t. average precision and run time.

7.1 Introduction

Given a large graph database containing directed weighted node-labeled graphs, how can we detect the anomalous graphs? Many studies succeed in detecting anomalies but fail to give satisfying interpretations. This raises another prominent problem — how can we spot anomalies and summarize the normal behavior without simultaneously losing information?

In recent years, graph [HYL17, NCV+17] and node embedding [GL16] have attracted a lot of attention. These methods have been used in anomaly detection in conjunction with off-the-shelf anomaly detectors. Embedding-based models, however, lack interpretability. In contrast, structure-based methods enable domain experts to conduct post-analysis to reveal root causes of anomalies. Several structure-based methods [NC03, EH07] detect anomalies by compressing graphs with a substructure that yields the largest compression. The selected substructure is replaced by a super-node and the process continues in iterations. As a result, graphs with more common substructures (and hence compress more) are deemed less anomalous than those with fewer substructures.

However, neither embedding- nor structure-based methods are perfect: (1) both of these methods cannot totally avoid information loss, which causes difficulty in interpreting results, and (2) none of the structure-based methods can handle weighted graphs, which prevents them from detecting anomalies caused by edge weights.

We propose GAWD to address the aforementioned problems, with following advantages:

- **Lossless Encoding:** GAWD builds on Noble and Cook [NC03] in terms of identifying frequent subgraphs and compressing the graphs in the input database by replacing each of its occurrences by a super-node. This results in a loss of connectivity information for nodes outside the substructure that are connected to nodes within. We address this issue by incorporating "rewiring" information into our encoding, such that the compressed graph can be reconstructed into the original graph *losslessly*.
- **Handling Weighted Graphs:** We propose a novel encoding scheme for handling numeric edge weights. Given a substructure we estimate a "representative" weight for its edges, as well as extend the encoding of a compressed graph to incorporate corrections for true weights such that decompression can be done losslessly.
- **Interpretability and Scalability:** The (lack of) frequent subgraphs common in the database provide a means to explain anomalousness. Moreover, GAWD exhibits linear scalability in the input size.

As shown in Figure 7.1, experimental results on four real-world datasets with injected anomalies show that GAWD provides better trade-off between detection performance and running time compared to both existing graph embedding- and structure-based methods. Moreover, GAWD is lossless in contrast with lossy compression of existing methods [CH94, NC03, EH07] and, therefore, able to backtrack the process after compression, while other lossy methods lose those information after compression.

Reproducibility: Our code and injected datasets along with labels (except for Accounting Dataset due to privacy issues) are made publicly available at `https://github.com/mengchillee/GAWD`.

7.2 Related Work

Being one of the closest real-world applications, anomaly detection has drawn a lot of attention from academics [MMA16, BJR17, SDS17, CYL+18, LZW+20]. Many anomaly detectors have been developed, such as LOF [BKNS00], Isolation Forest [LTZ08] and LODA [Pev16]. Some traditional machine learning methods such as kNN [RRS00] and PCA [SCSC03] can be used as an anomaly detector as well. To this reason, a useful toolkit for this field is also developed [ZNL19]. These detectors can be easily used with feature-based or embedding-based methods to achieve effective results. Our work introduces an anomaly detection for graphs approach, while using popular graph embedding methods (with anomaly detectors) as baselines. We will discuss them both in this section.

Nevertheless, none of the above methods fulfills all the specs of GAWD. Table 7.1 contrasts GAWD against the existing state-of-the-art.

7.2.1 Indirect Anomaly Detection via Graph Embedding

Graph embedding has been widely studied in the last decade. One reason for its popularity is its flexibility concerning downstream applications. Graph embedding can be used to de-

Table 7.1: **GAWD matches all specs**, while competitors miss one or more of the desired properties. * implies the method only accepts (categorical) edge labels.

Property \ Method	node2vec [GL16]	graph2vec [NCV+17]	Noble *et al.* [NC03]	GBAD [EH07]	OddBall [AMF10]	Yagada [DLMR11]	**GAWD**
Handle Graph Database	✔	✔	✔	✔		✔	✔
Handle Node Labels		✔	✔	✔	✔	✔	✔
Handle Edge Weights	✔				✔	✔*	✔
Lossless							✔
Scale Linearly	✔	✔			✔		✔

tect anomalous graphs in conjunction with off-the-shelf anomaly detectors. One of the most famous branches is node embedding, where each node in the graph is mapped to low dimensional space. node2vec [GL16] could highly identify graph structures with biased random walks using BFS and DFS. GraphSAGE [HYL17] increases the generalizability to the unseen nodes by training the aggregator function by node features. A simple way to extend node embedding to graph embedding is to sum all node vectors up, which has been widely used as a baseline in graph embedding approaches. Unlike node embedding, graph embedding aims to directly map each graph in the graph database into a vector. graph2vec [NCV+17] uses document embedding neural networks to embed node-labeled graphs. Node embedding methods could also be used in graph embedding by averaging the embedding of nodes. Variational graph autoencoder [KW16] encodes the graph by a two-layer graph convolutional network and decodes by an inner-product decoder. However, few graph embedding methods could handle node labels and edge weights at the same time. An additional drawback of graph embedding is the low interpretability of the representations, and as a result, anomalousness.

7.2.2 Direct Anomaly Detection for Graphs

To seek higher interpretability, we turned to graph-based anomaly detection. Anomaly detection has been studied extensively for its applicability to real-world scenarios. Careful scrutiny of these studies can be found in [ATK15]. On the one hand, some studies try to spot the graph anomalies by mining graphs' structural features. OddBall [AMF10] detects anomalous nodes in a *single* weighted graph, but does not extend to graph databases. ReFeX [HGL+11] includes recursive features to extract information even beyond direct neighbors. To increase interpretability, features are analyzed in pairs in [KLKF14], which can be easily visualized and point out the outliers. LookOut [GES+18] further turns the anomaly detection into a 2-dimension plots selection problem, and picks up the most explainable plots to the anomalies. SpotLight [EFGM18]

Table 7.2: Table of Symbols and Definitions.

Symbol	Definition
$\mathcal{G}$	Graph database
G_i	i-th graph in the database
I	Number of graphs in database
J	Total number of iterations in our algorithm
$\mathcal{V}_i$	Node set of i-th graph
$\mathcal{E}_i$	Edge set of i-th graph
$\mathcal{T}$	Set of unique node labels
$t(v)$	Label of node v
$w(u, v)$	Edge weight of edge (u, v)
P_j	Substructure found in iteration j
vbits	Encoding length for nodes
rbits	Encoding length for adjacency matrix
ebits	Encoding length for edges
mbits	Encoding length for edge weight corrections
wbits	Encoding length for rewiring

Table 7.3: Table of Acronyms.

Acronym	Definition
MDL	Minimum Description Length
AUC	Area Under Curve
AP	Average Precision

aims to detect the anomalies in the streaming graphs. On the other hand, some researchers seek to explain the graph anomalousness by frequent patterns among graphs. Cook *et al.* [CH94] proposed a graph substructure discovery framework, which Noble and Cook [NC03] leverage in anomaly detection by using compression rates in each iteration. Eberle *et al.* [EH07] detect unexpected structural deviations, defined as frequent patterns with slight changes. These structure-based methods do not take edge weights into consideration. For numerical weights, Yagada [DLMR11] uses discretization to assign edges with discrete labels. However, discretization loses information and underperforms as we demonstrate in our experiments.

7.3 Problem Definition and General Framework

We consider that a graph database is given, which consists of I *node-labeled directed weighted graphs* $\mathcal{G} = \{G_1(\mathcal{V}_1, \mathcal{E}_1), \ldots, G_I(\mathcal{V}_I, \mathcal{E}_I)\}$, where each graph $G_i(\mathcal{V}_i, \mathcal{E}_i)$ has a set of labeled nodes $\mathcal{V}_i$ and a set of weighted edges $\mathcal{E}_i$. For each node $v \in \mathcal{V}_i$, $t(v) \in \mathcal{T}$ denotes the label of node v, where $\mathcal{T}$ represents the set of unique node labels, e.g., types of company accounts. Each edge $(u, v) \in \mathcal{E}_i$ has a weight $w(u, v)$, e.g., number of transactions between two accounts. An overview of symbols and acronyms is provided in Table 7.2 and Table 7.3, respectively.

7.3.1 Problem Definition

Our anomaly detection problem is defined as follows:

Algorithm 7.1: General Framework for Graph Anomaly Detection (followed by [CH94, NC03], blue text points out the differences with GAWD)

Data: A graph database
Result: Anomaly scores for all graphs
1 **while** *True* **do**
2 Detect frequent patterns in graph database;
3 **if** *No pattern is found* **then**
4 Break;
5 **end**
6 Identify the pattern which can compress the graphs in database the most;
7 Compress the graphs by this pattern;
8 **end**
9 Compute the anomaly scores by compression rate;

Problem 7.1: Anomaly Detection in Graph Database

Given a node-labeled directed weighted graph database $\mathcal{G} = \{G_1(\mathcal{V}_1, \mathcal{E}_1), ..., G_I(\mathcal{V}_I, \mathcal{E}_I)\}$, *compute* anomaly scores a_i for each graph $G_i \in \mathcal{G}$, such that a higher score indicates a higher abnormality.

7.3.2 General Framework

Our method follows a general information-theoretic framework depicted in Algorithm 7.1. This framework generalizes previous graph anomaly detection methods, i.e., [CH94, NC03]. Given a graph database, the idea is to iteratively identify the "best" substructure that yields the largest compression, replacing each of its occurrences with a super-node. Each graph in the database is then scored by how much it compresses over iterations — the more the compression, the lower the anomaly score. In Algorithm 7.1, blue texts pinpoints the differences in GAWD compared to those in [CH94, NC03].

In particular, the existing method in [NC03] detects the frequent patterns in Line 2 by beam search. To identify the best pattern, they use the one that can minimize the total description length of graphs in the database in Line 6. However, they only take the compressed node and edge information into consideration (lossy encoding). They then compress the graphs by that pattern in Line 7. This process will keep running until no more pattern is found. The heart of their approach is the encoding scheme of graphs by Minimum Description Length (MDL) principle, which includes encoding the structure of graphs, i.e., nodes and edges.

The total encoding length for nodes is:

$$\text{vbits}(G_i) = \log^* |\mathcal{V}_i| + |\mathcal{V}_i| \log_2 |\mathcal{T}| \tag{7.1}$$

where $|\mathcal{T}|$ denotes the number of unique node labels in G_i. $\log^*$ is the universal code length used to encode the numeric value. We first need $\log^* |\mathcal{V}_i|$ bits to encode the number of nodes, and then need $\log_2 |\mathcal{T}|$ bits to encode the label for each node.

The total encoding length for the adjacency matrix is:

$$\text{rbits}(G_i) = \log^* b + \sum_{p=1}^{|\mathcal{V}_i|} \log_2 (b+1) + \log_2 \binom{|\mathcal{V}_i|}{k_p} \tag{7.2}$$

where b denotes the highest out-degree in G_i, and k_p denotes the particular out-degree of p^{th} node. $\log^* b$ bits are needed to encode the highest out-degree. For each row of adjacency matrix, $\log_2 (b+1)$ bits are needed to encode the degree of node. Given k_p as the number of 1(s) occurring in p^{th} row, we know that there are only $\binom{|\mathcal{V}_i|}{k_p}$ possible permutations, so we need $\log_2 \binom{|\mathcal{V}_i|}{k_p}$ bits to encode the positions of 1(s) in the p^{th} row.

The total encoding length for edges is:

$$\text{ebits}(G_i) = \log^* m + |\mathcal{E}_i| \log_2 m \tag{7.3}$$

where m denotes the largest edge weight. We first need $\log^* m$ bits to encode the largest edge weight, and then $\log_2 m$ bits to encode the weight for each edge.

Thus, the total encoding length for G_i in j-th iteration is:

$$DL(G_i(j)) = \text{vbits}(G_i(j)) + \text{rbits}(G_i(j)) + \text{ebits}(G_i(j)) \tag{7.4}$$

Different from [NC03], in GAWD, we replace the beam search in Line 2 by $gSpan$ [YH02], which is a much faster subgraph mining technique; we design a novel graph encoding scheme, being used in Line 6, which accepts edge weight (described in Section 7.4.1) and is lossless (described in Section 7.4.2).

7.4 Proposed Method: GAWD

Next we provide the details of GAWD. Given a substructure $P_j = (\mathcal{V}_j, \mathcal{E}_j)$ at iteration j, which is a node-labeled simple graph, the first task is to identify a "representative" weight for edge $(u, v) \in \mathcal{E}_j$, denoted $w^*_{P_j}(u, v)$. Given the edge-weighted P_j, our encoding scheme involves:

1. Encoding P_j,
2. Encoding each compressed graph $\overline{G}_i = (\overline{V}_i, \overline{\mathcal{E}}_i)$ resulted from replacing each occurrence/instance of P_j (ignoring edge weights) in G_i with a super-node, and
3. Encoding auxiliary information for lossless reconstruction of G_i, given P_j and $\overline{G}_i$.

Steps (1) and (2) use the encoding scheme in Subdue [CH94], the details of which we omit due to space limit. Our novel designs are in Step (3), specifically:

D1. Weight Encoding: for handling edge weights, and

D2. Rewiring Encoding: for enabling lossless reconstruction.

Total compression cost (or description length) of the graph database is the sum of bits used for encoding (1)–(3).

7.4.1 Design 1: Weight Encoding

7.4.1.1 Representative Weight Discovery

Given a substructure P_j, the representative edge weight $w^*_{P_j}(u,v)$ for each edge $(u,v) \in \mathcal{E}_j$ need to be identified before evaluating the substructures toward compression. Let $\mathcal{E}_{j,(u,v)}$ denote all the edges in the instances of substructure P_j in the database corresponding to $(u,v) \in \mathcal{E}_j$. We turn this into an optimization problem based on the Minimum Description Length (MDL) encoding. Given a candidate weight w, we denote the bits needed to correct with respect to the true weight of an edge instance $(s,t) \in \mathcal{E}_{j,(u,v)}$ by $L(w, w_{P_j}(s,t))$ (details in Section 7.4.1.2). The optimization problem is then formulated as:

$$w^*_{P_j} = \min_{w} \sum_{(s,t)\in\mathcal{E}_{j,(u,v)}} L(w, w_{P_j}(s,t)) \tag{7.5}$$

While not convex, the optimization in Equation (7.5) is 1-dimensional and hence easy to solve. We employ Dichotomous Search [CZ04], which returns the optimal solution in most cases. It efficiently takes only $O(|\mathcal{E}_{j,(u,v)}| \log_2 R)$, where R is the numeric search range of weights.

7.4.1.2 Weight Corrections

After discovering $w^*_{P_j}$, we encode the weights in each instance. For each super-node $s \in \overline{V}_i$ of $\overline{G}_i$, we denote by $g_s = (\mathcal{V}_s, \mathcal{E}_s)$ the substructure instance in G_i corresponding to s, which is isomorphic to P_j in structure. For each edge $(u,v) \in \mathcal{E}_s$, we encode its weight correction using:

$$L(w,w') = \begin{cases} 1 \text{ bit} & \text{if } w - w' = 0 \\ 2\log_2(|w-w'|) + 3 \text{ bits} & \text{otherwise} \end{cases} \tag{7.6}$$

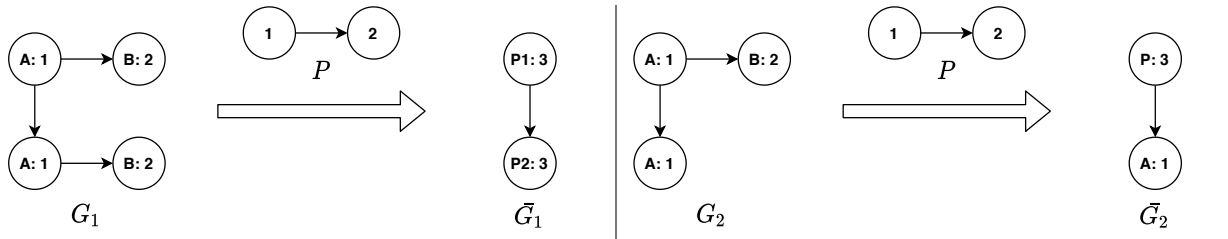
where $w = w^*_{P_j}(u,v)$ and $w' = w_{g_s}(u,v)$. 1 bit is used to identify whether the weight correction is needed. If so, an extra 1 bit is used to record the sign of the error. $2\log_2(|w-w'|)+1$ bits is used to encode the numeric value by universal code.

We remark that instead of discretizing edge weights into labels, our encoding scheme handles the numeric value and is lossless. Thus, the total encoding length of weight corrections is as follows:

$$\text{mbits}(G_i) = \sum_{s\in\overline{V}_i} \sum_{(u,v)\in\mathcal{E}_s} L(w, w^{'}) \tag{7.7}$$

7.4.2 Design 2: Rewiring Encoding

After replacing P_j with a super-node, all the edges connected to P_j merge into super-edges. Weight of a super-edge $e = (x,y) \in \overline{\mathcal{E}}_i$, denoted $w_{\overline{G}_i}(x,y)$, is the sum of the weights of all edges that it represents. For lossless reconstruction, the edge (re)connectivity information needs to be encoded. As shown in Figure 7.2, there are two possible cases: (1) both x and y are super-nodes corresponding to non-overlapping instances of P_j, and (2) only one of them is a super-node.



(a) Case 1: Two super-nodes

(b) Case 2: One super-node and one regular node

Figure 7.2: **Rewiring Encoding:** Two cases that the edge (re)connectivity information are different. (a) Case 1: A super-edge is created between two super-nodes after compression. (b) Case 2: A super-edge is created between one super-node and one regular node after compression.

For the former case, we first encode the cardinality of e, denoted c_e, depicting how many edges it represents, using:

$$L(c_e) = \log_2(|\mathcal{V}_j|^2) = 2\log_2(|\mathcal{V}_j|) \text{ bits} \tag{7.8}$$

For each edge, we encode substructure node IDs of its source and destination using $2\log_2(|\mathcal{V}_j|)$ bits total, and then encode its weight using $\log_2(w_{\overline{G}_i}(x,y))$ bits.

For the latter case, w.l.o.g. let x be the super-node. We encode how many edges e branches to, denoted b_y, using:

$$L(b_y) = \log_2(|\mathcal{V}_j|) \text{ bits} \tag{7.9}$$

In other words, b_y denotes how many distinct nodes within g_x that y connects to. For each edge we encode the substructure node ID of y's neighbor $n \in \mathcal{V}_x$ using $\log_2(|\mathcal{V}_j|)$ bits. We then encode the weight of each edge the same as in the former case.

The total encoding length for rewiring is:

$$\begin{aligned} &\text{wbits}(G_i) = \textstyle\sum_{e=(x,y)\in\overline{\mathcal{E}}_i}(|e|-1)\log_2(w_{\overline{G}_i}(x,y)) \\ &+ \begin{cases} (|e|+1)L(c_e) & \text{if } x,y \text{ are both super nodes} \\ (|e|+1)L(b_y) & \text{if } x \text{ is super node} \end{cases} \end{aligned} \tag{7.10}$$

where $e = (x,y)$ denotes the super-edge connecting from node x to node y in compressed graph $\overline{G}_i$, and $|e|$ denotes the multiplicity of the super-edge.

7.4.3 Overall Algorithm

The total encoding length of graph G_i in j-th iteration is given as the following:

$$DL^*(G_i(j)) = DL(G_i(j)) + \text{mbits}(G_i(j)) + \text{wbits}(G_i(j)) \tag{7.11}$$

Algorithm 7.2 gives the steps of GAWD. We use $gSpan$ [YH02] for frequent substructure mining in Line 3. In Lines 4-6, we search for the best weighted substructure yielding the largest

Algorithm 7.2: GAWD-Anomaly Scoring

Data: A database $\mathcal{G} = \{G_1, ..., G_I\}$, min support range $(ms_{\max}, ms_{\min})$, decay rate d (i.e., 0.9 by default).

Result: Anomaly scores $\mathbf{a} = \{a_1, ..., a_I\}$ for all graphs in $\mathcal{G}$

1 Initialization: $ms = ms_{\max}; j = 0;$
2 **while** $ms \geq ms_{\min}$ **do**
3 $\quad \mathcal{P}_j = gSpan(\mathcal{G}, ms);$
4 $\quad$ Discover $w^*_{P_j}(u, v)$ for all $P_j \in \mathcal{P}_j$ (See Section 7.4.1.1);
5 $\quad$ Identify $P^*_j \in \mathcal{P}_j$ yielding largest (positive) compression;
6 $\quad$ **if** *no* P^*_j *is found* **then**
7 $\quad\quad ms := ms * d;$
8 $\quad\quad$ Continue;
9 $\quad$ **end**
10 $\quad$ Compress $\mathcal{G}$ by P^*_j and save c^j_i in (7.12) for all $G_i \in \mathcal{G}$;
11 $\quad j := j + 1;$
12 **end**
13 $a_i = 1 - \frac{1}{j} \sum_{k=1}^{j} \left[(j - k + 1) * c^k_i\right]$ for all $G_i \in \mathcal{G}$;
14 Return: $\mathbf{a}$;

compression. To improve the efficiency, we gradually decrease minimum support from maximum value to minimum in Lines 7-9, if there is no substructure found. Once we identify the best substructure P_j in iteration j, we compress the graphs in the database by P_j and save the compression rate c^j_i, for each graph i, defined as:

$$c^j_i = \frac{DL^*_{j-1}(G_i) - DL^*_j(G_i)}{DL^*_0(G)} \tag{7.12}$$

where $DL_j(G_i)$ is the description length of G_i after j iterations. Finally, we compute the anomaly scores in Line 13. The anomaly score ranges from 0 to 1, where 1 means the most anomalous and 0 means the least anomalous. The compression rates are linearly weighted by the term $j - k + 1$, where it means that the earlier we identify the substructure as the best one, the less anomalous that the graphs containing it are.

Complexity Analysis The time complexity of GAWD is as follows:

Lemma 7.1

GAWD has linear time complexity, $O(nI|\mathcal{E}| \log |\mathcal{V}|)$, where n denotes the number of frequent substructures, I denotes the number of graphs, and $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the average numbers of nodes and edges.

Proof. In the worst case, n frequent substructures are detected by $gSpan$ and used for compression with no conflict, then GAWD at most will iterate n times. Moreover, $O(I|\mathcal{E}|\log|\mathcal{V}|)$ is the time complexity of $gSpan$, where I denotes the number of graphs in graph database, $|\mathcal{E}|$ denotes the average edge number of graphs, and $|\mathcal{V}|$ denotes the average node number. The Dichotomous Search is also efficient, taking $O(|\mathcal{E}_{j,(u,v)}|\log_2 R)$, where $\mathcal{E}_{j,(u,v)}$ denotes all the edges in the instances of substructure P_j, and R denotes the numeric search range of weights. For compression, it takes $O(I|\mathcal{E}|)$ to redirect edges for each graph. The complexity of GAWD is $O(n(|\mathcal{E}_{j,(u,v)}|\log_2 R + I|\mathcal{E}| + I|\mathcal{E}|\log|\mathcal{V}|))$. Empirically, $|\mathcal{E}_{j,(u,v)}|$ and $\log_2 R$ are small constant values which are negligible. Therefore, the complexity is $O(nI|\mathcal{E}|\log|\mathcal{V}|)$. ■

7.5 Experiments

We design experiments to answer the following questions:
RQ1. **Effectiveness**: How well does GAWD work on anomaly detection?
RQ2. **Scalability**: How does GAWD's running time grow with input size?
Experiments are run on a machine with 3.2 GHz CPU and 256 GB RAM.

Datasets We use four datasets illustrated in Table 7.4. The detailed description of all datasets are shown as follows:

- **UCI Message Dataset [OP09]:** This recorded the communications between students at UCI where nodes and edges denote students and messages respectively. To capture the role information, we adopt role2vec [ARL+18] to embed nodes in the complete graph, and use the 10 groups clustered by Agglomerative Clustering as the node labels. The data is split into hours to form a graph database.
- **Enron Email Dataset [KY04]:** This contains the emails passing between colleagues in Enron Company from 2000 to 2002. We assign the job positions to each employee as node labels. The data is split into day communication graphs to form a graph database.
- **Accounting Dataset:** This is from an anonymous accounting institution, containing accounts (nodes) and transactions (edges) that reflect the money flow between company accounts. Each graph captures a set of transactions within a unique expense report.
- **Synthetic Accounting Dataset:** Since the accounting dataset is proprietary, we generate a synthetic database with generated graphs following the same statistical characteristics as in the accounting graphs.

We treat edge multiplicities as weights for all four graph databases. There are no ground truth anomalies in all the databases. To evaluate effectiveness, we inject anomalies into randomly sampled 3% of the graphs in each database as [ATK15] suggested, which had also been done by several other studies [YCA+18, ZLL+19]. For each sampled graph, we (i) randomly select an edge $(u, v) \in \mathcal{E}_i$, and (ii) multiply its weight $w(u, v)$ by 10^d, where d denotes the digit count of the upper fence in the boxplot of weights for $(t(u), t(v))$ edges. This aims to simulate unusual behaviors between the users or accounts.

Table 7.4: Statistics of graph databases.

Name	Graphs	Nodes [min, max]	Edges [min, max]
UCI Message Dataset [OP09]	3320	[2, 159]	[1, 193]
Enron Email Dataset [KY04]	843	[2, 87]	[1, 127]
Accounting Dataset	16,026	[2, 13]	[1, 20]
Synthetic Accounting Dataset	15,935	[2, 13]	[1, 18]

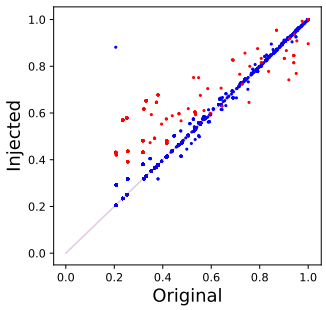


Figure 7.3: Anomaly score of each graph before (original) vs. after (injection): Database originally contains many graphs with high scores. Red dots depict the graphs being injected.

Evaluation Metric Datasets may originally contain anomalies, but we do not have ground truth. As shown in Figure 7.3, there originally exist multiple graphs with high anomaly scores along the horizontal axis, which strongly disturb the quality of evaluation. To solve this, rather than comparing all graphs in absolute terms, we look at the relative change in anomaly scores before and after injection. We quantify the relative change as:

$$RC_i = \frac{a_i^{\text{injected}} - a_i^{\text{original}}}{a_i^{\text{original}}} \,. \tag{7.13}$$

To evaluate the performance of anomaly detection, we use precision@k, recall@k, Area Under Curve (AUC) and Average Precision (AP) as our evaluation metrics. The choices of k are dependent on the database size since k cannot exceed the number of injected graphs.

Baselines We compare GAWD with four baselines:

- **Noble *et al.***: [NC03] iteratively finds the substructure generating the largest compression, and then assigns anomaly scores based on the compression rate in each iteration.
- **Subdue-W**: follows [NC03] but additionally discretizes edge weights into labels by ten bins with equal size.
- **node2vec**: [GL16] embeds each node into a vector, then inputs sum of node vectors in each graph to Isolation Forest [LTZ08].
- **graph2vec**: [NCV+17] embeds each graph into a vector and inputs it to Isolation Forest.

Method	Precision @45	Precision @90	Recall @45	Recall @90	AUC	AP	Time
node2vec	6.7	5.6	3.0	5.1	47.6	3.1	59
graph2vec	2.2	1.1	1.0	1.0	48.7	3.1	3
Noble *et al.*	0.0	0.0	0.0	0.0	47.6	3.1	43988
Subdue-W	100.0	60.0	45.5	54.5	94.8	68.6	32621
GAWD	100.0	92.2	45.5	83.8	93.8	90.3	760

(a) UCI Message Dataset

Method	Precision @10	Precision @20	Recall @10	Recall @20	AUC	AP	Time
node2vec	10.0	5.0	4.0	4.0	58.7	4.6	25
graph2vec	10.0	5.0	4.0	4.0	59.0	6.1	1
Noble *et al.*	0.0	0.0	0.0	0.0	51.7	3.2	7768
Subdue-W	10.0	5.0	4.0	4.0	48.2	4.9	8443
GAWD	90.0	85.0	36.0	68.0	82.1	73.8	207

(b) Enron Email Dataset

Method	Precision @200	Precision @400	Recall @200	Recall @400	AUC	AP	Time
node2vec	5.0	3.0	2.1	2.5	47.0	30.0	31
graph2vec	3.5	4.5	1.5	3.8	54.0	3.7	13
Noble *et al.*	3.0	3.1	1.2	2.6	50.6	3.1	460
Subdue-W	82.0	74.0	34.2	61.7	76.0	59.8	477
GAWD	100.0	81.5	41.7	67.9	88.0	71.0	75

(c) Accounting Dataset

Method	Precision @200	Precision @400	Recall @200	Recall @400	AUC	AP	Time
node2vec	2.5	3.3	1.0	0.8	48.3	2.8	26
graph2vec	0.0	2.0	1.7	3.1	49.6	3.0	14
Noble *et al.*	3.0	3.0	1.3	2.5	50.0	3.0	1426
Subdue-W	63.5	35.0	26.6	29.3	71.5	36.8	6584
GAWD	100.0	89.0	41.8	74.5	95.3	90.2	176

(d) Synthetic Accounting Dataset

Table 7.5: **GAWD significantly outperforms all the baselines:** We show the performance of GAWD and structure-based and embedding-based baselines on three real-world and one synthetic graph datasets. Green (■) marks the winner.

7.5.1 RQ1 – Effectiveness

In Table 7.5, GAWD outperforms most of the baselines significantly on all the datasets. Noble *et al.* and graph2vec fail as it cannot handle edge weights. GAWD shows 31.6%, 1365%, 18.7% and 145% improvement over Subdue-W in average precision on four datasets respectively, highlighting the insufficiency of discretization to handle edge weights. Even if node2vec accepts edge weights, it is not sensitive enough to detect the anomalies on weights.

The effectiveness of GAWD shows the necessity of handling numerical values instead of discretizing them into labels. In addition to improving performance on anomaly detection, we simultaneously maintain interpretability, where the common substructures identified in the course of iteratively compressing the database provide a peek into the expected structural patterns in the database, and the lack thereof in anomalous graphs.

7.5.2 RQ2 – Scalability

To quantify the scalability, we empirically vary the number of (i) total edges and (ii) graphs in the database, both of which highly correlate with running time. In Table 7.4, the total number of graphs in the first experiment is 12,617, and the total edge number in the second experiment is 24,137. In Figure 7.4, GAWD scales linearly w.r.t. both variables, with R^2 scores of linear-fit models higher than 0.97.

In Figure 7.1, GAWD achieves the best trade-off between performance and running time compared to the state-of-the-art approaches. In type injection, GAWD is 3.7× faster than Noble *et al.* and the average precision is 1.3× higher than node2vec; in path injection, GAWD is 3.9× faster than Noble *et al.* and the average precision is also 1.1× higher; in weight injection, GAWD is 4.1× faster than Subdue-W and the average precision is also 1.5× higher.

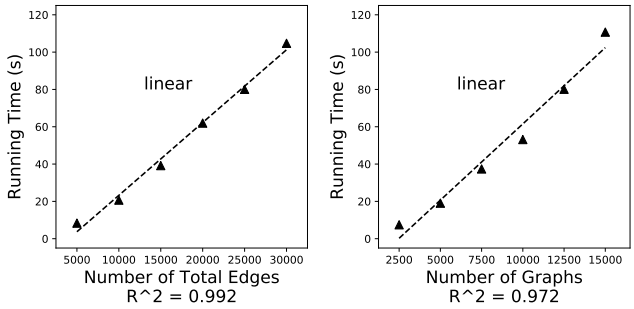


Figure 7.4: **GAWD is scalable:** linear on the number of total edges (left) and the number of graphs (right).

7.6 Conclusion

We present GAWD, addressing the graph anomaly detection problem in a directed weighted graph database. Using an MDL-based approach for encoding, GAWD iteratively identifies the "best" substructure yielding the largest compression of the database. Our novel encoding scheme includes lossless encoding as well as ability to handle weighted graphs.

- **Lossless encoding scheme** for graph compression;
- **Handling edge weights**, through discovering and encoding a representative weight followed by delta-corrections to maintain lossless encoding;
- **Interpretability** through structural patterns (i.e., substructures) and **linear scalability** with the input size.

Experiments on four data sets with injected anomalies show that GAWD achieves superior results among state-of-the-art baselines.

Chapter 8

RWK$^+$: Descriptive Kernel Convolution Network with Improved Random Walk Kernel

Chapter based on work that appeared at WWW 2024 [LZA24] [PDF].

Graph kernels used to be the dominant approach to feature engineering for structured data, which are superseded by modern graph neural networks (GNNs) as the former lacks learnability. Recently, a suite of Kernel Convolution Networks (KCNs) successfully revitalized graph kernels by introducing learnability, which convolves input with learnable hidden graphs using a certain graph kernel. The random walk kernel (RWK) has been used as the default kernel in many KCNs, gaining increasing attention.

In this chapter, we first revisit the RWK and its current usage in KCNs, revealing several shortcomings of the existing designs, and propose an improved graph kernel RWK$^+$, by introducing color-matching random walks and deriving its efficient computation. We then propose RWK$^+$CN, a KCN that uses RWK$^+$ as the core kernel to learn descriptive graph features with an unsupervised objective, which cannot be achieved by GNNs. Furthermore, by unrolling RWK$^+$, we discover its connection with GNNs, and propose our novel GNN layer RWK$^+$Conv.

In the first part of experiments, we demonstrate the descriptive learning ability of RWK$^+$CN with the improved random walk kernel RWK$^+$ on unsupervised pattern mining tasks; in the second part, we show the effectiveness of RWK$^+$ for a variety of KCN architectures and supervised graph learning tasks, and demonstrate the expressiveness of RWK$^+$Conv layer, especially on graph-level tasks. Our methods adapt to various real-world applications, including web applications such as bot detection in a web-scale Twitter social network, and community classification in Reddit social interaction networks.

8.1 Introduction

Graph kernels are functions that measure the similarity between pairs of graphs. They have historically been a popular approach to "flatten" graphs explicitly or implicitly into vector form that many downstream algorithms can more easily handle. While graph kernels exhibit mathematical expressions that lend themselves to theoretical analysis [GFW03], their handcrafted features may not be expressive enough to capture the complexities of various learning tasks on graphs [RG03]. More recently, graph kernels are superseded by modern graph neural networks (GNNs) which leverage multi-layer architecture and nonlinear transformations to learn task-adaptive graph representations [ZCH+20b].

Interestingly, GNNs bear a close connection to the Weisfeiler-Leman (WL) graph kernels [SSvL+11], as well as the related WL graph isomorphism test [LW68]. In fact, most recent work on the expressive power of GNNs heavily use the k-WL hierarchy [XHLJ19, Sat20], and others have derived inspiration from it to design novel GNN architectures [MRF+19, MBSL19, BFZB23, ZSA22]. The WL kernel, which is quite popular thanks to its attractive linear-time complexity [HK09], derives its simplicity from iterative neighborhood aggregation, akin to the convolution scheme of message-passing GNNs [GSR+17]. This type of connection has been recognized and leveraged in the recent few years to derive a series of "*GNNs meet graph kernels*" style models that bridge these two worlds [Mai16, LJBJ17, CJM20, DHS+19, NV20, CMB+21, FYWT22, APZ19, LJWS21], named as Kernel Convolutional Networks (KCNs).

RWKs, based on the number of (node label sequences along) walks that two graphs have in common, have been the starting point in the history of graph kernels [GFW03, KTI03]. A recent study by [Kri22] demonstrated that classical random walk kernels with minor modifications are as expressive as the Weisfeiler-Leman kernels and even surpass their accuracy on classification tasks. Inspired by this, our work extends from random walk neural network (RWNN) of [NV20], where each input graph is represented by its RWK similarity to a set of small graphlets (called hidden graphs) that are learned end-to-end by optimizing a classification objective.

We deepen the synergy between GNNs and graph kernels, and improve the RWK as utilized within GNNs in a number of fronts. First, toward capturing more representative patterns, we introduce several improvements to the RWK in both effectiveness and efficiency and propose an improved graph kernel RWK$^+$. Second, we propose a *descriptive* KCN RWK$^+$CN by flipping the objective from a discriminative one to a descriptive one that helps us capture relational patterns in the graph database. What is more, we derive the mathematical connection of RWK$^+$ to layer-wise neural network operators for the first time, which inspires us to propose a novel GNN layer RWK$^+$Conv. Finally, we employ our RWK$^+$ and RWK$^+$Conv on a suite of real-world tasks for graph data and achieve significant gains. A summary of our contributions is as follows:

• **RWK$^+$ with efficient color-matching:** We identify that the RWK originally developed in RWNN only enforces the same label at the start and the end of two walks while ignoring the intermediates. We reformulate it to count a walk as shared only if all corresponding node pairs exhibit the same node label (i.e. color) at all steps along the walk. While more reflective of graph similarity, RWK with color-matching incurs a memory and computational overhead. Therefore, we propose the improved graph kernel RWK$^+$ through transforming its formulation for efficient computation. In addition, we propose a learnable solution StepNorm to address the nontrivial task of combining similarity scores across steps with drastically different scales.

• **RWK$^+$CN learning descriptive hidden graphs**: The original RWNN is trained supervised for graph classification and thus learns discriminative hidden graphs. We propose RWK$^+$CN with an unsupervised objective, that uses RWK$^+$ as the core kernel and maximizes the total RWK similarity between the input graphs and hidden graphs. The learned hidden graphs are reflective of the frequent walks (i.e. patterns) in the database. To further enhance the descriptive ability, we use additional "structural colors" to help better capture structural similarity between graphs, and enforce a diversity regularization among the hidden graphs to capture non-overlapping subgraphs. Finally, we demonstrate the descriptive learning ability of RWK$^+$CN with our carefully designed testbeds.

• **RWK$^+$Conv, a novel GNN layer**: By unrolling RWK$^+$, we discover that the derivation can be re-written as a sequence (i.e. multiple layers) of graph convolutional operations, connecting with regular graph convolutional networks (GCN) layers. By viewing hidden graphs as learnable parameters, we transform the RWK$^+$ algorithm into a novel GNN layer called RWK$^+$Conv. The RWK$^+$Conv layer extracts graph features by using additional element-wise product operations that can potentially bring better expressiveness than the GCN layer.

• **Broad applications of RWK$^+$ and RWK$^+$Conv**: We employ RWK$^+$ as the core kernel inside different KCN architectures and evaluate it on four graph-level tasks: one discriminative (graph classification), and three descriptive (graph pattern mining, graph-level anomaly detection, and substructure counting). It is shown to be improved over the vanilla RWK especially on descriptive tasks. Moreover, we compare our proposed RWK$^+$Conv layer with the GCN layer on node- and graph-level tasks. RWK$^+$Conv outperforms GCN in both tasks, notably by a large margin in graph-level tasks, empirically demonstrating its better expressiveness. It is worth noting that our experiments contain a broad-range of real-world applications, including web applications such as bot detection in a web-scale Twitter social network with a million nodes, and community classification in Reddit social interaction networks.

Reproducibility: Our code is available at `https://github.com/mengchillee/RWK_plus`.

8.2 Related Work

8.2.1 Graph Kernels

In Section 4.2.4, we introduced graph kernels that generate node-to-node similarity and are used to transform node features; here, we introduce a broader set of graph kernels, including those that compute a single similarity score between two graphs. The literature on graph kernels is extensive and well established, thanks to the prevalence of learning problems on graph-structured data and the empirical success of kernel-based methods [NSV21, KJM20]. A large variety of graph kernels have been developed motivated either by their theoretical properties, or specialization or relevance to certain application domains like biology [Prz07, JLJZ16] or chemistry [RS10]. Those include graph kernels based on shortest paths [BK05], subtrees [RG03, MV09], graphlets [Prz07, SVP$^+$09], structural features [BKEF12], random walks [GFW03, KTI03, KTS12, KVF13], as well as variants such as random walk return probabilities [ZWX$^+$18], to name a few. A long line of work focused on designing computationally tractable kernels for large graphs

with discrete as well as continuous node attributes [SVP+09, CG10, FKP+13, MKKM16], while those such as the Weisfeiler-Leman (WL) kernel [SSvL+11] and others [HK09] gained popularity thanks to linear-time efficiency.

A key challenge with classical graph kernels is lack of learnability; today's graph neural networks (GNNs) are able to learn feature representations that clearly supersede the fixed feature representations used by graph kernels. At the same time, several connections can be drawn between graph kernels and GNNs, such as the similarity between the neighborhood aggregation of the WL kernel (a.k.a. color refinement) and the scheme of message-passing GNNs [GSR+17]. We discuss below recent line of work that tap into the synergy between graph kernels and GNNs to harvest the best of both worlds.

8.2.2 Synergizing Graph Kernels and GNNs

While many works bridge graph kernels with graph neural networks (GNNs), they have clear distinctions. Coined as Convolutional Kernel Networks [MKHS14], and others in similar lines [Mai16, CJM20], introduce neural network architectures that learn graph representations that lie in the reproducing kernel Hilbert space (RKHS) of graph kernels. Others design new classes of graph kernels using GNNs [DHS+19, HJ15]. In contrast, and closest to our work, coined very similarly as Graph Kernel Convolution Networks (KCNs) [CMB+21] and various others [LJBJ17, NV20, FYWT22] integrate a graph kernel *into* GNN architectures. In other words, they show how to realize a given graph kernel with a GNN module, which in effect unlocks end-to-end learnability for the graph kernel. We provide further background on KCNs in Section 8.3.1. Finally, while different in focus, there is also noteworthy work exploiting graph kernels for pre-training GNNs [NTS18], or to extract preliminary features that are passed onto a convolutional neural networks (CNNs) [NMT+18].

8.3 Kernel Convolution Networks with RWK: Issues

Kernel Convolution Network (KCN) [CMB+21, NV20, FYWT22] that convolves the input graph with learnable hidden graphs using a certain graph kernel has gained increasing attention, as it offers learnability to graph kernels. Given the simplicity of random walk kernel (RWK) and its differentiability, it has been used as the default graph kernel in many KCNs [NV20, FYWT22]. We first introduce the notation and background of KCN (Section 8.3.1). Then we revisit the RWK (Section 8.3.2), and discuss the issues of its current usage in KCNs (Section 8.3.3).

Notation An overview of symbols and acronyms is provided in Table 8.1 and Table 8.2, respectively. Let $G = (V(G), E(G), l_G)$ denote an undirected, node-attributed graph with n nodes in $V(G)$, e edges in $E(G)$, and an attribute or labeling function $l_G : V(G) \to C$ where C can be $\mathbb{R}^d$ for continuous attributes or $\{c_1, ..., c_d\}$ for distinct discrete labels. Let $\mathbf{A}_G$ denote the adjacency matrix, and $\mathbf{A}_{G\otimes H} := \mathbf{A}_G \otimes \mathbf{A}_H$ depict the Kronecker product of the adjacency matrices for graphs G and H, as introduced in Definition 2.7. Let $\mathbf{X}_G := [\mathbf{x}_{v_1}, \ldots, \mathbf{x}_{v_n}]^\intercal \in \mathbb{R}^{n\times d}$ be the node attributes in G.

Table 8.1: Table of Symbols and Definitions.

Symbol	Definition
G	Undirected and attributed graph
$V(G)$	Node set of G
$E(G)$	Edge set of G
$l(v)$	Label of node v
n	Number of nodes in G
e	Number of edges in G
d	Number of features in G
$\mathbf{A}_G$	Adjacency matrix of G
$\mathbf{X}_G$	Node feature matrix of G
W	Learnable hidden graph
$\mathbf{A}_W$	Learnable adjacency matrix of W
$\mathbf{X}_W$	Learnable node feature matrix of W
m	Number of nodes in hidden graph W
$\mathcal{K}$	Graph kernel for graph similarity computation
t	Number of steps for random walk kernel

Table 8.2: Table of Acronyms.

Acronym	Definition
RWK	Random Walk Kernel
KCN	Kernel Convolution Network
GNN	Graph Neural Network
GCN	Graph Convolutional Networks
AP	Average Precision
AUC	Area Under Curve
MAE	Mean Absolute Error

8.3.1 Kernel Convolution Networks

Graph kernels are designed to measure similarity on a pair of graphs. However, they produce fixed handcrafted features. [LJBJ17] derived the first neural network that outputs the RWK similarity scores between input graph and hidden learnable path-like graphs. [NV20] generalized [LJBJ17] such that the hidden graphs can have any structure without the path constraints. The designed model is claimed to be interpretable as the learned hidden graphs "summarize" the input graphs. Later, [CMB+21] and [FYWT22] extended RWNN [NV20] to a multi-layer architecture, in which each layer compares subgraphs around each node of the input with learnable hidden graphs. We refer to these models as Kernel Convolution Networks (KCNs) as they generalize the Convolutional Neural Network (CNN) from the image domain to the graph domain, with the help of a graph kernel. Each layer of the KCN has a number of learnable hidden graphs.

Formally, let G be the input graph with node $v \in V(G)$; let $\mathbf{h}^t(v) \in \mathbb{R}^{k_t}$ be the representation of node v at the t-th layer where k_t is the number of learnable kernels in KCN's t-th layer for $t > 0$, and k_0 be the dimension of original node attributes with $\mathbf{h}^0(v) = \mathbf{x}_v$. Let $W_1^t, ..., W_{k_t}^t$ denote the series of learnable hidden graphs in the t-th layer, and $\text{Sub}_G^t[v]$ be the subgraph around node v on G with attributes $\{\mathbf{h}^t(u) | u \in \text{Sub}_G[v]\}$, we have:

$$\mathbf{h}^{t+1}(v) = [\mathcal{K}(\text{Sub}_G^t[v], W_1^{t+1}), \ldots, \mathcal{K}(\text{Sub}_G^t[v], W_{k_{t+1}}^{t+1})] \,, \tag{8.1}$$

where $\mathcal{K}$ is the graph kernel used to compute graph similarity. Multi-layer KCNs stack graph kernel computations with layers, and output node representations at each layer which can be used for any downstream task. They exhibit strong representation ability however the output is not interpretable or descriptive. The single-layer KCN, while less expressive, can output meaningful similarity scores for descriptive unsupervised feature learning, which computes graph-level representation directly by:

$$\mathbf{h}(G) = [\mathcal{K}(G, W_1), \ldots, \mathcal{K}(G, W_k)] \tag{8.2}$$

8.3.2 Revisiting Random Walk Kernel

RWK has been used in KCNs as the default kernel. It has been originally proposed to compare two labeled graphs by counting the number of common walks on both graphs [GFW03, KTI03]. Formally, consider a labeled (discrete attribute) graph G such that $l(v)$ represents the label of node $v \in V(G)$. Let $\mathcal{R}_t(G)$ be the set of all t-step random walks on G. For a random walk $\mathbf{p} = (v_1, v_2, .., v_t) \in \mathcal{R}_t(G)$, let $l(\mathbf{p}) = (l(v_1), ..., l(v_t))$ denote the labels along the walk. Then the t-step RWK $\mathcal{K}^t_{rw}(G, H)$ computes the similarity of G and H by counting the common walks as follows:

$$\mathcal{K}^t_{rw}(G, H) = \sum_{i=1}^{t} \lambda_i \sum_{\mathbf{p} \in \mathcal{R}_i(G)} \sum_{\mathbf{q} \in \mathcal{R}_i(H)} \mathrm{I}\big(l(\mathbf{p}), l(\mathbf{q})\big) \tag{8.3}$$

where $\mathrm{I}(x, y)$ is the Dirac kernel where $\mathrm{I}(x, y) = 1$ if $x = y$, and 0 otherwise; and $\lambda_i \in \mathbb{R}$ denotes the weight of the i-th step's score.

Definition 8.1: Direct Graph Product

Given two labeled graphs G and H with labeling function l, their direct product is a new graph $G \times H$ with adjacency matrix $\mathbf{A}_{G\times H}$, vertices $V(G \times H) = \{(u, v) \in V(G) \times V(H) \mid l(u) = l(v)\}$ and edges $E(G \times H) = \{\big((u_1, v_1), (u_2, v_2)\big) \in V^2(G \times H) \mid (u_1, u_2) \in E(G)$ and $(v_1, v_2) \in E(H)\}$.

[GFW03] has shown that for any length t walk, there is a bijective mapping between $\mathcal{R}_t(G \times H)$ and $\{(\mathbf{p}, \mathbf{q}) \in \mathcal{R}_t(G) \times \mathcal{R}_t(H) \mid l(\mathbf{p}) = l(\mathbf{q})\}$. Therefore, Equation (8.3) can be rewritten as:

$$\mathcal{K}^t_{rw}(G, H) = \sum_{i=1}^{t} \lambda_i (\mathbf{1}^\intercal \mathbf{A}^i_{G\times H} \mathbf{1}) \tag{8.4}$$

where $\mathbf{1}$ denotes the all-ones vector of length $|V(G \times H)|$. Note that $\mathbf{A}_{G\times H}$ is *not* $\mathbf{A}_{G\otimes H}$, where the latter is the Kronecker product of $\mathbf{A}_G$ and $\mathbf{A}_H$ without label-matching along the walk.

8.3.3 Issues of Adapting RWK to KCN

The original RWK is designed for labeled graphs and cannot handle graphs with continuous node attributes KCNs are often used for. To that end, [NV20] proposed an extension of the RWK in their RWNN. Let $\mathbf{X}_G \in \mathbb{R}^{|V(G)|\times d}$ depict the d-dimensional continuous attributes for all nodes, and $\mathbf{X}_H \in \mathbb{R}^{|V(H)|\times d}$ and $\mathbf{A}_H \in \mathbb{R}^{|V(H)|\times|V(H)|}$ depict the learnable node features and the learnable adjacency matrix of the hidden graph H, respectively. For two graphs G and H, let $\mathbf{S} = \mathbf{X}_H \mathbf{X}_G^\intercal \in \mathbb{R}^{|V(H)|\times|V(G)|}$ encode the dot product similarity between the attributes of the vertices from two graphs, where $\mathbf{s} := \mathrm{vec}(\mathbf{S})$ is the 1-d vectorized representation of $\mathbf{S}$. The authors of RWNN proposed to compute the revised RWK as:

$$\mathcal{K}^t_{rw-}(G, H) = \mathbf{1}^\intercal (\mathbf{s}\mathbf{s}^\intercal \odot \mathbf{A}^t_{G\otimes H}) \mathbf{1} \ , \tag{8.5}$$

where $\odot$ denotes the element-wise (Hadamard) product in Definition 2.5. The revised kernel computes walks with length exactly t only. Mathematically, the term $\mathbf{s}\mathbf{s}^\intercal$ applies reweighting to $\mathbf{A}^t_{G\otimes H}$ such that the (i,j)-th element becomes $\mathbf{s}_i\mathbf{s}_j(\mathbf{A}^t_{G\otimes H})_{ij}$, where $(\mathbf{A}^t_{G\otimes H})_{ij}$ is the number of length-t walks from pair of nodes i to pair of nodes j in the Kronecker product graph $G\otimes H$.

Although the proposed adaptation of RWK can handle continuous node attributes, we identify two critical issues with Equation (8.5) that we outline below and later address in Sections 8.4.1 and 8.4.2, respectively.

Issue 1: Color Mismatch Let $\mathbf{p} = (u_1, ..., u_t)$ be a walk on G and $\mathbf{q} = (v_1, ..., v_t)$ be a walk on H. Equation (8.5) only considers reweighting the number of walks from (u_1, v_1) to (u_t, v_t), where (u_1, v_1) is the starting pair and (u_t, v_t) the ending pair, without comparing the intermediary nodes along the walk. In essence, their formulation of the RWK is limited to only partially shared walks.

Issue 2: Inefficient Parameterization Since $\mathbf{s} = \text{vec}(\mathbf{S}) = \text{vec}(\mathbf{X}_H\mathbf{X}_G^\intercal) = \sum_{i=1}^d(\mathbf{X}_G^{[i]} \otimes \mathbf{X}_H^{[i]})$, where $\mathbf{X}_G^{[i]}$ is the i-th column of $\mathbf{X}_G$, we can rewrite Equation (8.5) as:

$$\begin{aligned}
\mathcal{K}^t_{rw-}(G,H) &= \mathbf{1}^\intercal(\mathbf{s}\mathbf{s}^\intercal \odot \mathbf{A}^t_{G\otimes H})\mathbf{1} = \mathbf{s}^\intercal\mathbf{A}^t_{G\otimes H}\mathbf{s} \\
&= (\sum_{i=1}^d(\mathbf{X}_G^{[i]} \otimes \mathbf{X}_H^{[i]}))^\intercal(\mathbf{A}^t_G \otimes \mathbf{A}^t_H)(\sum_{i=1}^d(\mathbf{X}_G^{[i]} \otimes \mathbf{X}_H^{[i]})) \\
&= \sum_{i=1}^d\sum_{j=1}^d(\mathbf{X}_G^{[i]\intercal}\mathbf{A}^t_G\mathbf{X}_G^{[j]}) \otimes (\mathbf{X}_H^{[i]\intercal}\mathbf{A}^t_H\mathbf{X}_H^{[j]}) \\
&= \mathbf{1}^\intercal(\mathbf{X}_G^\intercal\mathbf{A}^t_G\mathbf{X}_G) \odot (\mathbf{X}_H^\intercal\mathbf{A}^t_H\mathbf{X}_H)\mathbf{1}
\end{aligned} \tag{8.6}$$

If H is a learnable hidden graph with parameters $\mathbf{A}_H \in \mathbb{R}^{m\times m}$ and $\mathbf{X}_H \in \mathbb{R}^{m\times d}$, the effective parameters are merely $\mathbf{X}_H^\intercal\mathbf{A}^t_H\mathbf{X}_H \in \mathbb{R}^{d\times d}$. That is, dimension d is an important degree of freedom for learnability, which can be small for certain real-world graphs.

8.4 Proposed Method

We first propose RWK$^+$ (Section 8.4.1), a color-matching based RWK along with an efficient computation method, which addresses the issues discussed earlier. Next, we propose RWK$^+$CN (Section 8.4.2), to increase the descriptive ability of the learned hidden graphs in the unsupervised setting. Finally, we propose RWK$^+$Conv (Section 8.4.3), a GNN layer inspired by the connections between RWK$^+$ and GNNs, to extract expressive graph features.

8.4.1 RWK$^+$: Proposed Efficient Color-Matching Random Walks

To address **Issue 1**, we propose an improved random walk kernel RWK$^+$ by deriving an effective formulation. First, notice that the original RWK for labeled graphs can be rewritten using the Kronecker product $\mathbf{A}_{G\otimes H}$ and one-hot encoded representation of the node labels.

We slightly change the $G \times H$ notation by introducing a set of "empty" nodes $\{(u,v) \in V(G) \times V(H) | l(u) \neq l(v)\}$ to the direct product graph $G \times H$. "Empty" nodes do not connect to any node hence this does not change the graph, rather they enlarge the size of $\mathbf{A}_{G\times H}$ to be the same as $\mathbf{A}_{G\otimes H}$. Given $\mathbf{X}_G$ and $\mathbf{X}_H$ as one-hot encoding of labels, the similarity matrix $\mathbf{S} = \mathbf{X}_H\mathbf{X}_G^\intercal$ is a binary valued matrix. Then, the following relation can be easily established:

$$\mathbf{A}_{G\times H} = \text{diag}(\mathbf{s})\mathbf{A}_{G\otimes H}\text{diag}(\mathbf{s}) = \mathbf{s}\mathbf{s}^\intercal \odot \mathbf{A}_{G\otimes H} \ , \tag{8.7}$$

where $\text{diag}(\mathbf{s})$ denotes a diagonal matrix with $\mathbf{s}$ being the diagonal. Thanks to Equation (8.7), we can rewrite the original RWK in Equation (8.4) as:

$$\mathcal{K}^t_{rw+}(G,H) = \sum_{i=1}^{t} \lambda_i [\mathbf{1}^\intercal (\mathbf{s}\mathbf{s}^\intercal \odot \mathbf{A}_{G\otimes H})^i \mathbf{1}] \ , \tag{8.8}$$

which is slightly different from Equation (8.5) with $\mathbf{s}\mathbf{s}^\intercal$ moving inside the power iteration. Although the derivation starts from labeled graphs, Equation (8.8) can be directly used for continuous attributed graphs without modification. Notice that this new formulation now takes all intermediary node attributes into consideration when comparing walks as intended.

Reformulation toward Efficient Computation The formulation in Equation (8.8) needs to compute the product graph between G and H which is inefficient in both memory and time. We establish an efficient computation by the mixed-product property of Kronecker product [Loa00] introduced in Definition 2.8, $(\mathbf{A}\otimes\mathbf{B})\text{vec}(\mathbf{S}) = \text{vec}(\mathbf{B}\mathbf{S}\mathbf{A}^\intercal)$, and rewrite the main part of Equation (8.8) step by step as follows:

$$\begin{aligned}
&\mathbf{1}^\intercal(\mathbf{s}\mathbf{s}^\intercal \odot \mathbf{A}_{G\otimes H})^i\mathbf{1} \\
=\ &\mathbf{1}^\intercal(\text{diag}(\mathbf{s})\mathbf{A}_{G\otimes H}\text{diag}(\mathbf{s}))^i\mathbf{1} \\
=\ &\mathbf{1}^\intercal\text{diag}(\mathbf{s}^{-1})(\text{diag}(\mathbf{s}^2)\mathbf{A}_{G\otimes H})^i\text{vec}(\mathbf{S}) \\
=\ &\mathbf{1}^\intercal\text{diag}(\mathbf{s}^{-1})(\text{diag}(\mathbf{s}^2)\mathbf{A}_{G\otimes H})^{i-1}\text{diag}(\mathbf{s}^2)\text{vec}(\mathbf{A}_H\mathbf{S}\mathbf{A}_G^\intercal) \\
=\ &\mathbf{1}^\intercal\text{diag}(\mathbf{s}^{-1})(\text{diag}(\mathbf{s}^2)\mathbf{A}_{G\otimes H})^{i-1}\text{vec}(\mathbf{S} \odot \mathbf{S} \odot (\mathbf{A}_H\mathbf{S}\mathbf{A}_G^\intercal))
\end{aligned} \tag{8.9}$$

The LHS thus can be computed *iteratively* by applying colored operations on the RHS repeatedly, using the procedure outlined in Algorithm 8.1 (where transpose is applied to all variables).

Algorithm 8.1: Fast Color-Matching RWK {and RWK$^+$Conv }

Data: $G = (\mathbf{A}_G \in \mathbb{R}^{n\times n}, \mathbf{X}_G \in \mathbb{R}^{n\times d})$; $H = (\mathbf{A}_H \in \mathbb{R}^{m\times m}, \mathbf{X}_H \in \mathbb{R}^{m\times d})$; max step t;
{$H \leftarrow W$ are parameters in RWK$^+$Conv }

Init: $\mathbf{Y}_0 \leftarrow \mathbf{X}_G\mathbf{X}_H^\intercal$, $\mathbf{Y} \leftarrow \mathbf{Y}_0$; {$\mathbf{Y}_0 \leftarrow \sigma(\mathbf{X}_G\mathbf{X}_H^\intercal)$ in RWK$^+$Conv }

for $i = 1, \ldots, t$ **do**

  $\mathbf{Y} \leftarrow \mathbf{A}_G\mathbf{Y}\mathbf{A}_H^\intercal$;

  $\mathbf{Y}^{(i)} \leftarrow \mathbf{Y}_0 \odot \mathbf{Y}$;

  $\mathbf{Y} \leftarrow \mathbf{Y}_0 \odot \mathbf{Y}^{(i)}$;

end

Return: $\sum_{i,j} \mathbf{Y}^{(t)}_{i,j}$ or $\sum_{i,j}(\sum_l \lambda_l \cdot \mathbf{Y}^{(l)})_{i,j}$;

Complexity Analysis The complexities of Equations (8.8) and (8.9) are as follows:

Lemma 8.1: Time and Memory Complexity

The time and memory complexities of the original color-matching random walk in Equation (8.8) are $O(em^2)$ and $O(em^2)$, respectively. The time and memory complexities of our efficient computation in Equation (8.9) are $O(em+nm^2)$ and $O(nm+m^2+e)$, respectively.

Proof. Let G be the sparse input graph with n nodes and e edges, and let H be the dense hidden graph W with m nodes. The original color-matching random walk in Equation (8.8) requires the explicit computation of Kronecker product, requiring runtime complexity $O(em^2)$ and memory complexity $O(em^2)$. In contrast, our efficient computation in Equation (8.9) requires runtime complexity $O(em + nm^2)$ and memory complexity $O(nm + m^2 + e)$. ■

Learnable Similarity Normalization In Equation (8.8) we compute the random walk similarity score for each step i iteratively. As each step counts the number of shared walks with length i, the scale of the similarity score across different steps can be considerably different, underscoring shorter walks. To combine these scores across different steps, we need to normalize the scores, which is nontrivial. To avoid hand-crafted normalization that needs hyperparameter tuning for different input, we introduce StepNorm that normalizes the score in a learnable way. StepNorm combines Batch Normalization [IS15] with the sigmoid function, where BatchNorm first standardizes the score to a Normal distribution and then shifts and rescales the score with learnable parameters. Sigmoid further normalizes the score to the range $[0, 1]$. We place StepNorm between lines 4 and 5 in Algorithm 8.1 to normalize the score step by step, and set $\lambda_l = 1$ for all $l \in [1, t]$.

8.4.2 RWK$^+$CN: Proposed Unsupervised Substructure Learning

KCNs were originally proposed for supervised learning, as such, the learned hidden graphs are discriminative for classification tasks. We claim that the KCN model can be paired with an unsupervised loss and used to learn *descriptive* hidden graphs instead, which is not achieved by existing GNNs. Given the output of a single-layer KCN model is the similarity scores to each hidden graph, one can train the KCN by maximizing the total similarity score. Specifically, the unsupervised objective is given as:

$$\max_{W_1,\ldots,W_k} \sum_{i=1}^{k} \mathcal{K}^t_{rw+}(G, W_i) \tag{8.10}$$

With this new objective, the learnable hidden graphs are to reflect or summarize the common patterns of the graph database. Put differently, similarities are maximized when the learned graphs capture frequent structural patterns that the kernel is designed to capture. Thanks to this unsupervised objective in Equation (8.10) and our RWK$^+$, KCNs can be used to learn descriptive hidden graphs. To further enhance the descriptive ability of the hidden graphs learned by KCN, we propose RWK$^+$CN, with two more important solutions:

Solution 1: Additional "Structural Colors" As discussed under **Issue 2** in Section 8.3.3, the input feature dimension d (i.e. number of node attributes) is an important degree of freedom for learnability for the original RWK in [NV20]. For RWK with color-matching, input features also play an important role as the similarity matrix $\mathbf{S} = \mathbf{X}_H \mathbf{X}_G^\intercal$ would be sparser with features that better characterize the nodes. Features that characterize structurally similar nodes also enable stronger feature matching at each step of a pair of walks between two graphs. Therefore, we propose to enrich the original node features with additional structural features in unsupervised learning of descriptive hidden graphs. As randomly initialized GNN can produce reasonable features for evaluating similarity in graph generation [TKG+22], we generate additional structural features through a fixed randomly initialized GNN to augment the original features.

Solution 2: Diversity Regularization When learning with more than one hidden graph without any constraint, the optimization may end up learning either the most frequent or otherwise very similar patterns. Therefore, we introduce diversity regularization $\mathcal{R}$ toward learning non-overlapping hidden graphs, defined as follows:

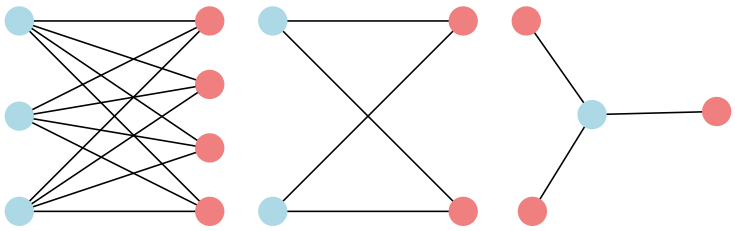
$$\mathcal{R}(W_1, \ldots, W_k) = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \mathcal{K}_{rw+}^{t}(W_i, W_j) \tag{8.11}$$

Overall unsupervised objective becomes maximizing the input graph to hidden graph similarities, while also minimizing the pairwise RWK similarities among hidden graphs $\mathcal{R}(W_1, \ldots, W_k)$.

8.4.3 RWK$^+$Conv: Proposed Graph Feature Extractor

Moreover, RWK$^+$ shares connections with Graph Convolutional Networks (GCN) [KW17]. If we view the hidden graph inside RWK$^+$ as learnable parameters, Line 4 of Algorithm 8.1 is given as $\mathbf{Y} \leftarrow \mathbf{A}_G \mathbf{Y} \mathbf{A}_W^\intercal$, which shares the *same* formulation as the graph convolutional operation in GCN, ignoring the activation function. Besides convolution-like computation, RWK$^+$ with learnable hidden graph also has a gated element-wise product as in Line 5 of Algorithm 8.1.

To demonstrate the connections with GNNs, we propose a novel GNN layer RWK$^+$Conv to extract graph features, based on Algorithm 8.1 (see the gray part). The major differences between RWK$^+$Conv and a normal GCNConv are: (1) element-wise product operation with $\mathbf{Y}_0$ motivated from node color matching; and (2) multi-step within a single convolution layer that shares the same parameter $\mathbf{A}_W$ and $\mathbf{X}_W$. Additionally, we make following changes to turn it into a neural network layer: (1) adding a sigmoid to $\mathbf{Y}_0$ to normalize the scale of similarity between 0 and 1; and (2) parameterizing $\mathbf{A}_W$ with a fully-connected layer. With the learnable hidden graphs and the additional element-wise product operation, we expect RWK$^+$Conv to bring better expressiveness than the GCN layer. We empirically demonstrate this point in Section 8.6.2, across many applications.



(a) Bipartite (b) Butterfly (c) Star

Figure 8.1: Task 1-1: Simple Subgraph Matching in Bipartite Graphs

8.5 Experiments I: Pattern Mining with RWK$^+$CN

Through a series of experiments, we show that RWK$^+$CN can be used for several unsupervised pattern mining tasks, and that each of our proposed solutions contribute to improved performance and descriptive ability. Pattern mining, which is typically a graph algorithm subject matter, is a very difficult task to achieve via machine learning. Since our major purpose is to demonstrate the advantages of RWK$^+$CN over RWNN, they are evaluated on a controlled testbed with ground truth, wherein we understand the nature of the graphs.

8.5.1 Task 1: Simple Subgraph Matching

We design two tasks where the subgraphs are easy to learn. The first task aims to show that RWK$^+$CN handles color-matching of every node pair along walks, while RWNN does not. The second task demonstrates that diversity regularization aids with learning non-overlapping hidden graphs. We report the matching accuracy for each experiment, where it is considered as a correct match when the model learns the desired subgraph pattern(s).

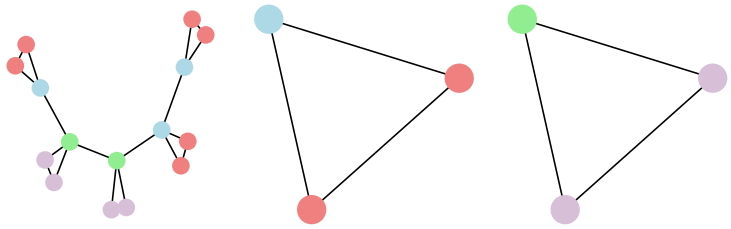
Task 1-1 We generate a database of 100 bipartite graphs with heterophily, where nodes on two sides of the graph have different colors/labels (e.g. Figure 8.1a). We use one hidden graph, and the task is to learn a bipartite core; "butterfly" (Figure 8.1b), or a 3-star with core and peripherals with different colors (Figure 8.1c). Two different objectives are used; one is to maximize the total similarities from all steps, and another one is to maximize the similarity only from the last step.

Table 8.3 reports the matching accuracies. Our RWK$^+$CN works well even if the similarity is only from the last step, regardless of the number of steps. Since RWK$^+$CN matches the labels of every node pair in each walk, maximizing the similarity from the last step needs to ensure the correctness of matching from previous steps at the same time. Although RWNN works when the similarity is from all steps, it fails when the similarity is from the last step when the number of steps equals 2. This is because the even-step neighbors in a bipartite heterophily graph have the same color.

However, this task is a special case, where the method only needs to realize that the neighbors should have the other color in the learned pattern. That is to say, RWNN still can not solve complicated cases just by summing up the similarity from all steps. As we will see later in this

Table 8.3: **Task 1-1: Simple subgraph matching in bipartite graphs.** Thanks to color-matching, RWK$^+$CN performs well even when the objective is based only on the last step.

Method	Objective	# of Steps	Acc.
RWNN	Sum of All Steps	2	26%
		3	100%
RWNN	Only Last Step	2	0%
		3	100%
RWK$^+$CN	Only Last Step	2	100%
		3	100%



(a) Chain (b) Pattern 1 (P1) (c) Pattern 2 (P2)

Figure 8.2: Task 1-2: Simple Subgraph Matching in Triangle Chain

section, while RWNN always learns rudimentary patterns because of ignoring the intermediate nodes in the walks, RWK$^+$CN learns more sophisticated ones by taking it into account.

In the rest of this section, we use "Sum of All Steps" as the objective for RWNN, and "Only Last Step" for RWK$^+$CN, which performs well and simplifies the optimization.

Task 1-2 To test diversity regularization, we generate a database with 100 node-labeled triangle chains, containing two frequent patterns (e.g. Figure 8.2a). Each triangle is either pattern P1 (Figure 8.2b) with probability 60% or otherwise P2 (Figure 8.2c) with lower frequency. The number of steps is set to 3, which is efficient and sufficient to capture both homophily (1-step) and heterophily (2-step) neighbors.

Table 8.4 reports the results, where accuracy depicts if *both* P1 and P2 are learned by the hidden graphs. Even without diversity regularization, RWK$^+$CN learns the most frequent pattern P1 with high accuracy. When diversity regularization is applied, accuracies for the second frequent pattern P2 and both patterns increase. The increase is larger when RWK$^+$CN is trained more flexibly with a larger number of hidden graphs to be learned. Notably, RWNN with vanilla RWK fails to learn either of the patterns. As it prefers the more frequent colors, it often learns all the node colors to be the same.

Table 8.4: **Task 1-2: Simple subgraph matching in triangle chains.** Diversity regularization helps RWK$^+$CN learn non-overlapping hidden graphs. Green (■) marks the winner.

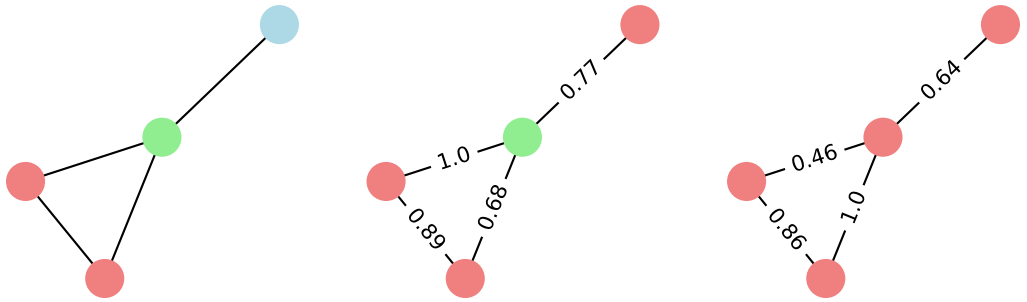
# of Hidden Graphs	Method	Diversity	P1 Acc.	P2 Acc.	Both Acc.
2	RWNN	No	0%	0%	0%
	RWK$^+$CN	No	82%	24%	12%
		Yes	72%	66%	44%
3	RWNN	No	0%	0%	0%
	RWK$^+$CN	No	88%	44%	32%
		Yes	76%	80%	62%
4	RWNN	No	0%	0%	0%
	RWK$^+$CN	No	98%	68%	66%
		Yes	84%	86%	74%

8.5.2 Task 2: GED-Based Evaluation

To further show the advantages of RWK$^+$CN, we design two more tasks each with two different testbeds. For evaluation, these experiments consider a database containing 100 identical graphs, which is used as the ground truth, i.e. only one hidden graph is used in both tasks. As the ground truth is more complex than the ones in Task 1, and it is difficult to learn the exact graph, we use graph edit distance (GED) [SF83] to measure how close the learned hidden graph is to the ground truth (the lower the better). While GED with node labels induces a penalty for editing the labels, GED without node labels purely focuses on the graph structure. The first task studies labeled graphs with different number of steps, and shows that RWK$^+$CN outperforms RWNN thanks to color-matching. The second task demonstrates the effectiveness of adding "structural colors", which improves both the learned structure and labels. We report p-values based on the paired t-test that quantify differences between two GED values statistically.

Task 2-1 We design two testbeds using node-labeled tailed triangles and rings, as shown in Figure 8.3a and 8.4a, respectively (best in color). We learn the hidden graph with the same number of nodes as the ground truth graph. Tables 8.3d and 8.4d report the GED comparison.

RWK$^+$CN achieves consistently lower GED than RWNN, demonstrating the importance of incorporating color-matching into the RWK. Experiments on both testbeds show that there is no clear choice for the number of steps, i.e., higher is not always better, where the p-values are high within RWK$^+$CN. We visualize the learned hidden graphs by removing the edges with the smallest edge weights. Figure 8.3b shows that RWK$^+$CN successfully assigns the green node with degree 3 in the correct position. Since the blue node only has degree 1, RWK$^+$CN reasonably learns to maximize the objective by adding one more red node in the hidden graph, which is the most frequent color; in Figure 8.3c, RWNN fails to handle the intermediate nodes, and hence includes only the most frequent color in the learned hidden graph. We observe a similar behavior in Figure 8.4b and 8.4c. While RWK$^+$CN pays much attention to learning the correct node labels, RWNN gives a rudimentary result, where all the nodes have the same labels.



(a) Colored Tailed Triangle: Ground Truth

(b) Hidden Graph Learned by RWK$^+$CN

(c) Hidden Graph Learned by RWNN

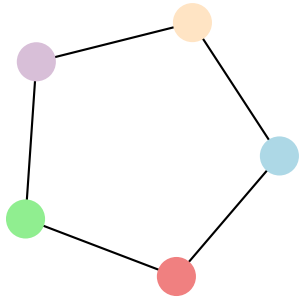
Method	# of Steps	GED w/ Node Labels	p-value
RWNN	2	3.35 ± 0.41	3.1e-05***
RWNN	4	3.15 ± 0.34	3.2e-03**
RWNN	6	3.25 ± 0.39	4.7e-04***
RWK$^+$CN	2	2.87 ± 0.58	0.20
RWK$^+$CN	4	2.82 ± 0.64	0.33
RWK$^+$CN	6	2.76 ± 0.86	-

(d) Table of results. Lower GED is better.

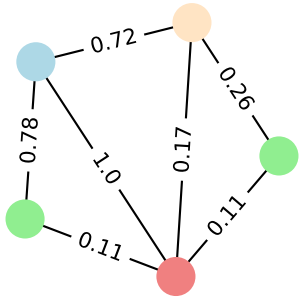
Figure 8.3: **Task 2-1: GED-based evaluation on tail-triangles.** Green (■) marks the winner.

Task 2-2 Two more testbeds are designed to evaluate structural colors. The number of steps is set to 3, which is effective and efficient. The first database contains 3-regular unlabeled graphs (Figure 8.5a). We evaluate RWK$^+$CN and RWNN by GED *without* node labels, focusing on the quality of the learned structure. Our assumption is, if the structural identifiers (node labels) are more unique, then the hidden graph can learn better graph structure. Therefore, we assume identity matrix as the best features in the evaluation, though it is not generalizable to the real datasets. We create the structural colors by a fixed and randomized Graph Attention Networks (GAT) [VCC$^+$18]. The results are reported in Table 8.5. Our proposed RWK$^+$CN using identity matrix as features receives the lowest GED without node labels, as expected. RWK$^+$CN using structural colors has competitive GED compared with using identity matrix, while being more generalizable. These results also empirically prove our assumption that using more unique identifiers as node features helps the hidden graph to learn better structure. In contrast, RWNN fails to utilize the features even if they are extremely informative.

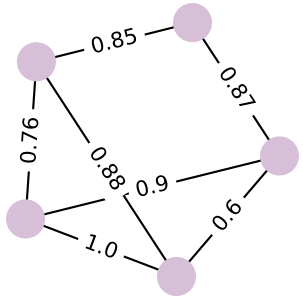
In the second testbed, we study a database containing 2-regular node-labeled graphs (i.e. 6-ring, Figure 8.5b), and report the results in Table 8.6. Only by replacing RWNN with our proposed RWK$^+$CN, the quality of learned hidden graph improves not only on labels, but also on the structure. In addition to the node labels, we further incorporate structural colors into RWK$^+$CN, and find the learned hidden graph improves even better, demonstrating the effec-



(a) Colored Ring: Ground Truth



(b) Hidden Graph Learned by RWK$^+$CN



(c) Hidden Graph Learned by RWNN

Method	# of Steps	GED w/ Node Labels	p-value
RWNN	2	7.08 $\pm$ 0.64	8.4e-14***
RWNN	4	7.16 $\pm$ 0.58	2.7e-14***
RWNN	6	6.92 $\pm$ 0.75	2.1e-11***
RWK$^+$CN	2	5.58 $\pm$ 0.65	0.21
RWK$^+$CN	4	5.59 $\pm$ 0.73	0.21
RWK$^+$CN	6	5.46 $\pm$ 0.86	-

(d) Table of results. Lower GED is better.

Figure 8.4: **Task 2-1: GED-based evaluation on rings.** Green (■) marks the winner.

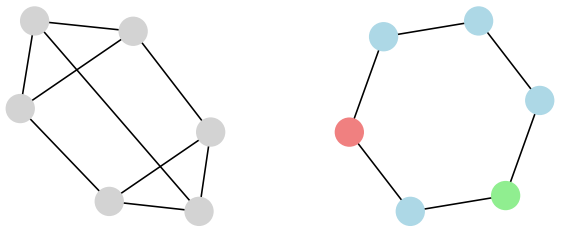
tiveness of structural colors. Notably, using identity matrix as features results in only slightly lower GED than using structural colors.

8.6 Experiments II: Adapting to Various Applications

In this section, the experiments is composed of two parts. In the first part, we demonstrate that different KCN architectures can perform better by employing our proposed RWK$^+$. In the second part, we compare our proposed RWK$^+$Conv with GCNConv, and empirically show that it has better expressiveness by extracting better graph features.

8.6.1 RWK$^+$: Employed to Different Architectures

We conduct three graph learning tasks for evaluating RWK$^+$. Since we focus on improving RWK across many tasks, rather than outperforming task-specific state-of-the-art methods, the experiments concentrate on comparing models using RWK versus RWK$^+$.



(a) 3-Regular Graph w/o Colors

(b) 2-Regular Graph w/ Colors

Figure 8.5: Task 2-2: Ground Truth Graphs in GED-Based Evaluation

Table 8.5: **Task 2-2: GED-based evaluation on 3-regular unlabeled graph.** Being used as unique identifiers of nodes, structural colors are shown to be as effective as identity matrix. Green (■) marks the winner.

Method	Additional Features	GED w/o Node Labels	p-value
RWNN	None	4.38 ± 0.65	-
RWNN	Identity	4.41 ± 0.56	0.57
RWK$^+$CN	Identity	3.89 ± 0.48	4.4e-05***
RWNN	SC	4.45 ± 0.70	0.72
RWK$^+$CN	SC	4.10 ± 0.50	0.010*

8.6.1.1 iGAD on Graph Anomaly Detection

Datasets We evaluate RWK$^+$ on supervised graph anomaly detection with 10 real-world datasets from PubChem [YCHY08], as in [ZYW+22]. Each graph is a chemical compound and labeled by its outcome from anti-cancer screen tests (active or inactive). The classes are highly imbalanced, where the ratio of the active samples is at most 12%, which are treated as the anomalous cases. We perform 5-fold cross-validation and split 10% of the training set as the validation set.

Settings iGAD [ZYW+22] incorporates RWK as a structural feature extractor to identify graph-level anomalies. For comparison we replace it with RWK$^+$ using StepNorm, with one-hot node degrees as the node features. Recall is used for both evaluation and model selection, as in the iGAD paper.

Results We report the average performance and standard deviation (stdev) in Table 8.7. iGAD with our proposed RWK$^+$ outperforms the original model on *all* datasets (p-val <0.001). This suggests that the hidden graphs learned through RWK$^+$ are consistently better than the ones extracted by RWK, assisting iGAD in better pointing out the anomalous graphs that deviate from these patterns.

Table 8.6: **Task 2-2: GED-based evaluation on 2-regular labeled graph.** Both techniques, color-matching and structural colors, improve the quality of structure and label learned by the hidden graph. Green (■) marks the winner.

Method	Additional Features	GED w/o Node Labels	p-value w/ Row 1	p-value w/ Row 2
RWNN	None	5.25 ± 0.64	-	-
RWK$^+$CN	None	5.02 ± 0.63	0.049*	-
RWK$^+$CN	Identity	4.81 ± 0.70	1.0e-03**	0.043*
RWK$^+$CN	SC	4.82 ± 0.70	1.1e-03**	0.043*

Method	Additional Features	GED w/ Node Labels	p-value w/ Row 1	p-value w/ Row 2
RWNN	None	7.25 ± 0.64	-	-
RWK$^+$CN	None	6.60 ± 0.93	1.8e-04***	-
RWK$^+$CN	Identity	6.12 ± 1.01	2.9e-08***	1.8e-03**
RWK$^+$CN	SC	6.25 ± 1.01	7.2e-07***	0.015*

Table 8.7: **Graph anomaly detection** on 10 real-world datasets. Recall is reported. iGAD using our RWK$^+$ as structural feature extractor outperforms original iGAD on all datasets. Green (■) marks the winner.

Dataset	MCF-7	MOLT-4	PC-3	SW-620	NCI-H23	OVCAR-8	P388	SF-295	SN12C	UACC-257
iGAD + RWK	75.1±1.1	74.1±0.8	77.9±1.2	78.6±0.9	78.7±1.3	78.8±0.3	83.1±1.7	78.3±1.1	79.4±0.6	78.0±1.0
iGAD + RWK$^+$	76.4±0.6	74.3±1.0	78.8±1.1	79.2±0.5	79.5±2.2	79.2±0.9	84.0±1.4	78.5±0.9	79.5±1.6	79.5±0.7

8.6.1.2 KerGNN on Substructure Counting

Datasets We evaluate RWK$^+$ on substructure counting with a simulated dataset [CCVB20] following the same setting in [ZJAS22], and the task is to predict the normalized count of substructures. This dataset includes four tasks, and the evaluation for different tasks are run separately. The dataset provides the training, validation, and testing sets with 1,500/1,000/2,500 graphs, respectively. One-hot node degrees are used as the node features.

Settings KerGNN [FYWT22] uses RWK to compare the similarity between the learnable hidden graphs and the egonets of nodes in a graph. The similarity from different learnable hidden graphs are used as the features for message passing. In this experiment, we replace RWK inside KerGNN with RWK$^+$. Mean absolute error (MAE) is used to measure the accuracy of counts.

Results As shown in Table 8.8, KerGNN with RWK$^+$ outperforms KerGNN with RWK in 3 out of 4 tasks. StepNorm is shown to effectively improve the performance of both methods by normalizing the similarity in each step. Without it, the similarity explodes after a number of steps, and is always dominated by the latest step. Although RWK performs better in counting

Table 8.8: **Substructure counting** on a simulated dataset. MAE is reported. RWK$^+$ wins in 3 out of 4 tasks, and StepNorm is shown to be effective. Green (■) marks the winner.

Task	Triangle	Tailed Tri.	Star	4-Cycle
KerGNN + RWK	0.1170	0.1346	0.1333	0.2153
KerGNN + RWK + StepNorm	0.1065	0.1251	0.0999	0.2140
KerGNN + RWK$^+$	0.1206	0.1246	0.1750	0.2078
KerGNN + RWK$^+$ + StepNorm	0.0802	0.1240	0.1312	0.1884

Table 8.9: **Graph classification** on 10 real-world datasets. Accuracy is reported. Although RWK$^+$ is built to capture descriptive features, it is competitive on most datasets. Green (■) marks the winner.

Dataset	MUTAG	D&D	NCI1	PROTEINS	MUTAGEN	TOX21	ENZYMES	IMBD-B	IMDB-M	REDDIT
KerGNN + RWK	81.9±5.3	75.2±1.5	71.6±2.6	75.3±1.2	74.4±2.4	89.1±0.3	47.3±3.9	71.2±2.1	48.1±2.9	77.2±0.5
KerGNN + RWK$^+$	83.0±6.4	74.8±2.4	72.3±1.3	76.2±1.2	75.1±1.0	89.2±0.3	44.0±2.7	71.6±1.0	49.2±0.6	77.5±0.6

stars, there are few paths to walk within a star, which decreases the necessity of adopting a kernel that is more accurate on similarity.

8.6.1.3 KerGNN on Graph Classification

Datasets We evaluate RWK$^+$ on graph classification with 10 real-world datasets from TU-Dataset [MKB+20], as in [FYWT22]. We use the node labels given by bio-informatics datasets (first 7), and the one-hot node degrees for the social interaction datasets (last 3). We perform 5-fold cross-validation and split 10% of training set as the validation set.

Settings Similar to substructure counting, KerGNN is used with RWK versus RWK$^+$. For fair comparison, StepNorm is employed for both models. We report average accuracy and stdev.

Results. Table 8.9 shows that although RWK$^+$ is designed with descriptive, structural graph features in mind, it offers competitive performance on most classification tasks (p-val < 0.1).

Scalability We verify RWK$^+$'s scalability by varying (1) the number of hidden graphs and (2) the number of steps on the NCI1 dataset, and report the runtime per epoch during training. In (1), the number of steps is 2, and in (2), the number of hidden graphs is 8. In Figure 8.6, KerGNN with RWK$^+$ is slightly slower than with RWK, though the overhead is negligible (< 1 second), and scales *linearly*, with significant speedup over regular color-matching computation.

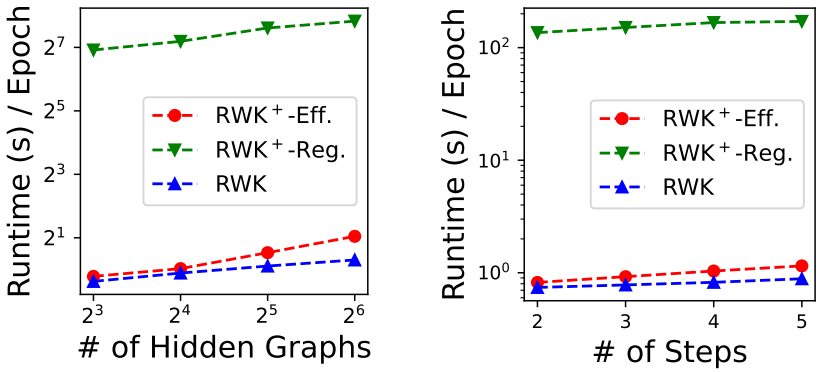


Figure 8.6: **Runtime of KerGNN with RWK$^+$** computed by regular Equation (8.8) versus efficient Equation (8.9), also compared to vanilla RWK.

Table 8.10: **Node classification** on 6 real-world datasets. RWK$^+$Conv wins and ties in most tasks on accuracy. Green (■) marks the winner.

Dataset	Cora	CiteSeer	PubMed	Cham.	Squirrel	Actor
GCNConv	85.8±0.7	73.4±0.5	88.0±0.2	69.7±0.9	55.7±0.4	28.3±0.6
RWK$^+$Conv	88.3±0.5	76.7±0.2	88.1±0.2	69.3±1.9	49.7±1.3	36.0±0.2

8.6.2 RWK$^+$Conv: Connections with GNNs

To show that RWK$^+$Conv is more expressive than GCNConv, a GCN message-passing layer [KW17], we compare them across both node- and graph-level tasks. In all experiments, we rigorously ensure that both kinds of layers share exactly the same message-passing backbone.

8.6.2.1 Node Classification

Datasets We evaluate RWK$^+$Conv on node classification with 6 datasets, including homophily graphs (first 3) [YCS16] and heterophily graphs (last 3) [RAS21, PWC$^+$20]. Each dataset is split into 60%/20%/20% for training, validation, and testing, respectively.

Results In Table 8.10, although expressiveness does not necessarily play a key role in achieving better accuracy on node-level tasks, RWK$^+$Conv still has competitive or better performance than GCNConv in most datasets.

Scalability We report the run time per epoch on the largest dataset PubMed in Table 8.11, where RWK$^+$Conv only creates negligible computational overhead (less than 0.05 second).

Table 8.11: **Runtime of RWK$^+$Conv**, with negligible overhead.

Step Length	**2**	**3**	**4**	**5**
GCNConv	0.0269	-	-	-
RWK$^+$Conv	0.0371	0.0464	0.0522	0.0619

Table 8.12: **Twitter bot detection** on a real-world web-scale social network. RWK$^+$Conv wins on F1-score. Green (■) marks the winner.

Dataset	**TwiBot-22**
GCNConv	53.7±0.2
RWK$^+$Conv	55.0±0.2

8.6.2.2 Twitter Bot Detection

Datasets We evaluate RWK$^+$Conv on a web application, namely bot detection in the TwiBot-22 dataset [FTW$^+$22]. This dataset contains a web-scale Twitter social network with one million users, where 86% of them are human, and the rest 14% are bots. We keep only the edges with types "followed" and "following", and make it undirected. The node features are embeddings of user descriptions, transformed by BERT [DCLT19]. The dataset provides the training, validation, and testing sets with 70%/20%/10% nodes, respectively.

Results In Table 8.12, RWK$^+$Conv outperforms GCNConv on F1-score. This suggests that RWK$^+$Conv has better ability to detect the bots by better utilizing the graph structure.

8.6.2.3 Graph Regression and Classification

Datasets We evaluate RWK$^+$Conv on graph regression and classification with three real-world datasets, ZINC [DJL$^+$23], ogbg-molhiv and ogbg-molpcba [HFZ$^+$20].

Results We include an additional baseline GINConv from GIN [XHLJ19]. In Table 8.13, we find that RWK$^+$Conv outperforms both baselines significantly across all datasets and tasks. This empirically demonstrates the better expressiveness of RWK$^+$Conv than GCNConv. It is also worth noting that, unlike other baselines, RWK$^+$Conv does not use edge features.

8.6.2.4 Summary and Future Work

All results strongly suggest the better expressiveness of our proposed RWK$^+$Conv, especially on graph-level tasks, and its connection to GCN motivates novel convolutional layers for better model design. This offers a direction with large potential to investigate further in the future. We also want to point out that the current design of the RWK$^+$Conv does not take edge features into consideration. Extending it to handle edge features by matching edge colors at every step of random walk could be a potential future work.

Table 8.13: **Graph regression and classification** on 3 real-world datasets. RWK$^+$Conv wins in all tasks. Green (■) marks the winner.

Dataset	**ZINC**	**ogbg-molhiv**	**ogbg-molpcba**
Metric	MAE ↓	ROC-AUC ↑	AP ↑
GCNConv	0.3258±0.0067	76.06±0.97	20.20±0.24
GINConv	0.2429±0.0033	77.78±1.30	22.66±0.28
RWK$^+$Conv	0.2082±0.0025	78.61±0.61	24.90±0.12

8.7 Conclusion

We first presented RWK$^+$, an improved random walk kernel with end-to-end learnable hidden graphs that can be used by various KCNs. RWK$^+$ incorporates color-matching along the walks that we showed can be efficiently computed in iterations, and combines similarities across steps in a learnable fashion. We then proposed RWK$^+$CN, a KCN that learns descriptive hidden graphs with an unsupervised objective and RWK$^+$. Thanks to additional "structural colors" and diversity regularization, it learns hidden graphs that better reflect the frequent and distinct graph patterns. Moreover, based on the mathematical connection of RWK$^+$ with GNNs, we propose a novel GNN layer RWK$^+$Conv, that extracts expressive graph features.

Experiments showed RWK$^+$'s descriptive learning ability on various unsupervised graph pattern mining tasks, as well as its advantages when employed within various KCN architectures on several supervised graph learning tasks. Furthermore, we showed that our proposed RWK$^+$Conv layer outperforms GCN, especially in the graph-level tasks by a large margin.

Chapter 9

DeltaShield: Information Theory for Human-Trafficking Detection

Chapter based on works that appeared at ICDE 2021 [LVK+21][PDF] and TKDD 2023 [VLK+23][PDF]*.

Given a million escort advertisements, how can we spot near-duplicates? Such micro-clusters of ads are usually signals of human trafficking. How can we summarize them to convince law enforcement to act? Spotting micro-clusters of near-duplicate documents is useful in multiple, additional settings, including spam-bot detection in Twitter ads, plagiarism, and more.

We present InfoShield, which makes the following contributions: (a) *Practical*, being scalable and effective on real data, (b) *Parameter-free and Principled*, requiring no user-defined parameters, (c) *Interpretable*, finding a document to be the cluster representative, highlighting all the common phrases, and automatically detecting "slots", i.e. phrases that differ in every document; and (d) *Generalizable*, beating or matching domain-specific methods in Twitter bot detection and human trafficking detection respectively, as well as being language-independent. Interpretability is particularly important for the anti human-trafficking domain, where law enforcement must visually inspect ads.

Our experiments on real data show that InfoShield correctly identifies Twitter bots with an F1 score over *90%* and detects human-trafficking ads with *84%* precision. Moreover, it is scalable, requiring only *8 hours* for *4 million* documents on a laptop. Our incremental version, DeltaShield, allows for fast, incremental updates, with minor loss of accuracy.

*These are joint works with Catalina Vajiac, who performed Sections 9.4.1 and 9.5.1, included here for clarity.

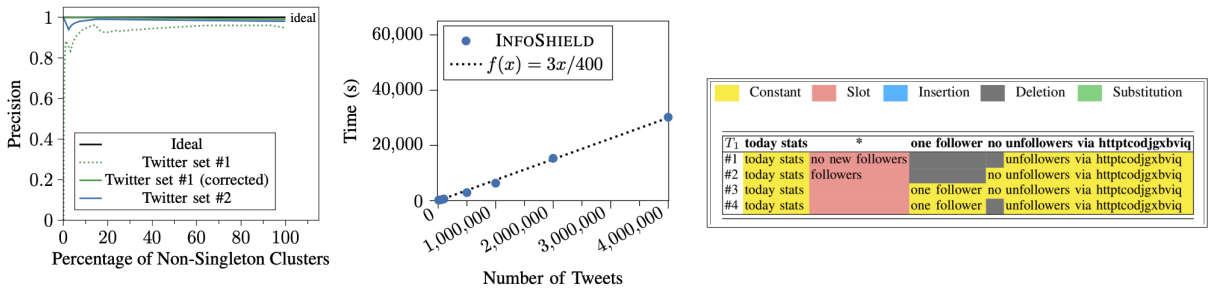


Figure 9.1: **INFOSHIELD is effective, scalable, and interpretable:** (left) precision@k on Twitter data is close to ideal, (middle) shows the scalability of INFOSHIELD over different data sizes, and (right) shows the interpretability of INFOSHIELD, detecting and visualizing slots (in red), i.e. portions of tweets that highly differ between otherwise duplicate documents.

9.1 Introduction

Given many documents, the majority of which do not belong to any cluster, how can we find small clusters of related documents? The driving application is human trafficking detection, where escort ads that are very similar are usually a sign of trafficking. Finding related documents is a problem with numerous applications, such as search engines, plagiarism detection, mailing-address de-dupliclation, and more.

We propose INFOSHIELD, a general, information-theory based method, and we illustrate its generality, effectiveness and scalability on two settings: escort advertisements, and Twitter data (both English as well as Spanish).

Application to the Human Trafficking Domain While INFOSHIELD is general, our main motivation is near-duplicate detection and summarization in escort advertisements. Human trafficking (HT) is a dangerous societal problem which is difficult to tackle. It is estimated that there are 24.9 million people trapped in forced labor, 55% of which are women and girls accounting for 99% of victims in the commercial sex industry [Off]. The majority of victims are advertised online and 56% of victims have no input on ad content [NAb]. The average pimp has 4-6 victims [NAa]. Thus, the majority of ads suspected of HT are written by one person, who is controlling ads for 4-6 different victims at a time. By looking for small clusters of ads that contain similar phrasing, rather than analyzing standalone ads, we are finding the groups of ads that are most likely to be organized activity, which is a strong signal of HT.

Currently, law enforcement looks for HT cases manually, often one at a time. Our proposed INFOSHIELD will help them save time by detecting micro-clusters of similar ads, grouping them, and summarizing the common parts, as shown in Figure 9.1, which depicts Twitter data – we refrain from showing escort ad results for the victims' safety.

Application to Twitter Bot Detection Detection of organized activity also has a clear application to bot detection; given millions of tweets, most of which come from legitimate users, how can we find tweets that exhibit bot-like behavior? The simplest kind of bot behavior is spam-

ming, i.e. posting tweets that are almost or exactly identical in text, to increase visibility. Bot detection has been well-studied, but the majority of algorithms use manually crafted features that are specific to certain platforms, for example, the number of retweets [DVF+16, CPP+16]. Our goal is to find near-duplicates in any application, which includes social media platforms containing text, such as Twitter. This particular application benefits from a vast amount of publicly available data.

Our Method Our first insight is to formalize the problem with information theory, and use the Minimum Description Length (MDL) principle to find good templates, which represent cluster text, with "slots", i.e. parts of the template that differ for each document. We mark slots with red highlights in Figure 9.1-bottom. We then use this summary to visualize the cluster. InfoShield is *parameter-free*, since MDL can automatically pick the best choice of parameter values for any algorithm by choosing the combination with the shortest compression length. This is the InfoShield-fine part of our method. The second insight is a novel preprocessing method, InfoShield-coarse, that dramatically improves scalability to be quasi-linear, by (a) eliminating single-copy documents/ads and (b) grouping the rest in coarse, but mainly homogeneous, clusters. Our algorithms, InfoShield and DeltaShield, have following desirable properties:

- **Practical**, being scalable and requiring no user-defined parameters thanks to the Minimum Description Language principle,
- **Interpretable**, providing a clear visualization and intuitive summarization of the discovered micro-clusters.
- **Generalizable** and *domain independent* – we show results on two diverse areas, namely, Twitter data, and HT data; as well as on multiple languages, i.e. Spanish, Italian, English.
- **Incremental**, processing new batches of documents on-the-fly without recomputing on historical documents.

A system diagram explaining the pipelines of InfoShield and DeltaShield is in Figure 9.2.
Reproducibility: Code available at `https://github.com/mengchillee/InfoShield` and `https://github.com/catvajiac/InfoShield-Incremental`.

9.2 Background and Related Work

There is a lot of work on HT detection, document clustering, and multiple sequence alignment, and we group it in the following sub-sections. In summary, while these methods have provided unique contributions, none have all of the same features as InfoShield and DeltaShield. Table 9.1 contrasts our proposed methods against the state of the art competitors.

9.2.1 Human Trafficking Detection

Some previous works try to classify whether or not a particular advertisement is suspected of HT [KDS+17, ASS17, TZJM17, ECP+19]. For instance, HTDN [TZJM17] proposes a supervised deep multimodal model trained on 10K manually labeled ads. Unfortunately, due to the adversarial nature of escort advertisements, these predefined or learned features do not stay relevant

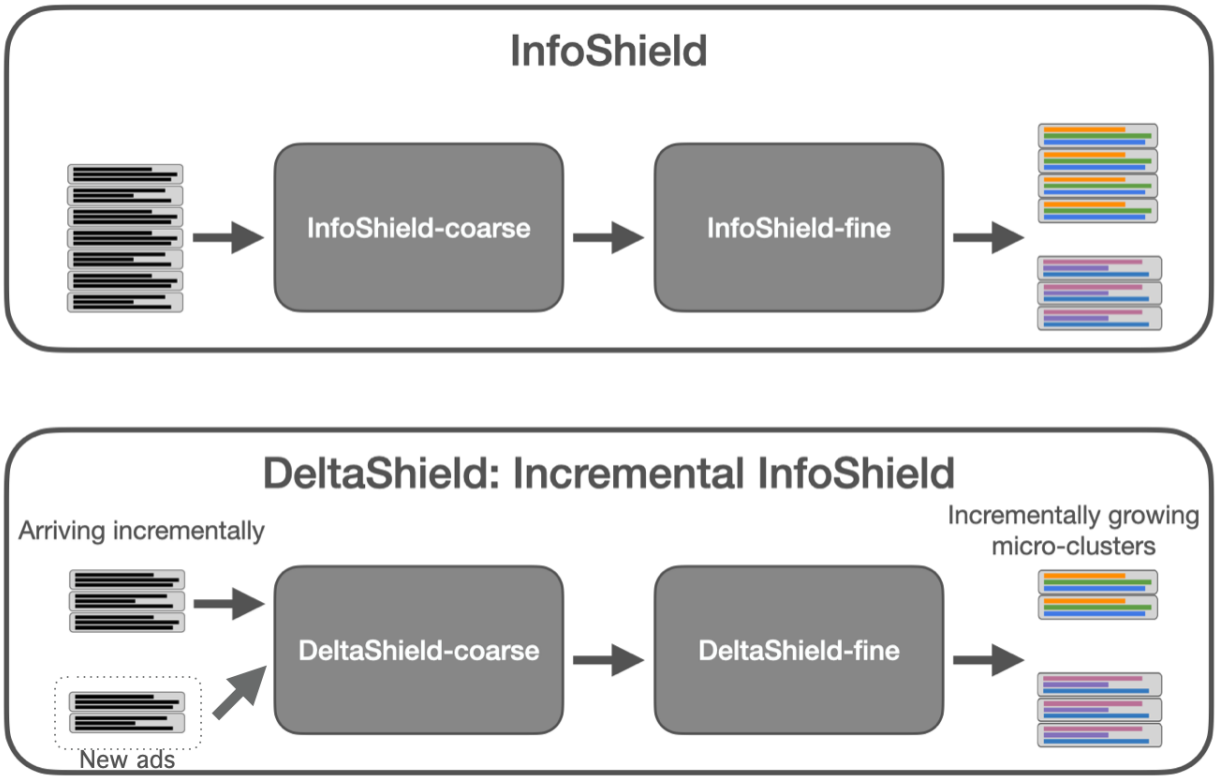


Figure 9.2: A system diagram of InfoShield and DeltaShield, showing the input, output, and intermediate steps.

over time. These labeled ads are also expensive to obtain (requiring the precious time of domain experts) and are error-prone (Section 9.6). Moreover, inspecting ads individually, we might overlook ads that are part of an organized activity but do not stand out on their own. Therefore, unsupervised algorithms that find connections between ads [NMBD17, PHD+17, RBD18] and *groups* of organized activity are preferred in this domain [LSL+18]. In particular, Template Matching [LSL+18] proposes the first anti-HT method to our knowledge to perform clustering. However, the interpretability of clusters is limited, and the algorithm is not scalable.

9.2.2 Social Media Bot Detection

Most efforts in detecting bots in social media platforms are formulated as supervised classification based on features from users and the content they post [SQJ+19, ZZ21]. Fewer works look for anomalies or fraud in networks, rather than in text, for instance [SLBF17]. A notable method, Botometer [DVF+16], formerly called BotOrNot, is an online service that provides a score of likelihood that a particular user is a bot. Since it is the only state-of-the-art method with public access to the implementation, we will use it as a baseline for our experiments in Section 9.6. [CPP+17] gives a more comprehensive overview of Twitter bot detection methods, and also provides the dataset we will use in Section 9.6. Very few works focus on detecting *organized activity* - groups working together to mislead people about who they are and what

Property \ Method	Clustering	Barzilay and Lee [BL03]	Shen et al. [SRPE06]	HDTN [TZJM17]	Template Matching [LSL+18]	InfoShield	DeltaShield
Practical - Scalable	?			✔	✔	✔	✔
Practical - Effective	?	?	?	?	✔	✔	✔
Practical - Ranked output	?			?	✔	✔	✔
Parameter-free	?					✔	✔
Principled						✔	✔
Interpretable	?	✔	✔		✔	✔	✔
Slot Detection		✔				✔	✔
Generalizable	✔					✔	✔
Incremental							✔

Table 9.1: **DeltaShield matches all specs**, while competitors miss one or more features. "?" indicates dependence on the clustering method or ambiguity in the original paper.

they are doing, which is a rising issue [Gle19]. ND-Sync [GCS+15] finds a related but different type of behavior, i.e. "retweet spam", where groups of multiple users exhibit organized behavior by consistently upvoting a particular user's tweets.

9.2.3 Document Embedding and Clustering

Much work has been done to represent documents in a machine-understandable format. The most widely-used approaches to represent documents include bag of words [Har54] and term frequency-inverse document frequency (tf-idf) [Jon04]. These methods are commonly used for plagiarism detection [KG16, MFHdC14, EK13, BDG95], which is a similar setting to near-duplicate detection. However, none of these methods do visualization or ranking, and some assumptions do not work in our case, i.e. [BDG95] assumes documents consist of multiple lines, which is not the case for tweets or the majority of escort advertisements.

Unsupervised word vector models such as Word2Vec [MCCD13], Doc2Vec [LM14], and FastText [BGJM17] assume that words occurring in the same context tend to have similar meaning, with much success. However, these methods require large amounts of time and data to train. Even when trained using large datasets from Twitter data and the HT domain, we find that these generic embedding methods do not perform as well (Section 9.6). Moreover, language model like BERT [DCLT19] does not perform well on escort ad text [Kul21], due to the sheer

number of misspellings, shortenings, and specific escort keywords not found in normal text. Instead, we take the approach of developing a lighter-weight solution that naturally handles the small amount of labeled data.

Given any document embedding, we can choose from many clustering algorithms. Density-based clustering techniques are most relevant to finding small dense text clusters, such as DBSCAN [EKSX96], HDBSCAN [MHA17], OPTICS [ABKS99], or k-means [DH04]. However, none of them do slot-detection. We compare InfoShield to HDBSCAN as part of our curated baseline for HT detection, see Section 9.6 for more details. In Table 9.1, we give several question-marks for clustering methods because some of the methods are scalable (k-means), while others are almost quadratic; some methods are parameter-free (G-means), but most are not.

Finding pairs of nearby points (or intersecting rectangles) is an old problem, under the name of "spatial joins" [LR94, BKS93]. However, these methods are best for low-dimensional spaces, since they use the R-tree [Gut84] spatial access method.

9.2.4 Multiple-Sequence Alignment

Multiple-Sequence Alignment (MSA) is a well-studied area with an application to biology, for comparing DNA sequences. The Barton-Sternberg algorithm [BS87] is an early profile-based approach which aligns sequences by updating a profile sequence iteratively. To resolve ambiguities among sequences caused by profile-based approaches, [LGS02] uses partial order graphs instead of profile sequences, which enables a base in dynamic programming to have multiple predecessors and successors. However, MSA focuses on finding the best alignment, which often requires arbitrary thresholds to determine the conserved regions within the sequences and ignores template complexity. In contrast, our goal is to automatically identify the best trade-off between template complexity and its ability to summarize the data.

Nature Language Processing (NLP) is another area benefiting from MSA. [BL03] applies MSA to learn the patterns of given word sequences by word lattices and rewrite the sentences. [SRPE06] focuses on aligning sentences by syntactic features to create the description for a particular fact. However, most of these methods highly rely on parameter tuning and English syntactical rules, assuming that all sentences are grammatically correct. This assumption does not hold for data on any social network or for escort advertisements, where misspellings and grammatical errors are common. Thus, these methods are not generalizable.

9.2.5 Minimum Description Length

The Minimum Description Length principle (MDL) [Ris78] assumes that the best model $M \in \mathbb{M}$ for data D minimizes $C(M) + C(D|M)$, where $C(x)$ is defined as the cost, i.e. number of bits, needed to describe x losslessly. The main insight is that it penalizes both the model cost $C(M)$, as well as the encoding of errors/deviations from the model $C(D|M)$ – while several other methods ignore the model complexity. MDL has been extremely successful in several data mining tasks [Grü07], including decision trees [MAR96], graph mining [CPMF04], time series segmentation and mining [MSF14], string similarity [KLR04], and many more applications. It formalizes the very intuitive "Occam's razor" idea: the simplest explanation for a phenomenon or dataset is the best explanation.

Doc#	Text
#1	Hi gentlemen, Korea super model just arrived...Alma and Joan specially selected...
#2	Hi gentlemen, Korea super model just arrived...Paula and Miya specially selected...
#3	Hi gentlemen, Korean super model just arrived...Paula specially selected...

Table 9.2: Simple Toy Example.

Doc#	Text
#4	Gentlemen, Korea super model just arrived...Miya is specially selected...
#5	I made 30K working on this job - call 123-456.7890 or visit scam.com
#6	I made 30K working from home - call 123-456.7890 or visit fraud.com
#7	Hello, Anna here! My hours are...

Table 9.3: Full Toy Example.

9.3 Proposed Method: Theory

In this section we present the theory behind our proposed method.

9.3.1 Intuition and Theory

Our problem is split into the following parts: Given N documents, where we suspect that there are small clusters of organized activity:

1. **Theory**: how do we measure the goodness of a set of clusters, and
2. **Algorithms**: how do we quickly find clusters that describe patterns in the data concisely (InfoShield-coarse – Section 9.4.1) and then how do we refine and visualize these clusters (InfoShield-fine – Section 9.4.2).

Suppose we have the documents of Table 9.2 in a particular cluster, which are a shortened version of escort ads InfoShield clustered into one template, how could we summarize them in a human-explainable form? One part of our proposed InfoShield is to use *templates*, which consist of constant strings and variable strings, called slots. We depict slots with '*', following the Unix convention. We also allow the usual string-editing operations (insertions, deletions, and substitutions). Thus, for the above 3-ad example, a human (and our InfoShield) would produce the template: "**This is a great *, and the * dollar price is great**", as shown in Table 9.4.

In addition to the documents in Table 9.2, suppose that we also have the documents in Table 9.3. Doc # 4 belongs in T_1, but with one deletion (omitting "a"), one insertion (adding "so") and one substitution (replacing "great" by "good"). However, Docs #5-7 clearly do not belong to the same template. We now would expect to see two templates T_1 and T_2, with T_1 representing Doc #1-4, T_2 representing Doc #5-6, and Doc #7 does not belong to any template. Furthermore, we would like to visualize the templates that we do find as in Table 9.4.

In more detail, but still informal, InfoShield should achieve lossless compression, with the cost being as follows:

Constant Slot Insertion Deletion Substitution

T_1	**Hi**	**gentlemen,**	**Korea**	**super model**	**just arrived...**	*	**specially selected...**
#1	Hi	gentlemen,	Korea	super model	just arrived...	Alma and Joan	specially selected...
#2	Hi	gentlemen,	Korea	super model	just arrived...	Paula and Miya	specially selected...
#3	Hi	gentlemen,	Korean	super model	just arrived...	Paula	specially selected...
#4		Gentlemen,	Korea	super model	just arrived...	Miya	specially selected...

T_2	**I made 30k working**	*	**- call**	*	**or visit**	*
#5	I made 30k working	on this job	- call	123-456.7890	or visit	scam.com
#6	I made 30k working	from home	- call	123-456.7890	or visit	fraud.com

Table 9.4: Templates for Full Toy Example.

Doc	Temp.	Slots	Ins.	Del.	Sub.
#1	T_1	{“Alma and Joan”}			
#2	T_1	{“Paula and Miya”}			
#3	T_1	{“Paula”}			3: “Korean”
#4	T_1	{“Miya”}		1	
#5	T_2	{“on this job", “scam.com"}			
#6	T_2	{“from home", “fraud.com"}			
#7	*N/A*	"Happy birthday to my dear friend Mike"			

Table 9.5: Example Encoding for $C(D|M)$.

1. *Model complexity* $C(M)$: the cost to encode the t templates we discover. In our working example, this would be the coding cost (roughly, the number of characters, below), for
   T_1: “Hi gentlemen, Korea super model just arrived... * specially selected...”
   T_2: “I made 30K working * - call * or visit *”
2. *Data compression* $C(D|M)$: the cost to encode slot-values, insertions, and deletions, for each of the documents, with respect to its best template (or just the listing of the words in the document, if no template matches). Thus, for each document, we must store (a) the tokens in slots, (b) position and token for insertions, (c) position for deletions, (d) position and token for substitutions, and (e) the template-id that best matches the document. Table 9.5 shows the information we include in $C(D|M)$ for our running example.

Notice that Docs #1-4 are compressed with much fewer characters when we use template T_1, since they have so many phrases in common. The coding cost is roughly proportional to the number of characters we need to describe (1) and (2) above. More formally, it is defined as:

Table 9.6: Table of Symbols and Definitions.

Symbol	Definition
N	Total number of documents in D
t	Total number of templates
V	Number of words in vocabulary
T_i	i-th template
l_i	Length of template T_i
s_i	Number of slots in T_i
$\hat{l}_d$	Alignment length of data d
$w_{d,j}$	Number of words in j-th slot in aligned data d
e_d	Number of unmatched words in aligned data d
u_d	Number of substituted/inserted words in aligned data d
$\langle n \rangle$	$\approx 2 \lg n + 1$: universal code length for a non-negative integer
$\lg(L)$	$= \log_2(L)$: code length for integer i $(1 \leq i \leq L)$

Table 9.7: Table of Acronyms.

Acronym	Definition
HT	Human Trafficking
MDL	Minimum Description Length
MSA	Multiple Sequence Alignment
POA	Partial Order Alignment
ES	Early-Stopping
ARI	Adjusted Rand Index

Definition 9.1: Total Encoding Cost

The total coding cost for a set of n documents with t templates is given by:

$$C = C(M) + C(D|M) \tag{9.1}$$

In Section 9.3.2, we explain the exact cost for N documents and t templates more precisely. Then, in Section 9.4, we propose algorithms on how to *discover* such a good set of templates.

We want to highlight that the separation of the cost function in Equation (9.1) from the algorithms makes InfoShield extensible: we can use any and every optimization algorithm we want. The ones we propose in Section 9.4 are carefully thought-out, and give meaningful results, but any other set of algorithms is fine to include – we can pick the solution with the best coding cost. Furthermore, InfoShield is parameter-free: any optimization algorithm minimizing total cost does not need user-defined parameters — we can try as many parameter values as we want, and pick the solution with the lowest cost.

9.3.2 Data Compression and Summarization

In this subsection, we give the details of the encoding cost in InfoShield. An overview of symbols and acronyms is provided in Table 9.6 and Table 9.7, respectively.

9.3.2.1 Template Encoding

We use the notation $\langle n \rangle$ for the coding cost of integer n, using the universal code length [Ris83], that is $\langle n \rangle = \log^* n \approx 2 \times \lg n + 1$. We also assume that we have V vocabulary words total and that each is encoded as an index, requiring $\lceil \lg V \rceil$ bits. For a length-l document, we need $\langle l \rangle$ bits to encode the number of words and $\lg V$ for each word, resulting the total cost $\langle l \rangle + l * \lg V$.

Definition 9.2: Model Encoding Cost

The coding cost for t templates is given by

$$C(M) = \langle t \rangle + \sum_{i=1}^{t} \langle l_i \rangle + l_i \lg V + (1 + s_i) \lg l_i \tag{9.2}$$

Let's describe every term of the above definition:

- $\langle t \rangle$ - universal coding, for the number of templates T
- For each template T_i, we need:
  - $\langle l_i \rangle$ to encode the number of words in the i-th template
  - $\lg V$ for each word in T_i
  - $\lg l_i$ for the number of slots s_i in the template, and
  - $\lg l_i$ for the location of each slot.

9.3.2.2 Alignment Encoding

Given a template and a document that it describes, what is the best way to encode the document? The intuition is to encode insertions, deletions, and substitutions in the template, and the tokens in slots. For the templates, we need only encode the word-location of a mismatch, its type, and, for insertion/substitution, we encode the relevant word.

Definition 9.3: Data Encoding Cost

The coding cost for N documents encoded with t templates is given by

$$\begin{aligned} C(D|M) = N &+ l_d \times \lg V \\ &+ \sum_{i=1}^{t} \sum_{d \in D_i} (\lg t + \left\langle \hat{l}_d \right\rangle + \hat{l}_d \\ &+ e_d \lg \hat{l}_d + u_d \lg V + \sum_{j=1}^{s_i} \mathcal{S}(w_{d,j})), \end{aligned} \tag{9.3}$$

where D_i denotes the data encoded by template T_i. In more detail, let D_U denotes the documents that do not match any template. The encoding cost for data $d \in D_U$ which is not encoded by template is simply computed by $l_d \times \lg V$. For the rest, the reasoning is as follows: Given a template T_i and a document $d \in D_i$, the alignment coding cost is:

- 1 bit for template flag yes/no
- $\lg t$ for template-id (if the flag is 'yes'):
- $\left\langle \hat{l}_d \right\rangle$ for length of the alignment
- 1 bit for each word in alignment if matched/unmatched

- $\lg \hat{l}_d$ for the location of each unmatched word
- $\lceil \lg 3 \rceil = 2$ bits for the operation type of each unmatched word (insertion, deletion, or substitution)
- $\lg V$ for word index in vocabulary if insertion/substitution
- $\mathcal{S}(w_{d,j})$ for the number of words $w_{d,j}$ in j-th slot:

$$\mathcal{S}(w_{d,j}) = 1 + \begin{cases} \langle w_{d,j} \rangle + w_{d,j} \lg V & \text{, if } w_{d,j} > 0 \\ 0 & \text{, otherwise} \end{cases} \tag{9.4}$$

- Repeat, for all other editing operations.

9.3.2.3 Overall Encoding

Notice that we ignored the cost of encoding the vocabulary, since it would be the same for all sets of templates, and roughly the number of bytes to spell out all the vocabulary words, separated by a word-delimiter, such as a newline character. More accurately, this would be: $\langle V \rangle + V \times (l+1) \times 8$ where l is the average word length, 8 bits per character, and 1 bit for the delimiter between words.

9.4 Proposed Method: Algorithms

How can we find templates that minimize our cost function in a scalable way? While the intuition described in Section 9.3 is correct, finding such templates is an expensive operation, being quadratic in the worst case. Thus, we first create reasonable clusters of related documents in a scalable way, using InfoShield-coarse, then work to find templates within each cluster using InfoShield-fine. If the average cluster size remains small, in comparison to N, then we process N documents in sub-quadratic time.

9.4.1 InfoShield-coarse

How do we quickly create coarse-grained clusters of documents with high text similarity? We start with document embedding, then perform clustering.

9.4.1.1 Document Embeddings

How do we generate a meaningful document embedding? We wish to capture similarity between documents that contain similar phrasing, but may have small variations (insertions, deletions, misspellings, etc). To this end, we first calculate the tf-idf weights for each phrase (n-gram)-document pair in the corpus. When calculating tf-idf, we consider phrases up to n-grams, with $n = 5$[†].

Then, for each document, we extract the top phrases with the highest tf-idf scores. By using tf-idf and limiting the number of phrases used, we only keep the most important phrases

[†]Phrase length has little impact on results past $n = 5$: see Section 9.6.3.

in the document that are unique to only a few advertisements, while ignoring commonly-used phrases. By making the number of phrases selected a function of input size, we reduce the risk of our results being heavily impacted by document length. Since some documents have a maximum length (i.e. tweets) but many do not, this helps to prevent InfoShield-coarse from being domain-specific.

9.4.1.2 Clustering

How do we quickly create meaningful candidate clusters? We construct a bi-partite graph of documents and phrases. For any document i and phrase j, we construct an edge i, j if j is a top phrase in i. Once all documents are processed, we consider all connected components in G to be our coarse-grained clusters.

In the case that these clusters end up too large (due to an "unimportant" phrase that combined documents that ideally should not be combined), we rely on InfoShield-fine to refine these clusters and split them if necessary. This is why InfoShield-coarse is very permissive, only requiring ads to share one important phrase to be connected. Algorithm 9.1 shows more formally how to construct a document graph using InfoShield-coarse.

Algorithm 9.1: InfoShield-coarse

Data: N documents
Result: candidate clusters generated from N
initialize empty document-phrase graph $G = (V_1, V_2, E)$;
forall *documents* d **do**
  forall *phrase* p *in FindTfidfPhrase(*d*)* **do**
    $E \leftarrow E \cup (d, p)$;
  end
end
clusters $\leftarrow$ FindConnectedComponents(G) ;

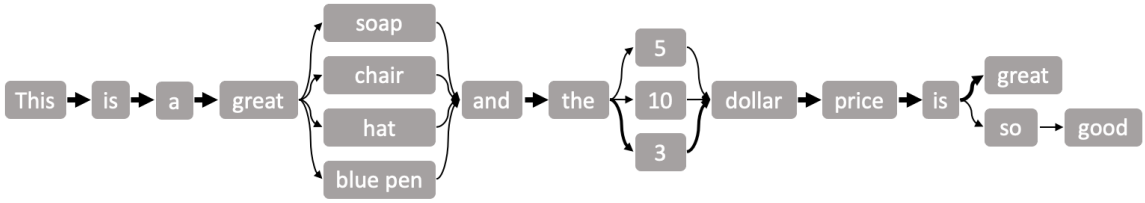
9.4.2 InfoShield-fine

Once we have coarse-grained clusters, how do we find templates and visualize the resulting clusters? Given data D containing multiple documents, split into coarse-grained clusters, the goal is to automatically find a template set M containing zero or more templates. Each template is expected to encode at least two documents. Within each coarse-grained cluster, the first task is to generate non-singleton candidate sets of documents and find potential templates. Next, we search for the best consensus document, i.e. the document that most represents the cluster, and detect possible slots by optimizing our cost function in Equation (9.3). We continue finding templates until we have processed all documents in a coarse-grained cluster, then move to the next cluster. We divide our algorithm into three major steps as follows:

1. **Candidate Alignment:** Identify the candidate set for a template and align all the documents in the set, using multiple sequence alignment (MSA).

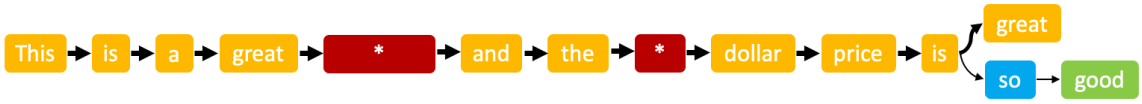
2. **Consensus Search:** Search for the best consensus document in the alignment.
3. **Slot Detection:** Detect slots in the consensus document to generate a template.

Let's take the first template from Table 9.4 as an example. We show a visual representation of what each step does in Figure 9.3.



(a) Step 1: Candidate Alignment

(b) Step 2: Consensus Search



(c) Step 3: Slot Detection

Figure 9.3: **Example pipeline of InfoShield-fine:** Here we show the output after each step of InfoShield-fine.

To compute the MSA, we carefully choose to use Partial Order Alignment (POA) [LGS02] as our alignment method for its effectiveness and efficiency. It is worth noting that InfoShield-fine can co-work with any off-the-shelf MSA approaches.

9.4.2.1 Candidate Alignment

Given data D from one cluster generated by InfoShield-coarse, containing multiple documents at iteration i, the candidate set for the template needs to be identified first. We first align all the documents $d \in D$ with the first document d_1 individually and then compute the cost $C(d|d_1)$ and $C(d)$ for every $d \in D$; if $C(d|d_1)$ is smaller than $C(d)$, meaning that d and d_1 have high similarity and can possibly be encoded by the same template, we add d into the set D_i containing all similar documents found in iteration i. Finally, we generate the alignment A_i by the POA method with all documents in D_i.

Algorithm 9.2: *Consensus-Search*

Data: An alignment A_i and a candidate set D_i
Result: A consensus document $T_i^{'}$

Initialize $h_L = 0, h_R = |D_i| - 1$;
while $h_L < h_R$ **do**
  $h_M \leftarrow (h_L + h_R)/2$;
  if $C(D_i|Sel(A_i, h_M - 1)) \leq C(D_i|Sel(A_i, h_M + 1))$ **then**
    $h_R \leftarrow h_M - 1$;
  else
    $h_L \leftarrow h_M + 1$;
  end
end
$T_i^{'} \leftarrow Sel(A_i, h_M)$;
Return $T_i^{'}$;

9.4.2.2 Consensus Search

After generating alignment A_i, how do we decide which tokens are part of the template, and which are insertions/deletions/substitutions? Keeping too many words in the template causes more unmatched operations (insertion/deletion/substitution); while keeping too few words hurts interpretability.

To solve this problem automatically, we turn it into an optimization problem by MDL. Function $Sel(A_i, h)$ is used to select the sub-alignment from the POA graph, where we only keep edges between words that occur more than h times in A_i. We aim to search for the best threshold h_i^* to generate the consensus of alignment with the lowest cost. The optimization problem can then be formed as follows:

$$h_i^* = \min_h C(D_i|Sel(A_i, h)) \tag{9.5}$$

Although our cost function is not convex, the optimization problem is only 1-dimensional, being relatively easy to solve. Hence we employ the Dichotomous Search algorithm [CZ04] as our optimization method, where it returns the optimal solutions in most cases. The optimization algorithm is shown in Algorithm 9.2, where we iteratively shrink the search space to half. The consensus document $T_i^{'}$ only contains one sequence and no slot.

9.4.2.3 Slot Detection

Once we have a template, how do we find slots? Slots contain parts of documents which we expect to differ, either in length or content, in the same location of each document. Slots inherently differ from unmatched words; instead of storing the location of each unmatched word per document as we would for unmatched words, we only store the location once, as part of the template.

Algorithm 9.3 shows how we do slot detection. We first recognize the operation types of words by each alignment $a \in A_i$, which are either insertions or substitutions. We identify

which words each potential slot p contains in the given consensus document T_i'. With this information, the computation of total cost with or without the slot p can easily be done. We only keep slots that decrease the total cost and store them in T_i.

9.4.2.4 Relative Length

To study the quality of compression by InfoShield-fine, we use relative length:

$$\text{Relative Length} = \frac{\text{Cost after compression}}{\text{Cost before compression}} \tag{9.6}$$

When relative length is close to 1, it means that the quality of compression is low; when it is close to lower bound, it means that the quality of compression is high, and the compressed documents are near-duplicate. For that reason, we derive the lower bound encoding cost of a cluster to study whether it is close to near-duplicate or not.

Lemma 9.1: Lower Bound of Encoding Cost

The lower bound encoding cost of a cluster by InfoShield-fine is:

$$\frac{t}{n} + \frac{1}{\lg V} \tag{9.7}$$

where t denotes the number of templates in the cluster, n denotes the number of documents in the cluster, and V denotes the number of words in vocabulary.

Proof. The encoding cost of n documents without a template is $nl \lg V$. In Equation (9.2), the encoding cost of t templates is $\langle t \rangle + t(\langle l \rangle + l \lg V + \lg l)$; and in Equation (9.3), the encoding cost for each document with no unmatched words is $(1 + \langle l \rangle + l)$. We can then derive:

$$\begin{aligned} &\frac{\langle t \rangle + t(\langle l \rangle + l \lg V + \lg l) + n(1 + \langle l \rangle + l)}{nl \lg V} \\ &\approx \frac{t \lg V + nl}{n \lg V} \approx \frac{t}{n} + \frac{1}{\lg V} \end{aligned} \tag{9.8}$$

where l is a small constant value that is negligible. The total encoding cost for n near-duplicate documents by t templates is approximately $\frac{t}{n} + \frac{1}{\lg V}$. ■

9.4.2.5 Overall Algorithm

The overall algorithm of InfoShield-fine is shown in Algorithm 9.4. Given data D containing multiple documents from one cluster by InfoShield-coarse, we first initialize the template set $\mathcal{T}$ and the number of detected template i. At iteration i, we initialize alignment by the first

Algorithm 9.3: *Slot-Detection*

Data: A consensus document $T_i^{'}$, an alignment A_i, and a candidate set D_i
Result: A template graph T_i with slot(s)

Initialize P as a dictionary, $T_i = T_i^{'}$;
for $a \in A_i$ **do**
    $x = 0$;
    for $j = 1, .., l_a$ **do**
        if a_j *is an insert or substitution word* **then**
            $P[x] \leftarrow P[x] + 1$;
        else
            /* a_j is a matched or deleted word */
            $x \leftarrow x + 1$;
        end
    end
end
for $p \in P$ **do**
    if $C(D_i|T_i^{'}(p.slot \leftarrow True)) < C(D_i|T_i^{'})$ **then**
        $T_i \leftarrow T_i(p.slot \leftarrow True)$;
    end
end
Return T_i;

document $d_0 \in D$. We compare with all other documents $d \in D$ to identify whether they should be encoded by the same template. After generating the alignment A_i and the data D_i that it encodes, we search for the best consensus sequence $T_i^{'}$ by optimizing the cost function. Then we detect the slots on the consensus sequence $T_i^{'}$ to generate template T_i. We include the T_i into our template set $\mathcal{T}$, and compute the total cost for both templates and data encoded by templates. If the total cost decreases by including T_i, we include it into $\mathcal{T}$ and update the total cost; otherwise, we treat D_i as noise. We run InfoShield-fine on every cluster generated by InfoShield-coarse, thus our final model M is $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \cdots \cup \mathcal{T}_m$, where m is the number of coarse clusters. It is worth noting again that InfoShield-fine is parameter-free, needing no human-defined parameters and optimizing for each template automatically.

9.4.3 Complexity Analysis

Algorithm 9.4: InfoShield-fine

Data: Data D consisting of multiple documents
Result: A template set $\mathcal{T}$

Initialize $\mathcal{T}, c^* = C(D), i = 1$;
while $|D| > 0$ **do**
  Initialize $A_i = d_1$ by the first document in D;
  Initialize candidate set D_i;
  for $d \in D[2:]$ **do**
    if $C(d|d_1) < C(d)$ **then**
      $D_i \leftarrow D_i \cup \{d\}$;
      $A_i \leftarrow MSA(A_i, d)$;
    end
  end
  $T_i^{'} \leftarrow ConsensusSearch(A_i, D_i)$;
  $T_i \leftarrow SlotDetection(T_i^{'}, A_i, D_i)$;
  $c \leftarrow C(\mathcal{T} \cup \{T_i\}) + C(D|\mathcal{T} \cup \{T_i\})$;
  if $c < c^*$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_i\}$;
    $c^* \leftarrow c, i \leftarrow i + 1$;
  else
    Treat D_i as noise(s);
  end
  $D \leftarrow D \backslash D_i$
end
Return $\mathcal{T}$;

Lemma 9.2

InfoShield is quasi-linear on the input size, taking time

$$O(Ncl) + O(k_{max} N log(N) l^2) \tag{9.9}$$

where N is the number of documents, l is the (maximum) length of a document, m is the number of coarse clusters, c is the maximum number of non-duplicate documents in a cluster, and k_{max} is the maximum number of templates in a coarse-grained cluster.

Proof. We analyze the runtimes of InfoShield-coarse and InfoShield-fine separately. For InfoShield-coarse, we iterate through N documents, picking the top 10% of phrases in N, and adding edges between these documents and phrases. Thus the runtime of InfoShield-coarse is $O(Nl)$, where l is the average length of the documents.

In InfoShield-fine, there are a total of k iterations, where k is the maximum number of templates generated from the given data. With the help of vectorization, MSA can be done

in $O(l^2)$. For each iteration, *Consensus-Search* requires $O(\log S' \times S' l^2)$ time, where S' is the average number of documents being aligned in each template; and *Slot-Detection* requires $S' l^2$ time. The time complexity of *Candidate-Alignment* in each iteration is $O(Sl^2)$, where $S \geq S'$ is the average number of documents in the each cluster. Thus the time complexity of InfoShield-fine is $O(\sum_{i=1}^{m} k_i S_i \log(S_i) l^2)$, which is upper-bounded by $O(k_{max} N \log(N) l^2)$, and where m is the number of coarse clusters generated by InfoShield-coarse, k_{max} is the maximum number of templates generated by a cluster.

In total, the algorithm takes time $O(Nl) + O(k_{max} N log(N) l^2)$ time. In practice, $k_{max} \leq 2$ in the Twitter datasets. Furthermore, the value of c will be quite low, since Twitter spambots post many duplicate tweets, which will make the runtime fast. Empirical evidence of this can be found in Figure 9.4, where we see that InfoShield-coarse scales linearly with input size. For the use cases presented in this paper, i.e. escort advertisements and tweets, we also note that l is bounded (280 for Tweets). ■

9.5 Proposed Method: Incremental

In order to do HT detection in practice, we must develop an algorithm that can process documents incrementally. Domain experts have hundreds of millions of ads and keep crawling additional ones each day. If we have already grouped historical ads into t clusters, we want to process a batch of newly-crawled documents without recomputing on historical documents. Here we will discuss the necessary modifications to InfoShield and present DeltaShield, with relevant experiments in Section 9.6.

9.5.1 DeltaShield-coarse

How can we modify InfoShield-coarse to be conducive to an online setting? We consider a setting where batches of documents come in during an aggregated time period, i.e. daily or weekly. Most of the algorithm can be adopted with minimal changes; since InfoShield-coarse incrementally adds to the document-term graph, we can process an entire batch, send the results to InfoShield-fine, and then continue processing documents as they arrive. The biggest challenge we have is in computing the tf-idf score of n-grams in a given document before seeing the entire corpus. To this end, we approximate the tf-idf score by computing the idf only on *the documents seen so far*, rather than the entire corpus. This approach is advantageous for two reasons: (1) as we process more and more documents, the approximate tf-idf score of a given n-gram will approach its actual tf-idf score, as verified empirically in Section 9.6.4, and (2) for the HT application, domain experts have a lot of historical, in-actionable data that can be processed first to improve the approximate tf-idf score.

9.5.2 DeltaShield-fine

In an online setting, we still need to generate new templates if needed, so Algorithm 9.4 will be performed in every batch. Moreover, a preprocessing step and an updating step must be included to keep DeltaShield efficient and effective.

Algorithm 9.5: Preprocess-Naive

Data: An incoming document d_1, and a template set $\mathcal{T}$
Result: An updated template set $\mathcal{T}$ or False if no appropriate template
// Examine all the templates
$i^* = \arg\min_{T_i \in \mathcal{T}} C(d_1|T_i)$;
if $C(d_1|T_{i^*}) < C(d_1)$ **then**
    $D_{i^*} \leftarrow D_{i^*} \cup d_i$;
    $T_{i^*}.added \leftarrow True$;
    // Find appropriate template
    // -> Continue with the next incoming document
    Return $\mathcal{T}$;
end
// No appropriate template
// -> Used as an initial document to generate the new template
Return False;

9.5.2.1 Preprocessing

We propose Algorithm 9.5 as a preprocessing step before generating a new template in Algorithm 9.4. If we were able to process all documents at once, the intuitive solution is to go through all the documents and generate templates. However, in an online setting, we often see documents from any one template span over multiple batches. To this end, the preprocessing step tries to encode an incoming document by all existing templates in its coarse cluster and select the template with the lowest encoding cost. If the cost by the selected template is lower than the encoding cost of the document itself, we consider the document to belong to that template.

Unfortunately, the time complexity of examining all the existing templates is $O(k_{max}l^2)$. If a coarse cluster has a large number of templates, we will incur a large overhead. To address this, we adopt an early-stopping (ES) mechanism in Algorithm 9.6. Instead of sequentially investigating all templates in a coarse cluster, we order the templates by the lengths of intersection between unigrams in the incoming document and each template. Then, we select the first template that lowers the encoding cost of the document.

Lemma 9.3

The time complexity of naive preprocessing step is $O(k_{max}l^2)$, but can be reduced by ES mechanism to $O(l^2 + k_{max}l)$, where l is the (maximum) length of a document, and k_{max} is the maximum number of templates in a coarse-grained cluster.

Algorithm 9.6: Preprocess-ES

Data: An incoming document d_1, and a template set $\mathcal{T}$
Result: An updated template set $\mathcal{T}$ or False if no appropriate template

$I_i \leftarrow |Intersection(d_1, T_i)|$ for all $T_i \in \mathcal{T}$;
$\mathcal{T}^* \leftarrow \mathcal{T}$ ordered by I;
for $T_{i^*} \in \mathcal{T}^*$ **do**
    // Early stopping
    if $C(d_1|T_{i^*}) < C(d_1)$ **then**
        $D_{i^*} \leftarrow D_{i^*} \cup d_i$;
        $T_{i^*}.added \leftarrow True$;
        // Find appropriate template
        // -> Continue with the next incoming document
        Return $\mathcal{T}$;
    end
end
// No appropriate template
// -> Used as an initial document to generate the new template
Return False;

Proof. To compute the cost after compression, we calculate the alignment, which takes $O(l^2)$. To search for the template with lowest encoding length, we examine $O(k_{max})$ templates. In total, the naive preprocessing step takes $O(k_{max}l^2)$ to find the template with lowest cost.

Next, we analyze the ES mechanism. The time complexity of computing the lengths of intersection for one template is $O(l)$. It takes $O(k_{max}l)$ to compute all the lengths for all templates in the coarse cluster. The total time complexity is then reduced to $cl^2 + k_{max}l$, where c denotes the number of templates examined. However, c is a small number in most cases (close to 1), which is negligible, so the final time complexity is $l^2 + k_{max}l$. ■

Later in Section 9.6.4.2, we will demonstrate that the ES mechanism largely improves the efficiency while achieving comparable effectiveness.

9.5.2.2 Template Update

Once the new documents are added into a template, its representation will be slightly changed. It is also important to update the template to represent new documents. Hence, we perform an updating step, Algorithm 9.7, right after Algorithm 9.4. It is worth noting that we only update templates that now represent any new documents. Furthermore, since changes in templates tend to be gradual over time, it is not necessary to process the updating step in every batch. We can set either a threshold (e.g. two hundred more documents) or an interval (e.g. one month) to trigger this step in order to improve the efficiency. We will demonstrate the trade-off between effectiveness and scalability using interval and threshold setting in Section 9.6.4.2.

Algorithm 9.7: *Template-Update*

Data: A template set $\mathcal{T}$ and data D
Result: An updated template set $\mathcal{T}$

```
for T_i ∈ 𝒯 do
    if T_i.added then
        A_i ← MSA(A_i, D_i);
        T_i' ← ConsensusSearch(A_i, D_i);
        T_i* ← SlotDetection(T_i', A_i, D_i);
        if C(D_i|T_i*) < C(D_i|T_i) then
            T_i ← T_i*;
        end
    end
end
Return 𝒯;
```

9.6 Experiments

We report experiments to answer the following questions:

RQ1. **Practical**: How fast is InfoShield, and how well does InfoShield work?
RQ2. **Interpretable**: How well does InfoShield visualize clusters? Are there any interesting results with respect to the relative length metric?
RQ3. **Robust**: How much does InfoShield-coarse change as we consider longer n-grams?
RQ4. **Incremental**: How does DeltaShield compare with InfoShield in terms of efficiency and effectiveness?

Dataset	Accounts	Tweets
genuine accounts	3,474	8,377,522
social spambots #1	991	1,610,176
social spambots #3	464	1,418,626
Test set #1 (spambots #1)	1,982	4,061,598
Test set #2 (spambots #3)	928	2,628,181

Table 9.8: Statistics for Twitter Bot Data.

Twitter Bot Data We use data from [CPP+17]. This data includes the tweet text and user id. The data is split into the following categories in Table 9.8. To create each test set, [CPP+17] sampled all tweets from 50% genuine accounts, and 50% from either social spambots #1 or social spambots #3. We use the provided test sets, which focus on social spambots only, so we can easily compare results to the best performing methods in [CPP+17]. This data not only contains binary labels as to whether particular tweets were posted from bots or legitimate users, but

also inherent clusters; i.e. user IDs that correspond to legitimate users or bots. We expect InfoShield to cluster most tweets from bots in clusters, ideally in one cluster per bot, and to have few clusters with legitimate users in them. With this intuition, we can create ground truth cluster labels in Twitter data as follows: (1) all legitimate users get labeled -1, since we assume their tweets are different enough that they shouldn't be clustered together; (2) all bots get labeled with their user ID.

Human Trafficking Data – Trafficking10k Dataset The Trafficking 10k dataset is created in [TZJM17], where expert annotators manually labeled 10,265 ads from 0-6. 0 represents "Not Trafficking", 3 represents "Unsure", and 6 represents "Trafficking".There are 6,551 ads labeled as not HT, 354 labeled as "Unsure", and 3,360 labeled as HT. Since the likelihood of an ad being HT is subjective, labeling is a difficult task. In fact, our analysis shows that 40% of exact duplicate ads (without any preprocessing) had label disagreement – i.e. multiple labels for the same exact text. Ads that are exact duplicates account for 12% of the dataset. We expect this labeling issue to occur for near duplicates as well. Therefore, we argue that looking at ads individually, whether manually or algorithmically, is a non-ideal way to find or to label HT cases. Despite the noisy labels, this is the only HT dataset to our knowledge with labeled data by human investigators. Thus, we use this dataset in our experiments, while being aware that noisy labels may impact results. This data does not have ground truth clusters. However, to create binary labels, we can call scores 0-3 as not HT, and 4-6 as HT.

Human Trafficking Data – Cluster Trafficking Cluster Trafficking is a new dataset provided by Marinus Analytics. This data contains cluster labels, provided by domain experts, for both HT, as well as for a strange behavior, that we will call 'escort spam':

Definition 9.4: Escort Spam

Script-generated advertisements that do not actually advertise real escort workers. The purpose of escort spam is not known, but it serves to confuse law enforcement.

We are given 6 spam clusters as well as ads from 96 massage parlors around the US. Cluster Trafficking consists of 157,258 ads, with 6,283 spam ads, 50,985 HT ads, and 99,990 normal ads.

Baselines Most state-of-the-art methods for HT detection are not open-sourced. Instead, we compare against HTDN [TZJM17], which uses the same Trafficking10k dataset, and develop three baselines using state-of-the art text embedding methods Word2Vec [MCCD13], FastText[BGJM17], and Doc2Vec [LM14]. We train all models using 1 million escort advertisements from the web. Then, we cluster using HDBSCAN [MHA17] with a minimum cluster size of 3. We call these methods Word2Vec-cl, Doc2Vec-cl, and FastText-cl.

On Twitter data, we compare to three supervised methods [YHG13, DVF+16, AA13] and one unsupervised method [CPP+16]. These methods all use Twitter-specific features that our domain-independent InfoShield does not use, such as number of mentions, favorites, retweets, posting time, etc. The unsupervised method is also not fully automatic, as a manually set thresh-

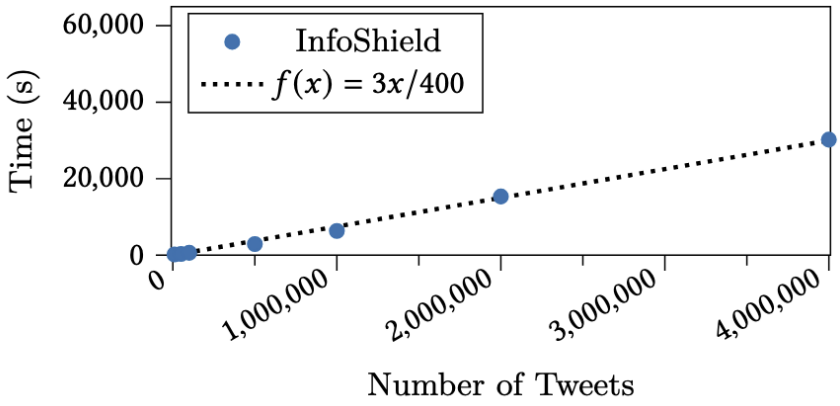


Figure 9.4: **InfoShield is scalable:** Linear on the input size; ≈ 8 hours for 4M tweets, on a stock laptop.

old discerns spam from legitimate tweets, which they change for each dataset. Regardless, InfoShield provides comparable results to these baselines.

Metrics For Twitter data, we have both binary labels and ground truth cluster labels. To compare binary labels, we can report precision, recall, and F1 score. For cluster labels, we use Adjusted Rand Index (ARI) [HA85]. We calculate precision, recall and F1 by calling all documents that ended up in templates to be suspicious, and all other documents as not suspicious.

9.6.1 RQ1 – Practical

How scalable is InfoShield? By using InfoShield-coarse to create coarse-grained clusters, and using the more expensive InfoShield-fine on smaller input sizes, we save time. We design an experiment on Twitter data by sampling Tweets the same way [CPP+17] did to create the test sets, and report the average runtime for each dataset out of five trials. The result is shown in Figure 9.4. Error bars were too small to be visible, so they were omitted.

How effective is InfoShield? We run InfoShield, as well as our developed baselines on both the Twitter data and Trafficking10k datasets. We report our results in Table 9.9, comparing against the two highest performing methods from [CPP+17].

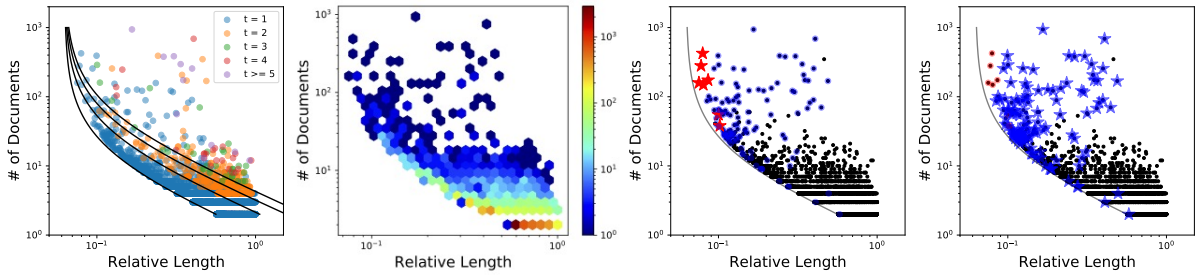
On Twitter data, InfoShield always performs within ten points of the top contender, despite using no features specific to Twitter such as retweets, favorites, or posting times.

For HT data, we see that InfoShield reports the highest precision; this is crucial since we want to avoid giving false positives to law enforcement at all costs. Law enforcement would rather know that they receive a real HT case (precision) than for all HT cases to be returned (recall) since they likely won't have time to pursue all cases. False positives cause law enforcement to lose trust in the algorithm and abandon it – as happened with previous applied solutions.

Twitter Data								
Dataset	Test Set #1				Test Set #2			
Metric	ARI	Prec.	Rec.	F1	ARI	Prec.	Rec.	F1
INFOSHIELD	83.2	93.0	91.2	92.1	75.7	96.7	88.9	92.6
Cresci [CPP+16]	n/a	98.2	97.2	97.7	n/a	100	85.8	92.3
BotOrNot [DVF+16]	n/a	47.1	20.8	28.9	n/a	63.5	**95.0**	76.1
Yang [YHG13]	n/a	56.3	17.0	26.1	n/a	72.7	40.9	52.4
Ahmed [AA13]	n/a	94.5	94.4	94.4	n/a	91.3	93.5	92.3

Human Trafficking Data							
Dataset	Trafficking10k			Cluster Trafficking			
Metric	Prec.	Rec.	F1	Prec.	Rec.	F1	ARI
INFOSHIELD	84.8	50.7	63.5	85.4	99.8	92.0	43.1
Word2Vec-cl	19.4	10.7	13.8	71.7	99.5	83.1	9.6
Doc2Vec-cl	25.6	10.9	15.3	74.2	98.8	84.7	16.2
FastText-cl	28.4	2 2.4	25.1	69.6	99.6	81.9	06.8
HTDN [TZJM17]	71.4	62.2	66.5	—	—	—	n/a

Table 9.9: **INFOSHIELD performs well**: INFOSHIELD beats or ties the best *domain-specific* method in both settings. Methods in red are supervised, while INFOSHIELD is unsupervised. Green (,) marks the top two and red ( ) denotes not applicable.



(a) Lower Bound (b) Heat Map of Clusters (c) Spam Clusters (red stars) (d) Human Trafficking Clusters (blue stars)

Figure 9.5: **Perpetrators seem separable**, thanks to our features: (a) shows all clusters (circles) and the lower bounds (black lines) – points are above the lower bound, as expected. (b) heatmap of the same: most points are close to the lower bound. (c) emphasizes the spam clusters as red stars, and (d) emphasizes the HT clusters as blue stars. Note that the majority of spam and HT clusters (red and blue stars) sit apart from the benign clusters.

Constant Slot Insertion Deletion Substitution

T_1		sismo	richter		km al sureste de puerto escondido oax lat lon pf km
#1		sismo	richter		km al sureste de puerto escondido oax lat lon pf km
Omit	21 Identical	Tweets	as #1 ...		
#23	sismologicomx	sismo	magnitud	loc	km al sureste de puerto escondido oax lat lon pf km

Table 9.10: **InfoShield is language-independent:** Spanish template from Twitter dataset.

Constant Slot Insertion Deletion Substitution

T_1	the	mostpopular	most	popular	stories on pr daily this week from	*	are	*
#1	the		most	popular	stories on pr daily this week from	instagram to mr t and perhaps even your grocers produce httptcokbfwdfts		
...								
#14	the		most	popular	stories on pr daily this week from	new cover photo rules on facebook and a battle of the soci httptcoeuetyugbku		
#15	the	mostpopular			stories on pr daily this week from	whimsical words to hillarys texts here	are	this weeks mos httptcoymwflapn
...								
#27	the	mostpopular			stories on pr daily this week from	understanding sopa to dating a pr professional here	are	the httptcoploce

Table 9.11: **InfoShield detects slots:** template from Twitter dataset.

9.6.2 RQ2 – Interpretable

How well does InfoShield visually interpret the clusters and templates we find? We show a few results of templates for Twitter data, and a censored version for the HT data, with discussion.

9.6.2.1 Twitter Data

As shown in Table 9.10, we find that 23 Spanish tweets are encoded by the given template. The first 22 ones are exact duplicates, but the last one contains three different words. InfoShield-fine automatically determines that representing those different terms as unmatched results, rather than as a slot, gives a smaller total cost. We can easily spot anomalies within clusters by using the template; the last tweet will have a lower compression rate than all other tweets.

In Table 9.11, we find that all the tweets are talking about the most popular weekly stories. While the first half of all tweets are almost identical, with minor syntax differences, the second half describes the particular stories, which all differ. InfoShield-fine then detects the second half of each sentence as a slot, which we expect to have different content in each tweet. This will help researchers pay attention to the most worth-studying parts.

9.6.2.2 HT Data

In Table 9.12, we show an example template from the HT domain. Unfortunately, we must censor the text to protect potential HT victims, so we only provide the highlighting from the template. For the slots, we give a description of the type of text they represent.

Notice that slots tend to include consistent user-specific information. For example, the second slot, if not empty, always discusses time. With a quick glance, a domain expert can easily find this data, rather than looking at a longer wall of text. For the HT domain, interpretability is key: law enforcement will only have to read one template, rather than each cluster member individually, to determine if this cluster is suspicious. The slots also contain messy data: i.e. while each slot has a specific purpose in Table 9.12, the text can be in multiple formats, i.e. "until

Constant Slot Insertion Deletion Substitution

T_1	**not shown for victim's**	**safety**						
#1	(empty)	(content)	time	(content)	(empty)	(content)	(empty)	(content)
#2	personal description	(content)	time	(content)	(empty)	(content)	cost	(content)
#3	(empty)	(content)	time	(content)	(empty)	(content)	cost	(content)
#4	personal description	(content)	(empty)	(content)	preferences	(content)	cost	(content)
...18	similar ads							

Table 9.12: **Slots contain user-specific information:** template from HT dataset.

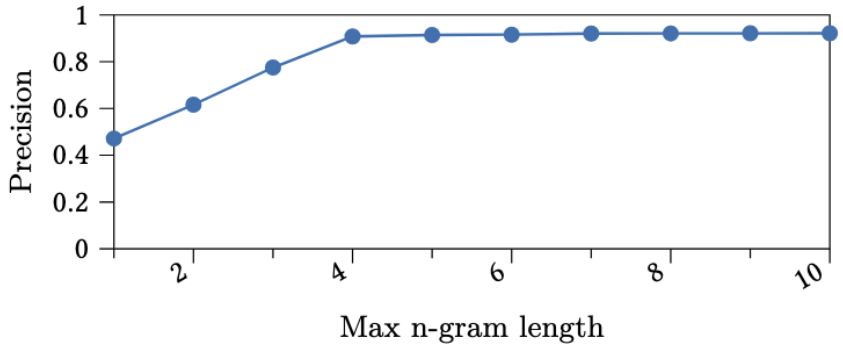


Figure 9.6: **5-grams are enough:** Precision stabilizes after $n = 4$.

9pm" vs. "9 P.M", etc. Work could be done to automatically extract and process the information within each slot, but this is beyond the scope of this paper.

9.6.2.3 Relative Length

Next, we consider the relative length, to further investigate the clusters detected by InfoShield. How does the relative length of a micro-cluster change as a function of the number of documents? Do we notice any differences between the relative lengths of spam clusters vs. HT clusters? Using the Cluster Trafficking dataset, we illustrate the lower bound of relative length versus number of documents per cluster in Figure 9.5a, where the black lines from left to right denote the lower bound of clusters with one to four templates. For example, the clusters with two templates (orange dots) cannot be on the left side of the second black line. As shown in Figure 9.5b, most clusters are concentrated by the lower bound, meaning that they do not have high numbers of documents. Further analysis surprisingly finds that spam and HT clusters follow patterns in this space. As shown in Figure 9.5c, most spam clusters (red stars) have small relative length with a high number of documents; in Figure 9.5d, there are two patterns of HT clusters (blue stars): (1) the near-duplicate clusters with a high number of documents (but slightly lower than spam clusters), (2) the outlier clusters that lie far from the lower bounds.

Twitter Data		
Dataset	Test Set #1	Test Set #2
ARI	99.1	99.9
HOM	99.9	99.9

Trafficking Data		
Dataset	Trafficking10k	Cluster Trafficking
ARI	97.1	99.2
HOM	96.1	96.5

Table 9.13: **DeltaShield-coarse is near-perfect:** we get almost exactly the same clustering for all datasets when processing ads incrementally.

9.6.3 RQ3 – Robust

How sensitive is InfoShield-coarse to the length of n-grams we use to calculate tfidf scores? We run an experiment on one of the datasets we used for our timing experiments, which contains 100,000 tweets by sampling all tweets from 50% legitimate accounts and 25% social spambot #1 accounts, and 25% social spambot #3 accounts. We detail the results in Figure 9.6.

9.6.4 RQ4 – Incremental

How does DeltaShield compare to InfoShield? We run experiments comparing the effectiveness and efficiency of these methods.

9.6.4.1 DeltaShield-coarse

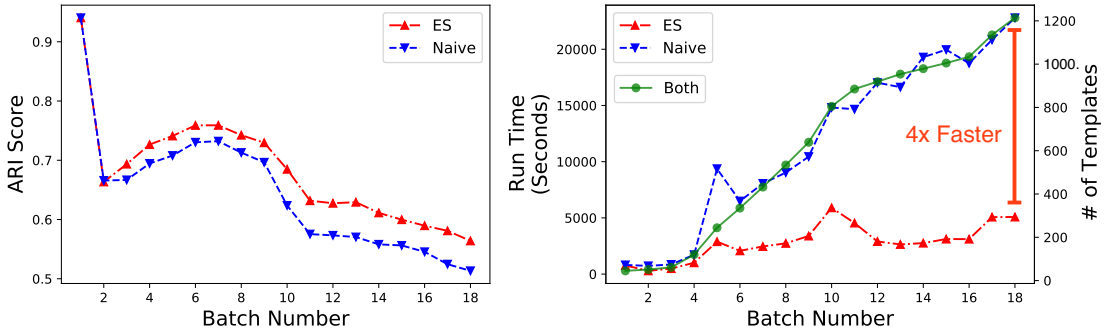
How do the document-term graphs generated by DeltaShield-coarse compare to the ones generated by InfoShield-coarse? The main difference between these algorithms is the approximation of tf-idf scores in DeltaShield-coarse. To measure the impact of this approximation, we compute the ARI and Homogeneity score (HOM) [RH07] between the cluster labels produced by DeltaShield-coarse and InfoShield-coarse. A high score signifies that the coarse clusters generated by DeltaShield-coarse are very close to the original coarse clusters generated by InfoShield-coarse. We run this experiment for both HT and both Twitter datasets, as shown in Table 9.13.

We see that all metrics are high, meaning that we do not lose much information by using DeltaShield-coarse and proccessing ads incrementally.

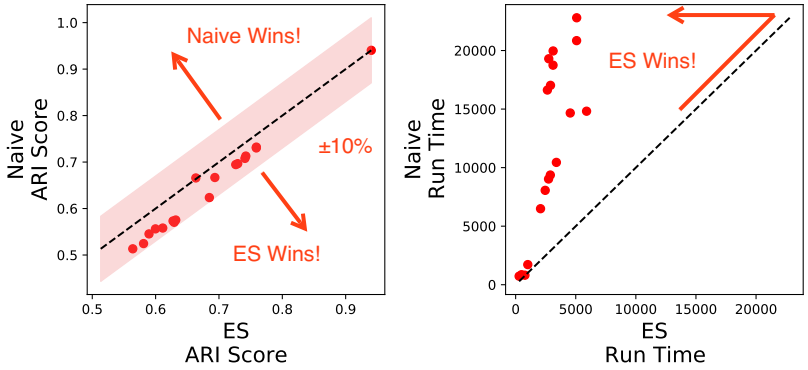
9.6.4.2 DeltaShield-fine

Here we compare the effectiveness and efficiency of DeltaShield-fine. We first compare Algorithm 9.5 (Naive) and Algorithm 9.6 (ES) to demonstrate that the ES method not only outperforms Naive one, but also dramatically decreases the running time. We then study the choice of update frequency, which results in a trade-off between effectiveness and efficiency.

We test an extreme case with the Cluster Trafficking dataset to truly reflect the difference between methods. The dataset is considered as a one big cluster and separated into 18 batches where each batch contains about 2000 advertisements. Note that InfoShield-coarse is not used in this experiment so that we can stress test DeltaShield-fine with a large number of



(a) ARI Score Over Time (b) Run Time and Number of Templates Over time

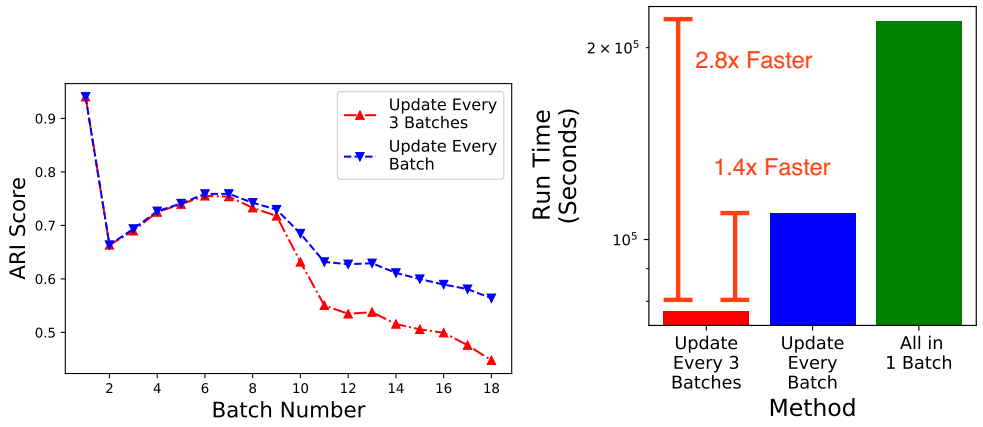


(c) ARI Score: ES vs. Naive (d) Run Time: ES vs. Naive

Figure 9.7: **DeltaShield-fine Preprocess-ES wins:** (a) shows the ARI score of Preprocess-ES remains higher compared to Preprocess-Naive over time. (b) demonstrates that run time is highly correlated with the number of templates, and shows Preprocess-ES is much faster. (c) and (d) show that Preprocess-ES outperforms Preprocess-Naive in terms of ARI score and run time respectively, where each data point denotes the result from one batch.

templates. Since our goal of incremental version is to output as close to the non-incremental one, ARI score is used as the effectiveness metric here, where the ground truth is clustering labels generated by InfoShield-fine. It is worth noting that this extreme case will not happen if InfoShield-coarse is still implemented, meaning the ARI score is expected to be low. Our empirical result shows that the average number of templates in one cluster generated by InfoShield-coarse is about 3, which is largely smaller than the number in our experiment (as shown in Figure 9.7b, it is already more than 200 after batch number 4).

ES vs. Naive The ARI scores over time are shown in Figure 9.7a, where we can find the ARI score of the ES method is always higher than the Naive method after the second batch. As depicted in Figure 9.7b, as the number of templates (green line) grows over time, the running time of fitting the templates increases linearly as well. If we compute the slope by the number of templates and running time, the slope of the Naive method is 18, while the one of the ES method is 3, which is 6 times smaller than the Naive method. In Figure 9.7c, the ES method



(a) ARI Score Over Time (1 batch = 20K ads) (b) Total Run Time

Figure 9.8: **DeltaShield-fine offers strong trade-off:** Even with large update frequency, the accuracy of DeltaShield-fine remains high. (a) shows large update frequency loses effectiveness increasingly over time. (b) shows increasing the update frequency leads to a much lower run time.

achieves a result with only 10% difference comparing to the Naive method, which is more less a tie. In Figure 9.7d, we find that the ES method always outperforms the Naive method more clearly in terms of run time.

Update Frequency Next, we study the trade-off between effectiveness and efficiency. We mainly compare DeltaShield-fine with update frequency every batch and every three batches. In Figure 9.8a, we find that as the number of incoming batches increases, the gap between two methods increases as well. Nevertheless, the running time of updating every three batches shown in Figure 9.8b is 1.4 times and 2.8 times faster than the one of updating every batch and InfoShield-fine, respectively. It will substantially mitigate the expensive overhead when the number of clusters and templates are large, which is especially important to the law enforcement where every second counts for them.

We notice that the low update frequency will slightly hurt the performance. However, we keep in mind that this experiment stress tests DeltaShield-fine, since the number of templates in one coarse cluster is much larger than it will be if we first use DeltaShield-coarse. Alternatively, an end user can consider doing the recomputation of all data periodically, depending on their idle time.

9.7 Discussion and Discoveries: InfoShield at Work

We note that InfoShield has the following advantages:

Advantage 9.1

InfoShield is *general*, using no language-specific or domain specific features.

In fact, the Twitter data includes tweets in Spanish, Italian, English, and Japanese, and we use no language-specific features in our methodology. In InfoShield-coarse, we automatically let tf-idf penalize common words, so there is no need to include stop-words in our algorithm. Note that the template in Table 9.10 is in Spanish, while the template in Table 9.11 is in English. This makes our method very powerful; it can be run on text in almost any language, or on other text data such as DNA strings.

Advantage 9.2

InfoShield is *extensible:* the goal of minimizing the total cost is separate from the algorithms we propose to do so.

In fact, one could replace InfoShield-coarse and InfoShield-fine with similar algorithms achieving the same end goal of pre-clustering and minimizing the total cost. We propose the algorithms above because they are scalable, and effective on real data.

Advantage 9.3

InfoShield does not require any user-defined parameters.

By using *Consensus-Search* to find the optimal algorithm, we remove the need for user-defined parameters in InfoShield-fine.

9.8 Conclusion

We presented InfoShield, which finds small clusters of near-duplicates in a collection of documents like escort ads for human trafficking detection, and visualizes the micro-clusters in a clear manner. The main contributions of the method are that it is:

- **Practical**, through scalability and using the MDL principle to be parameter-free,
- **Interpretable**, providing a clear visualization and summarization of clusters, and
- **Generalizable** and independent of domain (Twitter, HT), as well as of language (English, Spanish etc), and
- **Incremental**, by processing new documents on-the-fly, without having to recompute on historical documents.

Part III

Time Series Mining

Overview

Given time series data, how can we detect anomalous time periods?
How can we explain why they are anomalous by their group behavior in time series?

A time series data may contain anomalies that are not entirely random. For example, in an EEG recording, the signals during abnormal periods within a single seizure are similar to each other, but differ from those during normal periods. This suggests that, in addition to independent single-point outliers, anomalies can also exhibit group behavior. In this chapter, we aim to detect anomalous periods in time series data, as well as their group behavior, in order to explain why they are identified as anomalies.

We propose an **algorithm** to address this problem:

- § 10: TSAP detect sequence-level time series anomalies by optimizing hyperparameters to generate pseudo anomalies that capture the group behavior exhibited by true anomalies.

We extend this problem to a real-world time series **application**:

- § 11: In seizure detection, GEN^{2}OUT detects point-level time series anomalies by distinguishing groups of abnormal periods (seizures) from single-point outliers (noises).

Chapter 10

TSAP: Self-tuning Self-supervised Time Series Anomaly Detection

Chapter based on work that appeared at SDM 2025 [DLB+25] [PDF].

Time series anomaly detection (TSAD) finds many applications such as monitoring environmental sensors, industry KPIs, patient biomarkers, etc. A two-fold challenge for TSAD is a versatile and unsupervised model that can detect various *different types* of time series anomalies (spikes, discontinuities, trend shifts, etc.) *without any labeled data.* Modern neural networks have outstanding ability in modeling complex time series. Self-supervised models in particular tackle unsupervised TSAD by transforming the input via various augmentations to create pseudo anomalies for training. However, their performance is sensitive to the choice of augmentation, which is hard to choose in practice, while there exists no effort in the literature on data augmentation tuning for TSAD without labels.

In this chapter, we introduce TSAP for *time series anomaly "on autoPilot"*, which can *(self-)tune* augmentation hyperparameters end-to-end. It stands on two key components: a differentiable augmentation architecture and an unsupervised validation loss to effectively assess the alignment between augmentation type and anomaly type.

Case studies show TSAP's ability to effectively select the (discrete) augmentation type and associated (continuous) hyperparameters. In turn, it outperforms established baselines, including the state-of-the-art self-supervised models, on diverse TSAD tasks exhibiting different anomaly types.

10.1 Introduction

Time series are commonly observed in the real world, such as motion sensor signals [LP11, LHY+19, HLT+19], network traffic data [LPF10], and air quality measurements [CWL+18]. As one of the most important applications of time series, anomaly detection plays a key role in ensuring system safety and reliability, for example, by facilitating early warning [EFMN19]. Thus, there exists a large body of work on time series anomaly detection (TSAD) as presented in various surveys [BCML22, GGAH14].

Recent progress on self-supervised learning (SSL) offers significant improvements over traditional unsupervised (or one-class) learning approaches. The main advantage of SSL lies in its ability to self-generate labeled samples, i.e., *pseudo* anomalies, within the input space. This enables a focused exploration of a plausible subspace based on the semantics reflected in the pseudo anomalies, rather than an exhaustive, impractical search of the entire space. In SSL-based anomaly detection, data augmentation functions are used to create pseudo labels for the (self-)supervised training of an anomaly detector, such as by predicting whether the input is augmented or not [LSYP21], which augmentation function is used [GE18], or using contrastive learning [TMJS20]. In all these approaches, the success of SSL-based anomaly detection highly depends on the degree to which the augmented data mimics the true anomalies [YZA23b]. There exist few approaches for tuning data augmentation functions without labels, however, they have limitations. Some rely on non-differentiable validation losses [YZZA23], which are unsuitable for end-to-end learning frameworks. Others only address continuous hyperparameters, neglecting discrete ones, which are left for manual adjustment [YZA23a]. For instance, in the image domain, the CutOut augmentation cannot mimic semantic class anomalies (e.g. cats vs. cars), no matter how one tunes its (continuous) hyperparameters, mainly due to the mismatch in the discrete choice (i.e., augmentation type). Moreover, none of these efforts addresses anomaly detection for time series data, which is the focus of our work.

We introduce TSAP, a novel approach for SSL-based time series anomaly detection (TSAD) "on autoPilot" equipped with end-to-end hyperparameter tuning. Effectively tuning both discrete and continuous hyperparameters of augmentation enables TSAP to be an automated anomaly detector that is most suitable for a given task. TSAP stands on two main components: (i) a differentiable parameterized augmentation model and (ii) an unsupervised validation loss that measures the alignment between the augmented data and the unlabeled test data, quantifying the extent to which the former mimics the true anomalies.

We summarize our main contributions as follows:

1. **Problem:** Our work is the first attempt to tune both discrete and continuous hyperparameters of data augmentation in SSL-based TSAD, without labels at training time.
2. **Novel TSAD Method:** We propose TSAP, which accommodates various types of time series anomaly and enables automatic tuning of related hyperparameters (magnitude, duration, etc.) with a differentiable validation loss quantifying alignment of augmented and unlabeled test data.
3. **Effectiveness:** By carefully selecting augmentation type and its hyperparameters, our self-tuning TSAP outperforms existing unsupervised and self-supervised approaches, including the state-of-the-art baseline which also employs learnable augmentations, across diverse TSAD tasks.

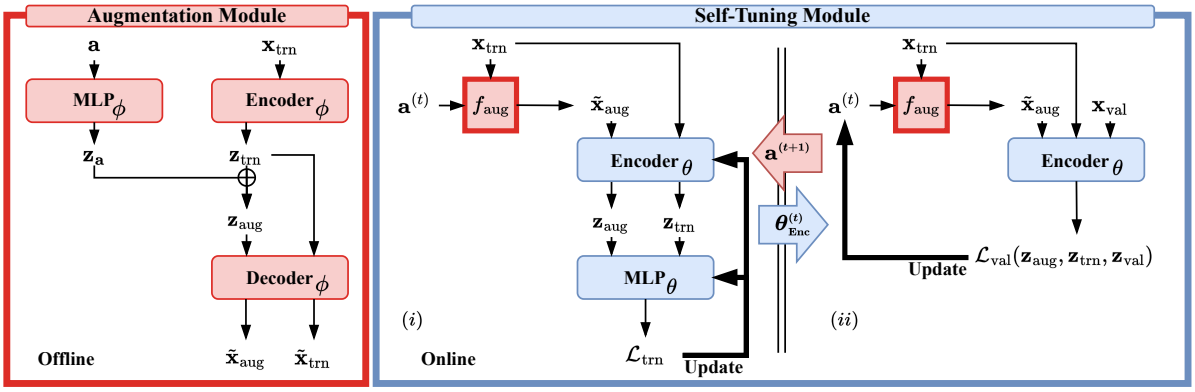


Figure 10.1: Our TSAP framework for end-to-end self-tuning TSAD. **Left:** Offline trained, differentiable augmentation model $f_{\text{aug}}(\cdot;\phi)$ takes as input the normal data and augmentation hyperparameter(s) $\mathbf{a}$, and outputs pseudo-anomalies $\tilde{\mathbf{x}}_{\text{aug}}$. **Right:** Self-tuning engine incorporates the pre-trained f_{aug} (with parameters ϕ frozen), alternating between two phases: (i) detection phase – given $\mathbf{a}^{(t)}$ at iteration t, estimate parameters $\boldsymbol{\theta}^{(t)}$ of detector f_{det} (consisting of Encoder$_\theta$ and discriminator MLP$_\theta$), by optimizing $\mathcal{L}_{\text{trn}}$ (classification loss); (ii) alignment phase – given $f_{\text{det}}^{\text{enc}}(\cdot;\boldsymbol{\theta}^{(t)})$, update augmentation (governed by $\mathbf{a}$) to better align the embedded time series $\mathbf{z}_{\text{trn}} \cup \mathbf{z}_{\text{aug}}$ with $\mathbf{z}_{\text{val}}$ in the learned discriminative space. $\mathbf{x}_{\text{val}}$ contains both normal and anomalous time series, but labels are *not* known or used at any point during training time.

Reproducibility: All code and datasets used in this work are available at the following repository: `https://github.com/B-Deforce/TSA-on-autoPilot`.

10.2 Background

An overview of symbols and acronyms is provided in Table 10.1 and Table 10.2, respectively.

10.2.1 Time Series Anomaly Detection

This work focuses on *time series anomaly detection* (TSAD). Consider a univariate time series $\mathbf{x} = \{x_1, x_2, \ldots, x_K\}$, where $\mathbf{x}$ is a sequentially ordered collection of K data points. Each $x_k \in \mathbb{R}$ corresponds to a scalar observation at each time step. Then, let $\mathcal{D}_{\text{trn}}$ be a set of training data containing only normal time series, and $\mathcal{D}_{\text{test}}$ be a set of unlabeled test data containing both normal and anomalous time series. The problem is defined as follows:

- **Given** $\mathcal{D}_{trn}$ containing only normal time series, and $\mathcal{D}_{test}$ containing both normal and anomalous time series without labels.
- **Predict** the label $y_i \in \{-1, +1\}$ for each time series $\mathbf{x}_i \in \mathcal{D}_{test}$, where $y_i = +1$ denotes there is at least one anomaly in $\mathbf{x}_i$.

Table 10.1: Table of Symbols and Definitions.

Symbol	Definition
$\mathbf{x}$	Univariate time series
$\mathcal{D}_{\text{trn}}$	Training data with only normal time series
$\mathcal{D}_{\text{aug}}$	Augmented data with time series with pseudo anomalies
$\mathcal{D}_{\text{test}}$	Testing data with both normal and anomalous time series
$\mathcal{D}_{\text{val}}$	Part of unlabeled $\mathcal{D}_{\text{test}}$ for validation
$\mathbf{a}$	Set of hyperparameters for augmentation
$\mathcal{A}^P$	Domain of possible hyperparameter value
$g(\mathbf{x}, \mathbf{a})$	Anomaly generation scheme based on $\mathbf{a}$ (no parameters)
$f_{\text{aug}}(\mathbf{x}, \mathbf{a}; \boldsymbol{\phi})$	Differential data augmentation function with parameters $\boldsymbol{\phi}$
$f_{\text{det}}(\mathbf{x}; \boldsymbol{\theta})$	Time series anomaly detector with parameters $\boldsymbol{\theta}$
$\mathcal{Z}$	Set of time series embeddings

Table 10.2: Table of Acronyms.

Acronym	Definition
TSAD	Time Series Anomaly Detection
SSL	Self-Supervised Learning
AUC	Area Under Curve
IF	Isolation Forest

10.2.2 Self-Supervised Anomaly Detectors

Let $f_{\text{aug}}(\cdot; \boldsymbol{\phi})$ be a data augmentation function, such as rotation or cut-out in the image domain (a discrete choice), that outputs *pseudo* anomalies. A given f_{aug} also exhibits continuous hyperparameter(s), such as angle (for rotation) or width and height (for cutout). Then, a popular approach [LSYP21] is to train a detector model $f_{\text{det}}(\cdot; \boldsymbol{\theta})$ to classify between normal data $\mathcal{D}_{\text{trn}}$ and augmented data $\mathcal{D}_{\text{aug}} = \{f_{\text{aug}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{D}_{\text{trn}}\}$. The working assumption is for f_{det} to predict the (unknown) anomalies in test data as *augmented*. As such, performance relies on how well the choice of augmentation type (discrete) as well as the choice(s) of its (continuous) hyperparameter(s) mimic the anomalies in $\mathcal{D}_{\text{test}}$. There are various other ways of using SSL for anomaly detection, based on the specific objective function and how the pseudo labels are generated [GE18, TMJS20], which are similar to each other in principle.

10.2.3 Data Augmentation on Time Series

To apply SSL to time series data, we need an augmentation function specifically designed for time series. For a given time series $\mathbf{x} \in \mathcal{D}_{\text{trn}}$, $f_{\text{aug}}(\mathbf{x}; \mathbf{a})$ transforms $\mathbf{x}$ based on hyperparameters $\mathbf{a} \in \mathcal{A}^P$, $P \geq 1$, where $\mathcal{A}^P$ represents the domain of possible hyperparameter values. For time series data, augmentations could involve mean shifts, trend injections, or other transformations relevant to the time-series domain, with $\mathcal{A}^P$ enclosing the respective hyperparameter space for a given transformation. We describe augmentation in more detail in Section 10.3.1.

10.2.4 Wasserstein Distance

The Wasserstein distance [ACB17], a distance measure between probability distributions, of order p between any two marginal distributions μ and ν is given as:

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu, \nu)} \mathbb{E}_{(x,y)\sim\gamma}[d(x, y)^p] \right)^{1/p}, \tag{10.1}$$

where $\Gamma(\mu, \nu)$ is the set of joint distributions μ and ν, and d is the distance function. That is, γ satisfies two conditions: $\int \gamma(x, y)dy = \mu(x)$ and $\int \gamma(x, y)dx = \nu(y)$. When $p = 1$, it

is also known as the earth mover's distance. We use it to measure the distance between two distributions of embeddings, since it can effectively handle distributions that are not overlapped, unlike KL divergence. We employ the Sinkhorn algorithm to feasibly apply the Wasserstein distance, which approximates it efficiently via entropy regularization [Cut13].

10.3 Proposed Method: TSAP

Our problem is defined as follows:

Problem 10.1: Self-Supervised Based Time Series Anomaly Detection

Given $\mathcal{D}_{\text{trn}}$ and $\mathcal{D}_{\text{test}}$, find augmentation hyperparameters $\mathbf{a}^*$ that generate pseudo anomalies $\mathcal{D}_{\text{aug}}$ such that $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ is best aligned with $\mathcal{D}_{\text{test}}$, and train the anomaly detector $\boldsymbol{\theta}^*$ using $\mathcal{D}_{\text{trn}}$ and $\mathcal{D}_{\text{aug}}$.

There are two notable challenges that need to be addressed for automatic selection of both discrete and continuous hyperparameters for SSL-based TSAD:

- **C1. Differentiable Augmentation:** Developing an augmentation function that is differentiable with respect to its hyperparameters, enabling gradient-based optimization.
- **C2. Comparable Validation Loss:** Formulating a validation loss that effectively quantifies alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{test}}$ while being comparable *across* different hyperparameter initializations.

Our framework tackles **C1** and **C2** with two key modules:

- **M1. Differentiable Augmentation Module:** TSAP implements the augmentation function as an Encoder-Decoder neural network, f_{aug} parameterized by $\boldsymbol{\phi}$, capable of approximating the anomaly-generating mechanism conditioned on $\mathbf{a} \in \mathcal{A}^P$. Importantly, this module is pre-trained independently and prior to the initiation of **M2**, establishing it as an *offline* component of the framework.
- **M2. Self-Tuning Module:** At test time *online*, TSAP iteratively refines the detector f_{det}'s parameters $\boldsymbol{\theta}$ as well as augmentation hyperparameters $\mathbf{a}$, through alternating detection and alignment phases. Alignment is performed on part of the unlabeled $\mathcal{D}_{\text{test}}$, referred to as $\mathcal{D}_{\text{val}}$.

Based on **M1** and **M2**, we propose TSAP, a self-tuning self-supervised TSAD framework demonstrated in Figure 10.1.

10.3.1 Module 1: Differentiable Augmentation Module

Anomaly Injection Scheme We accommodate six types of anomalies that are common in real-world time series; namely, trend, extremum, amplitude, mean shift, frequency shift, and platform. For brevity, we illustrate them by examples in Figure 10.2 with red lines. Each anomaly type has three hyperparameters; including its starting position (`location`), duration (`length`), and severity (`level`) as shown in Figure 10.2. Extremum captures a spike and only has two hyperparameters as its duration is always 1.

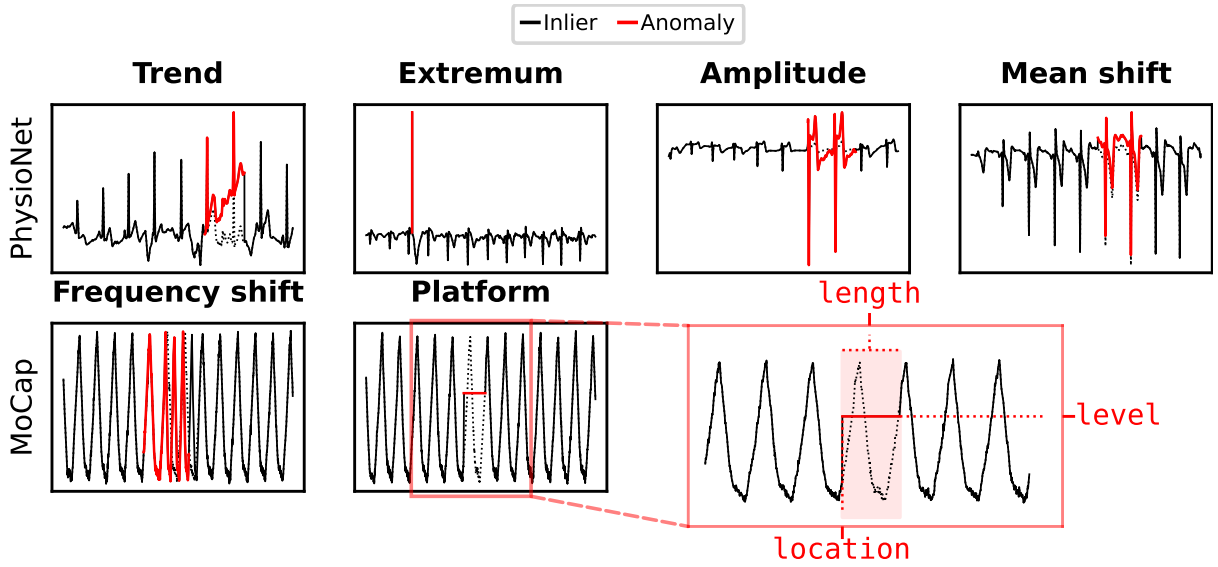


Figure 10.2: Examples of six different types of time series anomalies; (black) original real-world time series, (red) pseudo anomalies generated by g.

Based on $\mathcal{D}_{\text{trn}}$, the anomaly generation scheme g creates an augmented dataset $\mathcal{D}_{\text{aug}} = \{g(\mathbf{x}_{\text{trn}}; \mathbf{a}) \mid \mathbf{x}_{\text{trn}} \in \mathcal{D}_{\text{trn}}, \mathbf{a} \sim \mathcal{A}^P\}$, where $\mathbf{a}$ is the vector of augmentation hyperparameters, uniformly randomly sampled from the domain $\mathcal{A}^P$.

Model Design To build an augmentation model f_{aug} for time series, we use a convolutional neural network (CNN) to encode the input time series $\mathbf{x}_{\text{trn}}$ into the feature map $\mathbf{z}_{\text{trn}}$. We then encode the augmentation hyperparameters $\mathbf{a}$ into $\mathbf{z}_{\text{a}}$, which has the same shape as $\mathbf{z}_{\text{trn}}$, with a multilayer perceptron (MLP). Since the feature map $\mathbf{z}_{\text{trn}}$ generated by the CNN encoder keeps the positional information of the original time series, adding $\mathbf{z}_{\text{a}}$ to $\mathbf{z}_{\text{trn}}$ ensures that only the part with the desired `location` and `length` in $\mathbf{z}_{\text{aug}}$ is manipulated. To ensure that the feature maps $\mathbf{z}_{\text{aug}}$ and $\mathbf{z}_{\text{trn}}$ are in the same embedding space, they share the same decoder to reconstruct back to the time series $\mathbf{x}_{\text{aug}} = g(\mathbf{x}_{\text{trn}}; \mathbf{a})$ and $\mathbf{x}_{\text{trn}}$, respectively. As such, the loss function of f_{aug} is based on the reconstruction of both $\mathcal{D}_{\text{trn}}$ and $\mathcal{D}_{\text{aug}}$:

$$\mathcal{L}_{\text{aug}} = \sum_{\substack{\mathbf{x}_{\text{trn}} \in \mathcal{D}_{\text{trn}} \\ \mathbf{a} \sim \mathcal{A}^P}} \left((\mathbf{x}_{\text{trn}} - \tilde{\mathbf{x}}_{\text{trn}})^2 + (g(\mathbf{x}_{\text{trn}}, \mathbf{a}) - \tilde{\mathbf{x}}_{\text{aug}})^2 \right) \tag{10.2}$$

where $\tilde{\mathbf{x}}_{\text{aug}}$ is the output of $f_{\text{aug}}(\mathbf{x}_{\text{trn}}; \mathbf{a})$ for the hyperparameters sampled by g (Figure 10.1 left).

10.3.2 Module 2: Self-Tuning Module

Central to TSAP is the self-tuning module, which operates by iteratively refining the detector's parameters, $\boldsymbol{\theta}$, and the augmentation hyperparameters, $\mathbf{a}$. The process is structured into two phases: detection and alignment (see Figure 10.1 right). The overall algorithm is given in Algorithm 10.1.

Algorithm 10.1: Self-Tuning Module of TSAP

Data: Train & val. data $\mathcal{D}_{\text{trn}}$& $\mathcal{D}_{\text{val}}$, pre-trained augmentation model $f_{\text{aug}}(\cdot;\boldsymbol{\phi})$, detector $f_{\text{det}}(\cdot;\boldsymbol{\theta})$, max epoch T, #inner-loops L, and step sizes α and β

Result: Optimized augmentation hyperparameters $\mathbf{a}^*$ and optimized parameters $\boldsymbol{\theta}^*$

$\mathbf{a}^{(0)} \sim \text{Uniform}(\mathcal{A}^P)$;
for $t = 0, 1, \ldots, T-1$ **do**
  /* Phase (i) */
  for $l = 0, 1, \ldots, L-1$ **do**
    $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{trn}}(\boldsymbol{\theta}, \mathbf{a}^{(t)})$;
  end
  $\mathcal{Z}_{\text{trn}} \leftarrow \{f_{\text{det}}^{\text{enc}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{D}_{\text{trn}}\}$; // Phase (ii) starts
  $\mathcal{Z}_{\text{aug}} \leftarrow \{f_{\text{det}}^{\text{enc}}(f_{\text{aug}}(\mathbf{x}, \mathbf{a}^{(t)})) \mid \mathbf{x} \in \mathcal{D}_{\text{trn}}\}$;
  $\mathcal{Z}_{\text{val}} \leftarrow \{f_{\text{det}}^{\text{enc}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{D}_{\text{val}}\}$;
  $\mathcal{Z}_{\text{trn}}, \mathcal{Z}_{\text{aug}}, \mathcal{Z}_{\text{val}} \leftarrow \text{norm.}(\mathcal{Z}_{\text{trn}}, \mathcal{Z}_{\text{aug}}, \mathcal{Z}_{\text{val}})$;
  $\mathbf{a}^{(t+1)} \leftarrow \mathbf{a}^{(t)} - \beta \nabla_{\mathbf{a}} \mathcal{L}_{\text{val}}(\mathcal{Z}_{\text{trn}}, \mathcal{Z}_{\text{aug}}, \mathcal{Z}_{\text{val}})$;
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$; // Phase (ii) ends
  if $\mathcal{L}_{\text{val}}^{(t)} < \mathcal{L}_{\text{val}}^{*}$ **then**
    $\mathbf{a}^* \leftarrow \mathbf{a}^{(t+1)}$; $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$;
  end
end
Return $\mathbf{a}^*$ and $\boldsymbol{\theta}^*$;

Phase (i): Detection This phase focuses on estimating the parameters $\boldsymbol{\theta}^{(t)}$ of the detector f_{det} (comprising of an encoder $f_{\text{det}}^{\text{enc}}$ and a discriminator $f_{\text{det}}^{\text{mlp}}$) by minimizing the cross-entropy loss $\mathcal{L}_{\text{trn}}$. This aims to classify between the normal samples $\mathbf{x}_{\text{trn}}$ and the augmented pseudo anomalies $\tilde{\mathbf{x}}_{\text{aug}}$ by their embeddings $\mathcal{Z}_{\text{trn}}$ and $\mathcal{Z}_{\text{aug}}$, where $\mathcal{Z}_{\text{trn}} = \{f_{\text{det}}^{\text{enc}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{D}_{\text{trn}}\}$ and $\mathcal{Z}_{\text{aug}} = \{f_{\text{det}}^{\text{enc}}(f_{\text{aug}}(\mathbf{x}; \mathbf{a}^{(t)})) \mid \mathbf{x} \in \mathcal{D}_{\text{trn}}\}$ denote the embeddings of the training data and augmented data, respectively, given the current *fixed* $\mathbf{a}^{(t)}$ at iteration t. Note that the parameters $\boldsymbol{\phi}$ of the augmentation model f_{aug} are frozen throughout this phase.

Phase (ii): Alignment Subsequently, the alignment phase adjusts $\mathbf{a}$ to optimize the unsupervised differentiable validation loss $\mathcal{L}_{\text{val}}$, computed based on the embeddings from the now-updated $f_{\text{det}}^{\text{enc}}$. $\mathcal{L}_{\text{val}}$'s objective is to measure the degree of alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{val}}$ in the embedding space, as expressed by the Wasserstein distance in Equation (10.1). This metric is chosen for its effectiveness in capturing the overall distributional discrepancies between datasets, offering a more complete comparison than mere point-wise metrics [ACB17], which is especially important in TSAD given the often subtle nature of time series anomalies. The embeddings are normalized to ensure scale invariance before being passed to $\mathcal{L}_{\text{val}}$, so as to avoid the trivial solution of achieving alignment by setting all embeddings to the zero vector [YZA23a]. Lastly, as the embeddings $\{\mathcal{Z}_{\text{trn}}, \mathcal{Z}_{\text{aug}}, \mathcal{Z}_{\text{val}}\}$ in $\mathcal{L}_{\text{val}}$ are obtained through the updated $f_{\text{det}}^{\text{enc}}$, the optimization process needs to track the change in $\boldsymbol{\theta}$ caused by the update of $\mathbf{a}$. As such, TSAP uses a second-order optimization process, similar to [YZA23a].

Augmentation Type Selection While Algorithm 10.1 describes continuous hyperparameter tuning, the discrete hyperparameter (i.e. augmentation type) selection, corresponding to a specific anomaly type, is done through grid search as the number of anomaly types is finite. Hence, we initialize TSAP for different augmentation types and compare $\mathcal{L}_{\text{val}}$ across types to select the one that yields the best alignment. The idea is that the wrong augmentation type will have poor alignment, while one that reflects the true anomalies in $\mathcal{D}_{\text{test}}$ will result in better alignment, granted proper tuning of the continuous hyperparameters through Algorithm 10.1.

10.4 Experiments

In this section, we aim to answer the following questions:
RQ1. **Quantitative Comparison**: How well does TSAP perform on TSAD against baselines?
RQ2. **Qualitative Analysis**: How well does TSAP tune the augmentation hyperparameters?
RQ3. **Ablation Studies**: Which design decisions in TSAP drive its effectiveness?

Datasets We evaluate TSAP on six distinct TSAD tasks. Four of these are conducted in a *controlled* environment, while the remaining two are *natural*. In the controlled setting, the anomaly types are manually injected in $\mathcal{D}_{\text{test}}$ based on the types discussed in Section 10.3.1. The controlled environment allows for a thorough assessment of TSAP's ability to perform continuous hyperparameter tuning. For the natural environment, the anomaly types in $\mathcal{D}_{\text{test}}$ are a priori unknown and it is the goal of TSAP to find the type that yields best alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{val}}$ (part of $\mathcal{D}_{\text{test}}$), expressed by $\mathcal{L}_{\text{val}}$.

For controlled tasks, we use the 2017 PhysioNet Challenge dataset [CLM+17], which comprises real-world ECG recordings. Table 10.3 shows a selection of four different controlled TSAD tasks, constructed by manually injecting different anomaly profiles into the PhysioNet dataset. For PhysioNet A and B, given the anomaly type (Platform), the task is to infer or tune, respectively, the hyperparameter(s) `level` only and both `level` and `length`, while anomaly location is random (hence not tuned). For PhysioNet C and D, the respective tuning tasks are the same but for a different anomaly type (Trend). We choose Platform and Trend anomalies to represent, respectively, range-bound and range-exceeding anomalies.

The natural TSAD tasks are derived from the CMU Motion Capture (MoCap) dataset [*]. We consider the walking signal as normal data, and the signals of jumping and running as anomalies. To generate normal signals, we stitch the walking signals by identifying the start and end points of each gait phase and add random noise; whereas to generate anomalous ones, we stitch walking or running signals at a random location in the normal signal. This yields two distinct TSAD tasks as shown in Table 10.3. Different from PhysioNet A–D where we only tune the continuous hyperparameter(s) for the given (discrete) anomaly type, here for MoCap A and B, we aim to tune *both* the unknown anomaly type that corresponds to Jump and Run behavior, respectively, as well as the (continuous) hyperparameter `level` while `location` and `length` take random values.

[*] http://mocap.cs.cmu.edu/

Table 10.3: Anomaly profile of different TSAD tasks.

Dataset		Type	Level	Location	Length
PhysioNet	PhysioNet A	Platform	Fixed	Random	Random
	PhysioNet B	Platform	Fixed	Random	Fixed
	PhysioNet C	Trend	Fixed	Random	Random
	PhysioNet D	Trend	Fixed	Random	Fixed
MoCap	MoCap A	Jump	Fixed	Random	Random
	MoCap B	Run	Fixed	Random	Random

Baselines We compare TSAP with a selection of established baselines, including traditional and deep learning-based methods with demonstrated efficacy in TSAD [SWP22]. The traditional methods consist of different modeling approaches; namely, One-Class Support Vector Machines (**OC-SVM**) [SWS+99]; Local Outlier Factor (**LOF**)[BKNS00]; (**ARIMA**) [BJRL15]; Isolation Forest (**IF**) [LTZ08]; and the Matrix Profile (**MP**) [YZU+16]. On the deep learning side, we benchmark against the Encoder-Decoder LSTM (**EncDec-LSTM**) [MRA+16]; the Spectral Residual Convolutional Neural Network (**SR-CNN**) [RXW+19]; the Unsupervised Anomaly Detection (**USAD**) for TSAD [AMG+20]; and a recent time series foundation model (**TimeGPT**) [GC23]. Lastly, we include a state-of-the-art competing method which learns augmentations in the embedding space, called Neural Transformation Learning for (TS)AD (**NeuTraL-AD**) [QPK+21]. This diverse set of baselines allows for a comprehensive analysis across different approaches within the TSAD domain.

Model Configurations The Encoder_ϕ in f_{aug} and Encoder_θ in f_{det} are constructed using 1D CNN blocks [BKK18] (transposed 1D CNN for Decoder_ϕ) for efficient temporal feature extraction. W e choose the number of epochs T to allow sufficient time for the convergence of $\mathbf{a}$, with empirical evidence suggesting that $T = 100$ typically suffices. For the number of inner-loops L, we set $L = 5$, aligned with [YZA23a], such that f_{det} has adequate time to learn effective discriminative embeddings for $\mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{trn}}$.

Evaluation Metrics Our method calculates anomaly scores on an entire sequence level $\mathbf{x}$, similar to [QPK+21]. This is a different set-up compared to novelty detection in time series which typically operates on a point level. Detection on a sequence level can be especially important to spot long-term patterns (e.g. Trend anomalies). As such, we use the F_1 score and the Area Under the Receiver Operating Characteristic Curve (AUROC) as key performance metrics to quantify detection capability of anomalous sequences. All results are reported on the unseen $\mathcal{D}_{\text{test}}$. We determine the optimal F_1 score, by enumerating all possible thresholds, given by the anomaly scores for a given segment $\mathbf{x}$. We then compute the corresponding precision and recall for each threshold and select those that yield the highest F_1. As such, AUROC provides a balanced view whereas F_1 shows the optimal scenario. Both metrics range from 0 to 1, with higher values indicating superior performance.

Table 10.4: **TSAP outperforms most baselines** and has the best (lowest) average rank w.r.t. both F_1 and AUROC, with low standard deviation (in parentheses). Detection peformance of baselines w.r.t. F_1 and AUROC on test data across six TSAD tasks. Four tasks are performed in a controlled setting (manually injected anomaly type, cf. Section 10.3.1), based on the PhysioNet ECG data. Remaining two TSAD tasks exhibit natural anomalies (unknown real-world anomaly type), based on the MoCap data. Green (■, ■) marks the top two.

Methods	PhysioNet A		PhysioNet B		PhysioNet C		PhysioNet D		MoCap A		MoCap B		Avg. Rank	
	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC	F_1	AUROC
OC-SVM	0.182	0.468	0.182	0.472	0.373	0.803	0.393	0.806	1.000	1.000	0.546	0.806	7.2 (3.7)	6.8 (3.3)
LOF	0.999	1.000	0.999	1.000	0.354	0.738	0.358	0.725	0.196	0.506	0.221	0.577	6.8 (3.9)	6.5 (4.4)
ARIMA	0.885	0.960	0.829	0.965	0.991	0.999	0.999	0.999	0.870	0.955	0.225	0.537	4.3 (3.2)	4.7 (3.7)
IF	0.255	0.587	0.232	0.576	0.183	0.402	0.182	0.356	0.864	0.965	0.342	0.758	8.8 (1.7)	8.7 (1.9)
MP	0.812	0.743	0.812	0.744	0.280	0.712	0.284	0.734	1.000	1.000	1.000	1.000	5.0 (3.6)	4.8 (3.4)
EncDec-LSTM	0.190	0.508	0.190	0.508	0.415	0.812	0.442	0.819	0.980	0.999	0.909	0.996	6.5 (2.0)	6.7 (1.9)
SR-CNN	0.965	0.990	0.964	0.998	0.983	0.999	0.991	0.999	0.302	0.700	0.214	0.512	5.2 (4.2)	4.8 (4.5)
USAD	0.183	0.425	0.184	0.428	0.430	0.822	0.409	0.828	1.000	1.000	1.000	1.000	5.5 (4.0)	5.7 (4.5)
NeuTraL-AD	0.211	0.732	0.263	0.679	0.561	0.868	0.526	0.862	1.000	1.000	1.000	1.000	4.2 (2.9)	3.8 (2.5)
TimeGPT	0.327	0.714	0.318	0.711	0.218	0.580	0.217	0.525	0.348	0.743	0.385	0.683	8.0 (1.9)	8.3 (1.6)
TSAP (ours)	1.000	1.000	1.000	1.000	0.973	0.999	0.991	0.998	0.889	0.969	1.000	1.000	2.3 (2.0)	2.2 (2.0)

10.4.1 RQ1 – Quantitative Results

Table 10.4 provides the detection results for all six TSAD tasks. TSAP ranks the best overall in terms of both F_1 and AUROC. This shows that detector $f_{\text{det}}(\cdot; \boldsymbol{\theta}^*)$, trained through the alternating mechanism of TSAP, is able to generalize to unseen and unlabeled anomalies in $\mathcal{D}_{\text{test}}$.

While some competing methods perform strongly on a subset of tasks, they lack consistency across all TSAD tasks. Among the traditional baselines, which perform subpar as compared to deep TSAD approaches, LOF performs very well in PhysioNet A and B, relatively poorly on C and D, but exhibits near-random performance on MoCap. This discrepancy can be attributed to the nature of the anomalies in the PhysioNet data, which are manually injected and often characterized by pronounced and abrupt changes as shown in Figure 10.2. These abrupt changes lead to large differences in local density, which makes LOF thrive. In contrast, the MoCap dataset contains anomalies that emerge more subtly, as one gradually transitions from walking to running or jumping, reflecting more natural variations in the data which, in turn, lead to more subtle differences in local density. Similarly, SR-CNN and ARIMA perform strongly on the PhysioNet TSAD tasks, yet their effectiveness diminishes on MoCap tasks. Both methods exploit the abrupt changes present in the PhysioNet data but fail to detect the more subtle anomalies in MoCap B. This suggests that, while these methods work well for certain types of anomalies, their utility may be limited in scenarios where anomalies are more subtle. On the contrary, reconstruction-based baselines such as EncDec-LSTM and USAD face challenges on PhysioNet due to its high variability among inliers, which complicates the task of accurate data reconstruction. Yet, these methods excel with the MoCap dataset, where its consistent near-periodic pattern (Figure 10.2, bottom) lends itself to more reliable reconstruction, thereby enhancing anomaly detection performance. Similarly, MP struggles with the noisy PhysioNet data but excels at discord discovery in MoCap. Finally, NeuTraL-AD, a state-of-the-art augmentation-based baseline, demonstrates proficiency in properly augmenting the inlier data within MoCap. However, its performance on the PhysioNet variations reveals some weaknesses. NeuTraL-AD

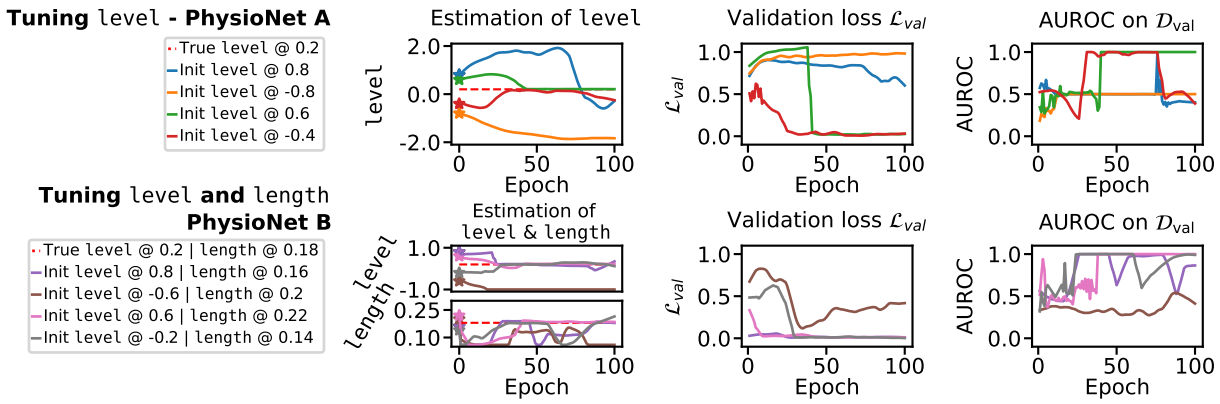


Figure 10.3: **TSAP find true continuous augmentation hyperparameter(s).** Top: Given Platform anomalies at true `level` (red dashed line), various initializations converge accurately near the true value (left), following the minimized values of val. loss (center), and leading to high detection performance (right). Bottom: Multiple continuous hyperparameters, here both `level` and `length`, are accurately tuned to near true values (left), as guided by minimizing the val. loss (center), achieving high AUROC (right). Notice that the diverged results in both cases associate with high (or hard-to-optimize) val. loss, which help us effectively reject low performance models.

struggles with the high variability in the PhysioNet inliers, inherent to real-world ECG signals. This suggests that the augmentation functions considered in NeuTraL-AD lack robustness when inliers are inherently noisy. We remark that only TSAP provides robust and consistent performance across all TSAD tasks, showcasing the effectiveness and generalizability of TSAP.

10.4.2 RQ2 – Qualitative Results

A key contributor to TSAP's consistent performance is the validation loss through which TSAP automatically learns the augmentation hyperparameters $\mathbf{a}$. Once $\mathbf{a}$ is determined, the task essentially reduces to a supervised learning problem. Next, we show that TSAP not only effectively tunes the continuous augmentation hyperparameters $\mathbf{a}$, but also that its validation loss guides the accurate selection of the discrete hyperparameter (i.e. anomaly type).

Controlled Environment Consider PhysioNet A, where we aim to tune the continuous hyperparameter a, i.e. `level`, of the Platform anomalies present in $\mathcal{D}_{\text{test}}$. That is, the `level` in $\mathcal{D}_{\text{test}}$ is fixed and tuning aims to estimate its value using TSAP while the other hyperparameters (`location, length`) are randomized. Figure 10.3 (top) shows TSAP's estimation process for different initializations of a. We observe that the initialization for $a \in \{-0.4, 0.6\}$ leads to the true a=0.2 (left). Simultaneously, the validation loss drops substantially once TSAP has arrived at the true a (center). This is also reflected in the performance of f_{det} on $\mathcal{D}_{\text{val}}$ which soars upon estimation of the true a (right). Conversely, the initialization for $a \in \{0.8, -0.8\}$ leads to a high validation loss, indicating poor alignment between $\mathcal{D}_{\text{trn}} \cup \mathcal{D}_{\text{aug}}$ and $\mathcal{D}_{\text{val}}$. Indeed, the performance of f_{det} on $\mathcal{D}_{\text{val}}$ now suffers from poor alignment.

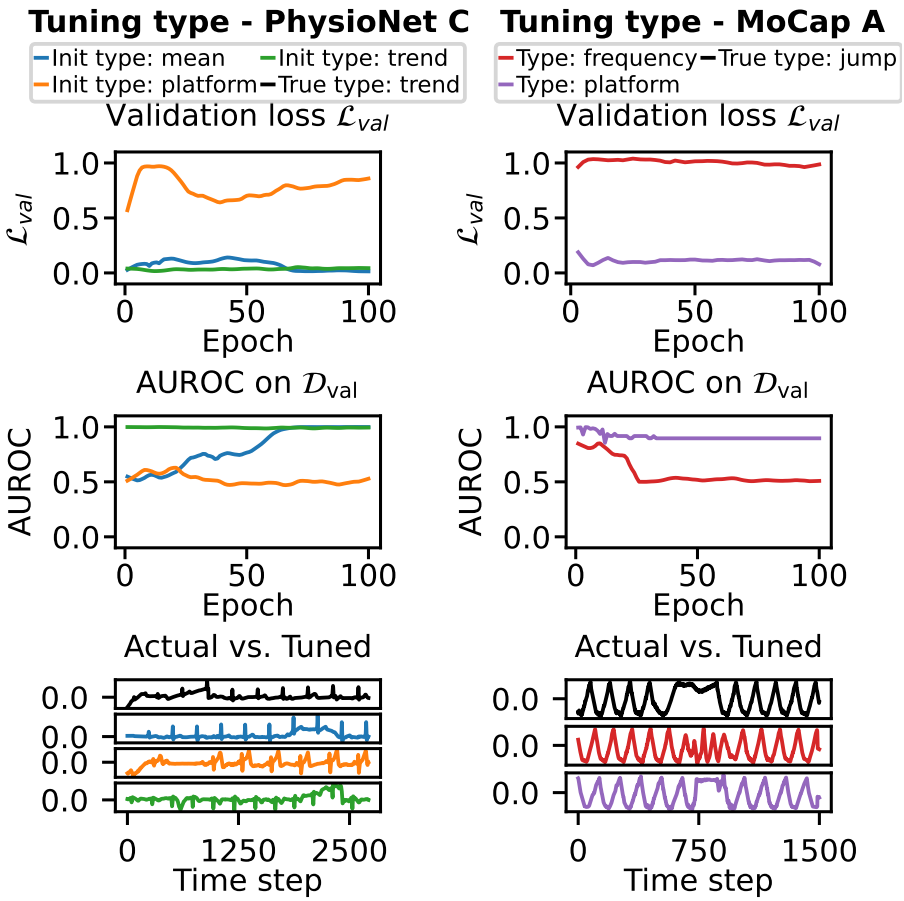


Figure 10.4: **TSAP finds true discrete hyperparameter (anomaly type).** Left: Given true Trend anomaly (black), val. loss favors both Trend (green) and Mean shift (blue) type, both with high AUROC (center) and high resemblance (right), and effectively rejects type Platform (orange) with poor performance. Right: For Jump anomalies in MoCap A with unknown type (black), val. loss favors type Platform (purple) that leads to high AUROC (center) and mimics well the true anomaly (right), and rejects type Frequency (red) with poor performance.

For PhysioNet B, we estimate both `level` and `length` while `location` is randomized. Figure 10.3 (bottom) demonstrates TSAP's ability to accurately estimate the `level` and `length`. While this is a by-product of our method, there are several real-world use-cases that can directly benefit from accurately learning the anomaly profile, from industrial equipment monitoring, to network security, and healthcare monitoring. This demonstrates TSAP's versatility given its capability of estimating continuous hyperparameters.

Figure 10.4 (left) showcases TSAP's ability to perform discrete hyperparameter selection. TSAP has been initialized and trained with three different anomaly types (Mean shift, Platform, Trend) on PhysioNet C (true anomaly type is Trend). The validation loss clearly showcases a misalignment between the Platform and Trend types (top), also reflected in the AUROC of f_{det} on $\mathcal{D}_{\text{val}}$ (center). Note how the Mean shift anomaly type has a low $\mathcal{L}_{\text{val}}$ towards the end of the training epochs, which is also reflected in the AUROC on $\mathcal{D}_{\text{val}}$. Indeed, comparing TSAP tuned

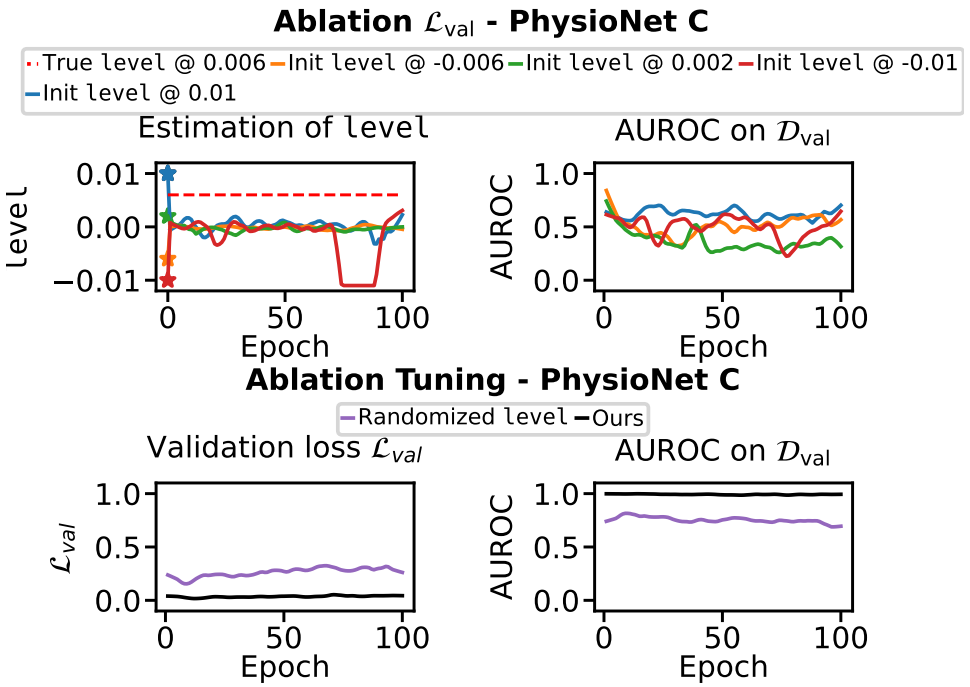


Figure 10.5: Overview of ablation studies. Top: TSAP's $\mathcal{L}_{\text{val}}$ is replaced by a point-wise val. loss leading to an erroneous estimation of a (left) and poor performance (right). Bottom: TSAP's self-tuning module is disabled, a is now randomized. Val. loss indicates poor alignment, reflected in poor performance.

for Mean shift anomalies and for Trend anomalies show strong resemblances with the true underlying anomaly type (bottom). This shows that the true underlying anomaly type is not necessarily the only type that yields high alignment, and in turn a high-performing detector.

Natural Environment In MoCap datasets, the anomaly types are a priori unknown. As such, we initialize TSAP with different augmentation types (Frequency and Platform) to perform discrete hyperparameter selection. Figure 10.4 (right) highlights its effectiveness as the validation loss clearly prefers one type over the other. Indeed, the natural anomalies defined by jumping signals in MoCap A have close resemblance to platform anomalies.

10.4.3 RQ3 – Ablation Studies

Validation Loss In Figure 10.5 (top), we illustrate the `level` estimation for PhysioNet C under the condition where our Wasserstein-based $\mathcal{L}_{\text{val}}$ is substituted with a point-wise metric, as used in [YZA23a]. This comparison shows that a point-wise validation loss tends to favor solutions where the `level` of the Trend anomaly approximates zero, neutralizing the anomaly. Although this might produce high alignment, it leads to poor f_{det} performance in $\mathcal{D}_{\text{val}}$ (right). This shows that the distributional characteristics captured by our $\mathcal{L}_{\text{val}}$ are a key contributing factor to the success of TSAP.

Randomization vs. Tuning In Figure 10.5 (bottom), the self-tuning module is disabled for PhysioNet C, where `level` is instead randomized (along with `location`, and `length`). We observe substantially higher $\mathcal{L}_{\text{val}}$, indicating poor alignment. In turn, f_{det} struggles to detect the unlabeled anomalies in $\mathcal{D}_{\text{val}}$. This showcases the utility of TSAP's systematic hyperparameter (self-) tuning over random choice.

10.5 Related Work

10.5.1 SSL for Anomaly Detection

SSL has emerged as a significant approach in machine learning. Foundation models [BHA+21] like large language models (LLMs) [Ope23, RPG+21], whose training heavily relies on SSL, have shaken up the world with outstanding performance. Time series tasks such as forecasting can also benefit from pre-trained LLMs [AST+24]. SSL has also transformed representation learning and significantly boosted several tasks in NLP, computer vision, recommender systems and medicine [SDS+21, BHX+22, KRT22]. SSL is especially attractive for unsupervised problems like anomaly detection [HHA24], because of its nature to create proxy tasks and loss functions without any labels. There are various types of SSL-based methods on anomaly detection [LZH+23], but the core idea is to use data augmentation functions [CSL21, GE18, LSYP21] on inliers to create pseudo anomalies, and learn a classifier that can detect the pseudo anomalies.

10.5.2 Time Series Anomaly Detection

There exists a large body of work on TSAD, for which we refer to surveys [BCML22, GGAH14]. SSL has been studied as the main approach to address TSAD problems [WSY+21, ZWZ+24]. BeatGAN [ZLH+19] and RobustTAD [GSW+20] use data augmentation to enrich training data. COUTA [XWJ+24] proposes three different augmentations, respectively mimicking local, contextual and collective anomalies. NeuTraL-AD [QPK+21] propose several augmentation functions that are an integral part of the learning process. AMSL [ZWC+23] also uses signal transformations as data augmentation. TimeAutoAD [JYST22] employs three different strategies to augment the training data for generating pseudo anomalies. Motivated by masked language models [DCLT19], MAD [FX22] investigated various masking procedures for multivariate TSAD based on a predictive pretext task. Note that only [JYST22, QPK+21] parameterize the augmentation procedures, but they tune those using *labeled* validation data.

10.5.3 Unsupervised Model Selection

Unsupervised hyperparameter tuning (i.e. model selection) is non-trivial in anomaly detection due to the absence of labeled data [MZZA23], and the literature is slim with quite recent efforts [ZA22, DZA24, ZRA21, ZZA22]. Specifically on SSL-based anomaly detection, a recent study [YZA23b] has revealed the impact of the choice of augmentation on SSL-based AD, leading to several works that aim to automatically search for the optimal choice [YZA23a, YZZA23]. However, none of the existing works addressed the model selection problem of SSL-based anomaly detection for time series data.

10.6 Conclusion

We introduced TSAP for self-supervised time series anomaly detection, which is the first attempt that automatically (self-)tunes the augmentation hyperparameters on time series data in an unsupervised manner. TSAP includes a differentiable model to augment the input time series data with various anomaly types, and an unsupervised validation loss that assists in aligning the augmented and test data.

Experiments show TSAP's ability in effectively selecting the augmentation type along with its continuous hyperparameters. Across various datasets with different types of time series anomalies, TSAP outperformed a diverse list of baselines, including modern neural and self-supervised approaches. While being the first self-tuning SSL solution to TSAD, our work opens new research directions. For instance, future extensions of TSAP could include an expanded catalog of supported anomaly types, broadening its applicability. Additionally, TSAP could be enhanced to deal with multiple different anomaly types within a given dataset, further strengthening its robustness. There is also potential to test the framework on various other datasets from different domains. Moreover, these ideas could be expanded to multivariate time series data, allowing TSAP to tackle more complex temporal relationships and dependencies effectively.

Chapter 11

GEN^{2}OUT: Detecting and Ranking Generalized Anomalies – Seizure Detection in EEG Recordings

Chapter based on work that appeared at Big Data 2021 [LSF+21] [PDF].

In a cloud of m-dimensional data points, how would we spot, as well as rank, both single-point- as well as group- anomalies? We are the first to generalize anomaly detection in two dimensions: The first dimension is that we handle both point-anomalies, as well as group-anomalies, under a unified view – we shall refer to them as *generalized anomalies*. The second dimension is that GEN^{2}OUT not only detects, but also ranks, anomalies in suspiciousness order. Detection, and ranking, of anomalies has numerous applications: For example, in EEG recordings of an epileptic patient, an anomaly may indicate a seizure; in computer network traffic data, it may signify a power failure, or a DoS/DDoS attack.

We start by setting some reasonable axioms; surprisingly, none of the earlier methods pass all the axioms. Our main contribution is the GEN^{2}OUT algorithm, that has the following desirable properties: (a) *Principled and Sound* anomaly scoring that obeys the axioms for detectors, (b) *Doubly-General* in that it detects, as well as ranks generalized anomal – both point- and group-anomalies, (c) *Scalable*, it is fast and scalable, linear on input size. (d) *Effective*, experiments on real-world epileptic recordings (*200 GB*) demonstrate effectiveness of GEN^{2}OUT as confirmed by clinicians.

Experiments on *27* real-world benchmark datasets show that GEN^{2}OUT detects ground truth groups, matches or outperforms point-anomaly baseline algorithms on accuracy, with no competition for group-anomalies and requires about *2 minutes for 1 million data point* on a stock machine.

11.1 Introduction

How would we spot and rank single-point- as well as group-anomalies? How can we draw attention of the clinician to strange brain activities in multivariate EEG recordings of an epileptic patient? How could we design an anomaly score function, so that it assigns intuitive scores to both point-, as well as group-anomalies? Our goal is to design a principled and fast anomaly detection algorithm for a given cloud of m-dimensional point-cloud data that provides a unified view as well as a scoring function for each generalized anomaly. This has numerous applications (intrusion detection in computer networks, automobile traffic analysis, outlier* detection in a collection of feature vectors from, say, medical images, or twitter users, or DNA strings, and more).

Our motivating application is seizure detection in EEG recordings [KLF+17, PSLF24]. Specifically, we want to spot those parts of the brain, and those time-ticks, that a seizure happened. Epilepsy is a neurodegenerative disease that affects 1-2% of the world's population and is characterized by recurrent seizures that intermittently disrupt the normal function of the brain through paroxysmal electrical discharges [Sho09]. At least 30% of patients with medically refractory epilepsy are resistant to the mainstay treatment by anti-epileptic drugs (AEDs). These patients may benefit from surgical therapy. A significant challenge of this therapy is identification of the region of the brain where seizures are originating, that is, the epileptogenic focus [KVW+15, VKT+17]. This region is then surgically either resected or electrically stimulated over time to control upcoming seizures long prior to their occurrence [TI06, CSTI09, HPPI18]. Accurate identification of the epileptogenic focus, as well as providing explanations to clinicians [NMN+23], is therefore crucial for effective epilepsy treatment.

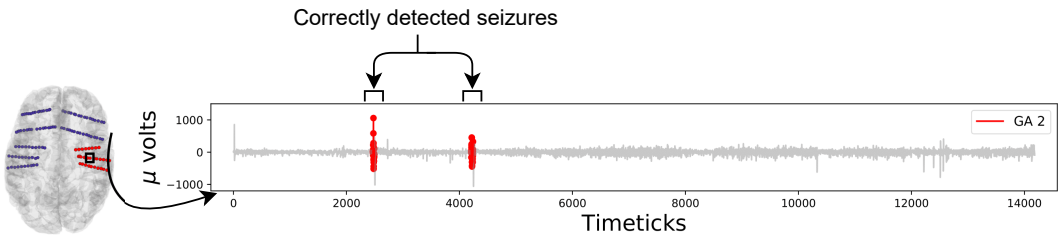
As suggested by the application domain, to achieve better outcomes for patients, it is critical to direct attention of the clinician to the anomalous time periods in the brain activity in order of their suspiciousness. The problem is two-fold: (a) *detection*, as well as (b) *ranking* of generalized anomalies. We want a scoring function for generalized anomalies, such that in the EEG/epilepsy setting it would score the groups which may correspond to anomalous periods e.g. seizure and draw attention to most anomalous time periods; thus aiding a domain expert in decision making. As we show in Section 11.3.1, we propose some intuitive axioms, that a generalized anomaly detector should obey.

Informal Problem 11.1: Doubly-General Anomaly Problem

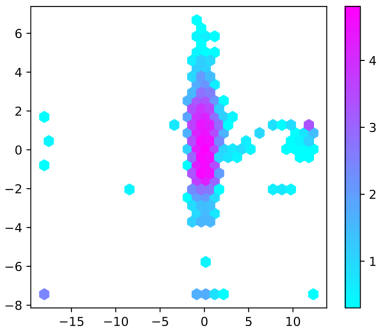
- **Given** a point-cloud dataset from an application setting,
- **Find** the point-anomalies and group anomalies, and
- **Rank** them in suspiciousness order.

Generality of Approach In most machine learning (ML) algorithms, we operate on clouds of points (after embedding, after auto-encoding etc). For example, time series is transformed

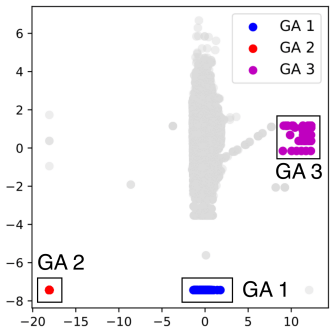
*We use outlier and anomaly interchangeably in this work.



(a) EEG data



(b) Heatmap of `http` data



(c) Detected group anomalies

Figure 11.1: **GEN²OUT is effective.** (a) **GEN²OUT matches ground truth.** Brain scan of the patient with electrode positions (*left*), and detected groups shown in color red (*right*), that matches the ground truth seizure locations. (b) Heatmap of *http* intrusion detection dataset. (c) **GEN²OUT correctly spots group (DDoS) attacks** in the intrusion detection dataset, marked GA1, GA2 and GA3.

into some form of $m-$dimensional cloud [BCML22] for further analysis; in images, numerical features are generated for learning tasks e.g. Imagenet [KSH12]. Thus, the proposed approach can be applied to diverse real data including point cloud, time-series and image data.

We propose GEN²OUT, which has the following properties:

- **Principled and Sound:** We identify five axioms (Section 11.3.1) and show that the proposed GEN²OUT obeys them, in contrast to top competitors.
- **Doubly-General:** GEN²OUT is doubly general. First dimension of generalization is size of anomalies – detecting point- and group-anomalies. Second dimension of generalization is scoring and ranking of generalized anomalies – both point- and group-anomalies.
- **Scalable:** Linear on the input size (Figure 11.9).
- **Effective**: Applied on real-world data (Figure 11.1 and 11.7), GEN²OUT wins in most cases over benchmark datasets for point anomaly detection. For group anomaly detection, GEN²OUT has no competitors as they need group structure information, and it agrees with ground truth on seizure detection.

Figure 11.1 illustrates the effectiveness of GEN²OUT, which detects group anomalies that correspond to seizure period in the patient; and, detects DoS/DDoS attack as group anomalies.

Table 11.1: **GEN²OUT matches all the specs.** Qualitative comparison of GEN²OUT against top competitors showing that every competitor misses one or more features.

Property \ Method	LODA [Pev16]	RRCF [GMRS16]	IF [LTZ08]	OCSMM [MS13]	AAE-VAE [CTC18]	MGM [XPS+11]	GLAD [YHL15]	GEN²OUT
Obey Axioms (Section 11.3.1)				?	?	?	?	✔
Discover Point Anomalies	✔	✔	✔					✔
Rank Point Anomalies	✔	✔	✔					✔
Discover Group Anomalies							?	✔
Rank Group Anomalies				✔	✔	✔	✔	✔
Jointly Rank Point- and Group- Anomalies								✔
Scalable	✔	?	✔	?	✔	?	?	✔

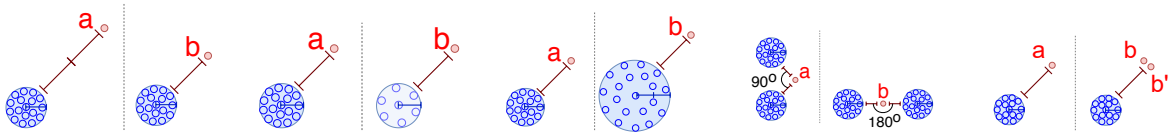
Reproducibility: Our source code and public datasets are at `https://github.com/mengchillee/gen2Out`.

11.2 Background and Related Work

Anomaly detection is a well-studied problem. Recent works [CBK09, GGAH14, Agg13, TC18, BZA21] provide a detailed review of many methods for anomaly and outlier detection. As shown in Table 11.1, GEN²OUT is the only method that matches the specs. Here, we review anomaly detection methods for point- and group-anomalies.

11.2.1 Point Anomaly Detection

Model-based and density-based methods for outlier detection are quite popular for point cloud data. Principal component analysis (PCA) based detectors [SCSC03] assume that the data follows a multi-variate normal distribution. Local outlier factor (LOF) [BKNS00] flags instances that lie in low-density regions. Clustering based methods [HXD03] score instances or small clusters by their distance to large clusters. However, these methods suffer from too many false positives as they are not optimized for detection [LTZ12]. Recently, a surge of focus has been on ensemble-based detectors that have been shown to outperform base detectors and are considered state-of-the-art for outlier detection [EDD+15]. Isolation forest [LTZ08] (IF), a state-of-the-art ensemble technique, builds a set of randomized trees that allows approximating the density of instances in a random feature subspace. [EDD+15] show that IF significantly outperforms other detectors such as LOF. IF shows that LOF has a high computation complexity (quadratic) and does not scale for large datasets. After that two more methods LODA [Pev16]



(a) A1: Distance Axiom (b) A2: Density Axiom (c) A3: Radius Axiom (d) A4: Angle Axiom (e) A5: Group Axiom

Figure 11.2: Illustration of Axioms

and Random Cut Forests (RRCF) [GMRS16] have been proposed as state-of-the-art methods. LODA is projection-based histogram ensemble that works well in many real settings. RRCF improves upon IF and use a data sketch that preserves pairwise distances.

11.2.2 Group Anomaly Detection

Numerous methods have been proposed for group anomaly detection [XPS+11, MS13, YHL15, CTC18]. Earlier approaches [MS13, CTC18, XPS+11] require the group memberships of the points known apriori, while Yu et al. [YHL15] requires information on pairwise relations among data points. Moreover, these methods focus only on scoring group-anomalies, and ignore point-anomalies unlike our method. GEN^{2}OUT detects and ranks anomalous points and groups, without requiring additional information on group structure of the dataset. As mentioned above, Table 11.1 summarizes comparison of GEN^{2}OUT against state-of-the-art point and group anomaly detection methods. As such none of the methods has all the features of Table 11.1 .

11.2.3 Fractals and Multifractals

In order to stress test our method, we use self-similar (fractals) clouds of points. We created the fractal images (Sierpinski triangle, biased line and 'fern' etc.), using the method and the code from Barnsley and Sloan [BS88]. We used the 'uniform' version (that is, for the Sierpinski triangle, all the miniature versions have the same weight of 1/3), also generated the 'biased' version of triangle using weights (0.6, 0.3, 0.1), and 'biased line' with *bias* $b = 0.8$ using weights (0.8, 0.2) that is b of the data points go to the first half of the line, and in this half, b of the data points go to first quarter of the line, and so on recursively (this, informally, is the 80-20 law).

11.3 Proposed Axioms and Insights

In this section, we explain our proposed axioms in detail and provide insights. An overview of symbols and acronyms is provided in Table 11.2 and Table 11.3, respectively. It is worth noting that these axioms are proposed to examine whether an anomaly detector is provided with the ability to compare the scores across datasets. The assumption is that, there are two different datasets with the same application setting. Although some of the axioms seem to be popular in single dataset setting, they are not considered and even ever mentioned by other studies when

there is more than one dataset. The observed insights are critical and penetrate this research. These greatly inspire us on selecting the core part of our anomaly detector.

11.3.1 Proposed Axioms

We propose five axioms an ideal anomaly detector should follow: producing higher anomaly scores when an instance is farther away from data kernel (*d*istance aware), or lies in low density locality (*density, radius and group aware*), and not aligned with majority of data (*a*ngle aware [KSZ08]). In the following, let $\boldsymbol{a} \in \mathrm{R}^m$ and $\boldsymbol{b} \in \mathrm{R}^m$ be m-dimensional anomalies in point cloud datasets S_a and S_b respectively. Additionally, suppose that normal observations are distributed uniformly in a disc in the datasets as shown in Figure 11.2 and $s(.)$ is the generalized anomaly score function.

Axiom 11.1: Distance Aware

All else being equal, the farther point from the normal observations is more anomalous.

$$\left.\begin{array}{l} S_a - \{\boldsymbol{a}\} = S_b - \{\boldsymbol{b}\}, \\ dist(\boldsymbol{a}, S_a) > dist(\boldsymbol{b}, S_b) \end{array}\right\} \implies s(\boldsymbol{a}) > s(\boldsymbol{b}) \tag{11.1}$$

Axiom 11.2: Density Aware

All else being equal, denser the cluster of points, more anomalous the outlier.

$$\left.\begin{array}{l} dist(\boldsymbol{a}, S_a) = dist(\boldsymbol{b}, S_b), \\ density(S_a) > density(S_b) \end{array}\right\} \implies s(\boldsymbol{a}) > s(\boldsymbol{b}) \tag{11.2}$$

Axiom 11.3: Radius Aware

All else being equal, for a given number of observations, smaller the radius of the cluster of points, more anomalous the outlier.

$$\left.\begin{array}{l} |S_a| = |S_b|, \\ dist(\boldsymbol{a}, S_a) = dist(\boldsymbol{b}, S_b), \\ radius(S_a) < radius(S_b) \end{array}\right\} \implies s(\boldsymbol{a}) > s(\boldsymbol{b}) \tag{11.3}$$

Axiom 11.4: Angle Aware

All else being equal, smaller the angle of a point with respect to cluster of observation, more anomalous the outlier.

$$\left.\begin{array}{l} |S_a| = |S_b|, \\ density(S_a) = density(S_b), \\ radius(S_a) = radius(S_b), \\ angle(\boldsymbol{a}, S_a) < angle(\boldsymbol{b}, S_b) \end{array}\right\} \implies s(\boldsymbol{a}) > s(\boldsymbol{b}) \tag{11.4}$$

Axiom 11.5: Group-Size Aware

All else being equal, the least populous group, the more anomalous it is. Let $g_a \subset S_a, g_b \subset S_b$ are the groups.

$$\left.\begin{array}{l} |g_a| < |g_b|, \\ |S_a - g_a| = |S_b - g_b|, \\ density(S_a) = density(S_b), \\ radius(S_a) = radius(S_b) \end{array}\right\} \implies s(g_a) > s(g_b) \tag{11.5}$$

Justification for Axioms Axiom A1 is self explanatory as shown in Figure 11.2a. The outlier point (shown in color red) in the left dataset (Figure 11.2a) being farther from the normal observations should be more anomalous. Consider the case of social networks. A node reachable via k hops from a close friends group should be more anomalous compared to reachable via k hops from a colleagues group. Figure 11.2b illustrates Axiom A2 where the outlier in the left dataset should be more anomalous. As shown in Figure 11.2c, for the same number of observations, the larger radius cluster would have a larger distance among points. Therefore, the outlier in the left dataset with smaller radius should be more anomalous. The farther points would tend to form a smaller angle with the cluster of observations (see Figure 11.2d) and should be more anomalous in the left dataset. The group $g_a = \{\boldsymbol{a}\}$ consisting of one point Figure 11.2e is intuitively more anomalous compared to group $g_b = \{\boldsymbol{b}, \boldsymbol{b}'\}$ containing more data points. For example, if g_b has 1000 points, it is not an anomaly anymore.

11.3.2 Insights

In this section, we are given the observations $X = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ where $\boldsymbol{x_i} \in \mathrm{R}^m$ for the anomaly detection. Our goal is to design an anomaly detector that obeys the axioms proposed in Section 11.3.1. The intuition for the selection of basic model is that, according to the five axioms in Figure 11.2, point 'a' in the first dataset should always have higher probability to be separated out comparing the point 'b' in the second dataset. ATOMICTREE has the properties which are very close to our demand. Here, we consider a randomized tree ATOMICTREE data

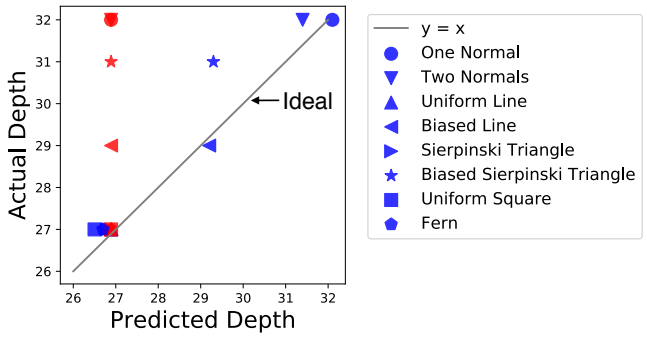


Figure 11.3: **GEN^{2}OUT wins** (in color blue) as the estimated depth is close to 45^o line. IF estimates the same depth for each dataset with #samples=1M.

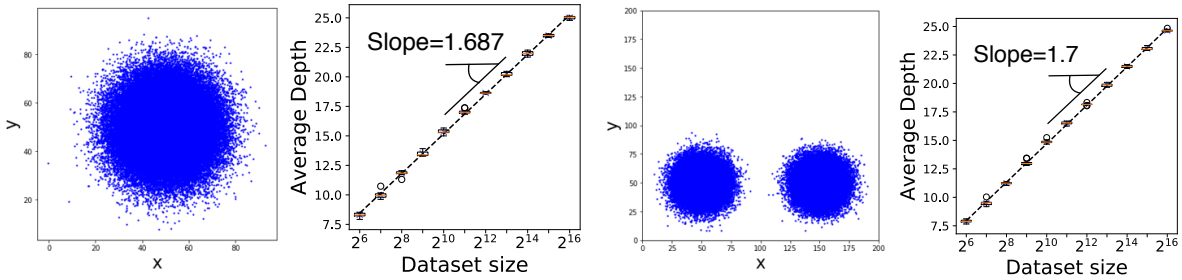
structure with the following properties – (i) Each node in the tree is either *leaf* node, or an internal node with two children, (ii) internal nodes store an attribute-value pair and dictate tree traversal. Given $X = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$, ATOMICTREE is grown through recursive division of X by randomly selecting an attribute and a split value until all the leaf nodes contain exactly one instance (hence the name ATOMICTREE) of observations assuming that observations are distinct. We randomly generate more than one tree to build a forest, to reduce the variance and detect outliers in subspaces.

We make a number of interesting observations while empirically investigating the process of tree growth for a variety of data distributions including multi-fractals. In Figure 11.4, we report depth (height) distribution of randomized trees averaged over 100 trees. We sample a number of points ($|X| \in \{2^{10}, 2^{11}, 2^{12}, 2^{20}\}; m = 2$) from each data distribution (left) and plot their corresponding depth (height) distribution (right). Notice that the number of points ($2^{\mathrm{x}}; \mathrm{x} \in \{6, 7, \ldots\}$) in the tree grows linearly with the average depth for any given dataset. In Figure 11.3, we plot the predicted depth for each of the distributions against the actual depth of the tree for the distributions shown in Figure 11.4 by fitting this linear trend. We present the following lemma based on the observations and draw the following insights.

Insight 11.1: Power Depth Property (PDP)

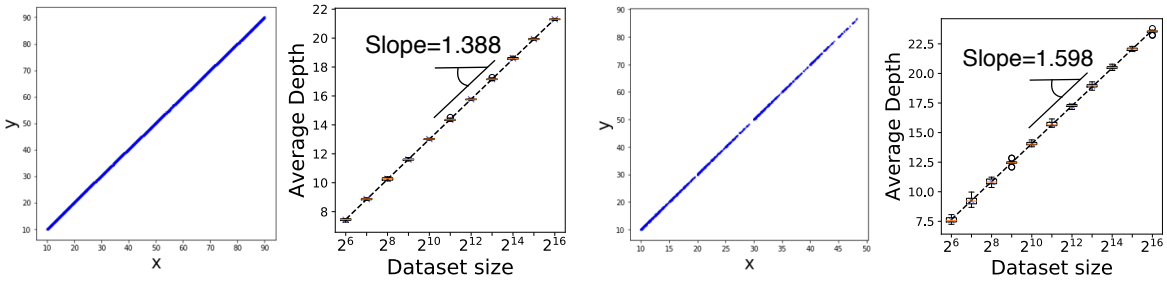
The growth of the tree depth with the logarithm counts of observations is linear irrespective of the data distribution.

Justification for PDP property In our attempt to explain PDP property, we study the expected depth computation for datasets with known distributions. However, in general, it is difficult. Let us consider *biased* line dataset with a bias factor b. Here we study a related setting: random points, but with fixed cuts. We refer to this model as 'fixed-cut' tree FIXEDCUTTREE. For this case, we can show that the PDP property holds, and the slope grows as the 'bias' factor b grows. Then, the depth of FIXEDCUTTREE for a *biased* fractal line (data in Figure 11.4d) obeys the following lemma.



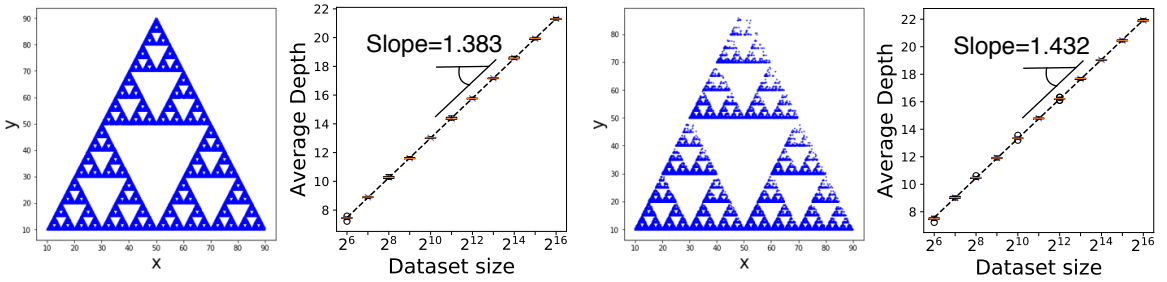
(a) One Normal

(b) Two Normals



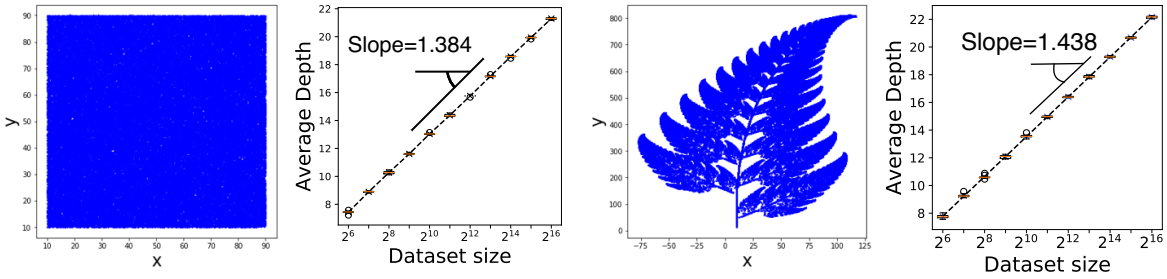
(c) Uniform Line

(d) Biased Line



(e) Sierpinski Triangle

(f) Biased Sierpinski Triangle



(g) Uniform Square

(h) Fern

Figure 11.4: Illustrating depth distribution for several diverse datasets (including Gaussian, Uniform, multifractals).

Table 11.2: Table of Symbols and Definitions.

Symbol	Definition
$X = \{\boldsymbol{x_i}\}$	Point cloud dataset where $\boldsymbol{x_i} \in \mathrm{R}^m$ for $i \in 1, 2, \ldots n$
$s(.)$	Anomaly score function for an outlier detector
$h(\boldsymbol{q})$	Path length estimate for instance $\boldsymbol{q}$ in a depth-limited AtomicTree
$\mathbb{E}[h(\boldsymbol{q})]$	Path length averaged over the ensemble
$H(n)$	Depth estimation function for an AtomicTree with n observations
d_{limit}	Depth limit of a AtomicTree

Table 11.3: Table of Acronyms.

Acronym	Definition
IF	Isolation Forest
AP	Average Precision
ROC	Receiver Operating Characteristic

Lemma 11.1: Expected Depth of fixedCutTree

The expected tree depth $H(n, b)$ for a *biased* line with a bias factor b containing $n \geq 2$ data points is given as:

$$H(n,b) = \sum_{k=0}^{n} \Big[\binom{n}{k} b^k (1-b)^{n-k} \times \tag{11.6}$$

$$\big(\frac{k}{n} H(k,b) + \frac{n-k}{n} H(n-k,b) + 1\big)\Big] \tag{11.7}$$

Proof. Let $H(k, b)$ be the depth of fixedCutTree with k observations constructed using $X_k \subseteq X$. Since fixedCutTree is grown via recursive partitioning on a randomly chosen attribute-value, therefore, for a biased line, $b =$ probability of a point going to left node i.e. the point less than chosen attribute-value. Let k be the number of points partitioned onto the left node, then $n - k$ points go to right node. Define $\mathrm{B}(n, k, b) = \binom{n}{k} b^n (1-b)^{n-k}$ the Binomial probability for a fixed k. Let $f(n, k, b)$ be the estimate of the depth when k observations are in left node, then $f(n, k, b) = \big(\frac{k}{n} H(k, b) + \frac{n-k}{n} H(n-k, b) + 1\big)$ as each random partition increases depth by 1. Therefore, the expected depth of the tree is given as $H(n, b) = \sum_{k=0}^{n} f(n, k, b)\mathrm{B}(n, k, b)$. ■

We denote $H(n, b) = H(n)$ for $b = 1/2$. A tree with one data point would have a depth of one i.e. $H(1, b) = 1 = H(1)$; and $H(0, b) = 0 = H(0)$. In Figure 11.5, we show the effect of bias on the the (analytical) depth computed using $H(n, b)$. Notice that increase in bias – indicating deviation from uniformity – increases depth which matches intuition. For bias $1 - b$, $H(n, 1-b)$ follows the results for $H(n, b)$.

Following the PDP property, the depth estimation function is given as

$$H(n) \approx w_0 + w_1 log_2(n) \tag{11.8}$$

where w_0 and w_1 are parameters that we estimate for each data distribution, and n is the number of instances in the dataset.

Insight 11.2

The slope of the linear fit varies significantly depending on the dataset distribution.

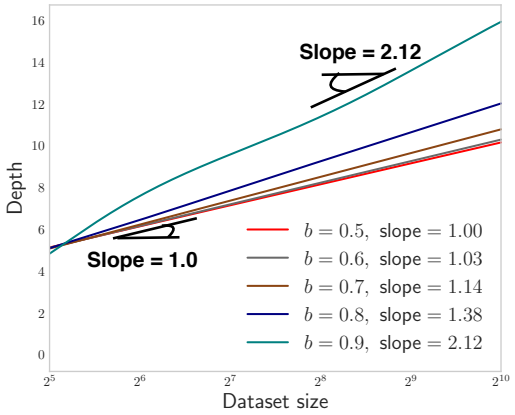


Figure 11.5: Depth ($\boldsymbol{H(n, b)}$) vs. Dataset Size: slope increases with increase in bias for a *biased* line data.

Table 11.4: **GEN^{2}OUT$_0$ obeys all the axioms** a generalized anomaly detector should follow. We compare the methods statistically, by conducting two-sample t-test based on scores obtained for points $\boldsymbol{a}, \boldsymbol{b}$. A positive difference in score indicates that the detector follows that axiom (see Figure 11.2). ■ indicates that the detector follows the axiom, ■ indicates that the detector does not obey the axiom.

Method	**LODA**		**RRCF**		**IF**		**GEN^{2}OUT$_0$**	
Metric	Statistic	p-value	Statistic	p-value	Statistic	p-value	Statistic	p-value
A1: Distance Axiom	0	1	3.6	0.002**	2.1	0.054	11.4	1.2e-9***
A2: Density Axiom	7e15	2e-275***	-0.14	0.89	-10	8.6e-9***	25.2	1.7e-15***
A3: Radius Axiom	0	1	6.4	4.8e-6***	11.9	5.9e-10***	21.3	3.4e-14***
A4: Angle Axiom	6.6	3.2e-6***	17.5	9.6e-13***	-0.2	0.83	53.7	2.5e-21***
A5: Group Axiom	-14.7	1.8e-11***	1.1	0.27	0.95	0.35	28.2	2.6e-16***

For example, the slope for Uniform Line (see Figure 11.4c) is 1.38, while for a Uniform Square (see Figure 11.4g) is 1.66. These insights lead to the following lemma.

Lemma 11.2

GEN^{2}OUT includes IF as a special case.

Proof. In Equation (11.8), setting $w_0 = 2 \times 0.57 - (2(n-1)/n)$ and $w1 = 2 \times log_e(2)$ yields the average path length function used in IF. Here, 0.57 is the Euler's constant, and $log_e(2)$ accounts for the difference in log bases. ■

Drawing from these insights, next we present the details of our anomaly detection algorithm.

Algorithm 11.1: GEN^{2}OUT$_0$-FIT

Data: A data matrix X, number of ATOMICTREE estimators `numTrees`, ATOMICTREE depth limit d_{limit}

Result: w_0, w_1 of depth estimation function $H(\cdot)$ and ATOMICTREE ensemble

Initialize Y and Z;
/* Estimating the function $H(\cdot)$ */
for $i = n_1, n_1 + 1, \ldots;$ // a small n_1 e.g. 10
do
  Draw $X_s \subset X$ s.t. $|X_s| = 2^i$;
  $F_s \leftarrow$ CONSTRUCT-ATOMICTREE (X_s, ∞);
  $Z \leftarrow Z \cup$ average depth of F_s containing observations X_s;
  $Y \leftarrow Y \cup i$;
end
$H(.) \leftarrow$ Fit linear regression Y and Z;
$w_0, w_1 \leftarrow$ `coefficients`$(H(.))$;
/* Construct the ensemble */
for *t = 1 to* `numTrees` **do**
  `ensemble` $\leftarrow$ `ensemble` $\cup$ CONSTRUCT-ATOMICTREE (X, d_{limit});
end
Return w_0, w_1, `ensemble`;

11.4 Proposed Method

For ease of exposition, we describe the algorithm in two steps – GEN^{2}OUT$_0$ for point anomalies, and then GEN^{2}OUT for generalized anomalies.

11.4.1 Point Anomalies: GEN^{2}OUT$_0$

Given the observations $X = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_M}\}$ where $\boldsymbol{x_i} \in \mathrm{R}^m$, GEN^{2}OUT$_0$'s goal is to detect and assign anomaly score to outlier points. GEN^{2}OUT$_0$ uses an ensemble of depth-limited randomized tree ATOMICTREE (Section 11.3.2) that recursively partition instances in X.

Definition 11.1: Depth Limited ATOMICTREE

An ATOMICTREE that is constructed by recursively partitioning the given set of observations X until a depth limit d_{limit} is reached or the *leaf* nodes contain exactly one instance.

As evidenced in prior works, the random trees induce shorter path lengths (number of steps from root node to leaf node while traversing the tree) for anomalous observations since the instances that deviate from other observations are likely to be partitioned early. Therefore, a shorter average path length from the ensemble would likely indicate an anomalous observation. Anomaly detection is essentially a ranking task where the rank of an instance indicates its

Algorithm 11.2: CONSTRUCT-ATOMICTREE

Data: A data matrix X,d_{limit}, `currDepth:0`
Result: ATOMICTREE

Initialize ATOMICTREE;
if $d_{limit} \leq$ *currDepth or* $|X| \leq 1$ **then**
    Return a leaf node of size $|X|$;
else
    Pick an attribute at random from X;
    Pick an attribute value at random;
    $X_l \leftarrow$ set of points on the left ($<$) of the chosen attribute-value pair;
    $X_r \leftarrow$ set of points on the right ($\geq$) of the chosen attribute-value pair;
    `left` $\leftarrow$ CONSTRUCT-ATOMICTREE (X_l,d_{limit}, `currDepth + 1`);
    `right` $\leftarrow$ CONSTRUCT-ATOMICTREE (X_r,d_{limit}, `currDepth + 1`);
    Return an internal node with {`left`, `right`, {chosen attribute-value pair}};
end

relative degree of anomalousness. We next design anomaly score function for our algorithm to facilitate ranking of observations.

Proposed Anomaly Score We construct anomaly score using the path length $h(\boldsymbol{q})$ for each instance $\boldsymbol{q} \in \mathrm{R}^m$ as it traverses a depth limited ATOMICTREE. The path length for $\boldsymbol{q}$ is $h(\boldsymbol{q}) = h_0 + H(l_{busy})$ if $l_{busy} > 1$; otherwise $h(\boldsymbol{q}) = h_0$ where h_0 is the number of edges $\boldsymbol{q}$ traverses from *root* node to *leaf* node that contains l_{busy} points in a depth limited ATOMICTREE. When $l_{busy} > 1$, we estimate the expected depth from the leaf node using $H(l_{busy})$ (uses Equation (11.8)). We normalize $h(\boldsymbol{q})$ by the average tree height $H(n)$ (height of ATOMICTREE containing n observations) for the depth limited ATOMICTREE ensemble to produce an anomaly score $s(\boldsymbol{q}, n)$ for a given observation $\boldsymbol{q}$. Referring to the PDP insights we presented in Section Section 11.3.2, we estimate the data dependent $H(\cdot)$ using Equation (11.8) since the tree depth grows linearly with the number of observations (in log_2) in the tree (see Figure 11.4). The slope of the linearity is characterized by underlying data distribution; each distribution follows a linear growth. The score function is

$$s(\boldsymbol{q}, n) = 2^{-\frac{\mathrm{E}[h(\boldsymbol{q})]}{H(n)}} \tag{11.9}$$

where $\mathrm{E}[h(\boldsymbol{q})]$ is the average path length of observation $\boldsymbol{q}$ in the ATOMICTREE ensemble, n is number of data points used to construct each ATOMICTREE, and $H(n)$ is the function for estimating depth of the tree given in Equation (11.8).

GEN^{2}OUT$_0$ Parameter Fitting GEN^{2}OUT$_0$ is a depth limited ATOMICTREE ensemble. The algorithm for fitting GEN^{2}OUT$_0$ parameters is provided in Algorithms 11.1 and 11.2.

GEN^{2}OUT$_0$ Scoring To assign anomaly scores to the instances in a data matrix X, the expected path length $\mathrm{E}(h(\boldsymbol{q}))$ for each instance $\boldsymbol{q} \in X$. $\mathrm{E}(h(\boldsymbol{q}))$ is estimated by averaging the

Algorithm 11.3: GEN^{2}OUT$_0$-Scoring

Data: A data matrix X, ATOMICTREE ensemble
Result: Anomaly scores `scores` for observations in X

Initialize `depths`, `scores`, and l_{busy};
$n \leftarrow$ `numSamplesInATOMICTREE`;
for $\boldsymbol{x} \in X$ **do**
    `depths` $\leftarrow$ `depths` $\cup$ compute path-lengths for $\boldsymbol{x}$ (Section 11.4.1);
    $l_{busy} \leftarrow l_{busy} \cup$ compute number of samples in `leaf` where traversal of $\boldsymbol{x}$ terminated;
end
for *depth* $\in$ `depths`, $l \in l_{busy}$ **do**
    $h =$depth$+H(l)$;
    $s = 2^{\frac{-h}{H(n)}}$;
    `scores` $\leftarrow$ `scores` $\cup\, s$;
end
Return `scores`;

path length after tree traversal through each ATOMICTREE in GEN^{2}OUT ensemble. We outline the steps to assign anomaly score to a data point using GEN^{2}OUT$_0$ in Algorithm 11.3. As shown in Table 11.4, GEN^{2}OUT$_0$ is the only anomaly detector that obeys all our proposed axioms, and thus can be used for group anomaly detection.

11.4.2 Full Algorithm: GEN^{2}OUT

How can we design an algorithm that can spot both point- as well as group-anomalies, simultaneously? The main insight is to exploit the less-appreciated ability of sampling to drop outliers, with high probability. How can we use this property to spot group-anomalies, of size, say n_g (in a population of n data points)? The idea is that, with a sampling rate of n_g/n, a point $\boldsymbol{a}$ of the group will probably be stripped of its cohorts, and thus behave like a point-anomaly, exhibiting a high anomaly score. For disambiguation versus the sampling of GEN^{2}OUT$_0$, we will refer to this sampling process as '*qualification*', and to the corresponding rate as qr= *qualification rate*.

In more detail, to determine whether point $\boldsymbol{a}$ belongs to a group-anomaly, we compute its (GEN^{2}OUT$_0$) score $s(\boldsymbol{a}, qr)$ for several qualification rates qr; when the score peaks (say, at rate n_g/n) then n_g is roughly the size of the group-anomaly (= micro-cluster) that $\boldsymbol{a}$ belongs to. Some definitions:

Definition 11.2: X-RAY Line

For a given data point $\boldsymbol{a}$, the X-RAY line is the function ($score(\boldsymbol{a}, qr)$ versus qr).

Algorithm 11.4: GEN^{2}OUT

Initialize $n \leftarrow |X|$;
/* Step 0: Fit a sequence of gen^{2}Out$_0$ */
for $qr \in \{1, 1/2, 1/4, \cdots\}$ **do**
  Draw $X_s \subset X$ s.t. $|X_s| = n \times qr$;
  GEN^{2}OUT$_0$-*ensembles* $\leftarrow$ GEN^{2}OUT$_0$-*ensembles* $\cup$ GEN^{2}OUT$_0$-FIT (X_s, ., .);
end
/* Step 1: create X-ray plot */
for *e* $\in$ *GEN^{2}OUT$_0$-ensembles* **do**
  /* generate score for specific qualification rate */
  scores $\leftarrow$ scores $\cup$ GEN^{2}OUT$_0$-Scoring(X, e);
end
/* Step 2: Apex extraction */
/* max score and rate for each point across qualified datasets */
max_scores, max_qr $\leftarrow \arg\max$(scores) ;
candidate-points $\leftarrow X$[max_scores $\geq$ *threshold*];
/* Step 3: Outlier grouping */
for *r* $\in$ *unique(max_qr)* **do**
  candidate-points_r ; // candidate points at this rate
  /* identify more than one group per qualification rate */
  clusters $\leftarrow$ *cluster* candidate-points_r;
end
/* Step 4: Compute iso-curves */
for $cl \in$ clusters **do**
  /* points closer to (score=1, qr=1) is more anomalous */
  iso_scores $\leftarrow \frac{2-ManhattanDistance([\frac{\log_2 max\_qr(\boldsymbol{a})}{10}+1, max\_score(\boldsymbol{a})],[1,1])}{2}$ $\forall \boldsymbol{a} \in cl$;
end
/* Step 5: Scoring micro-clusters */
Assign scores $\leftarrow$ *median*(iso_scores(cl)) $\forall cl \in$ clusters

Definition 11.3: X-RAY Plot

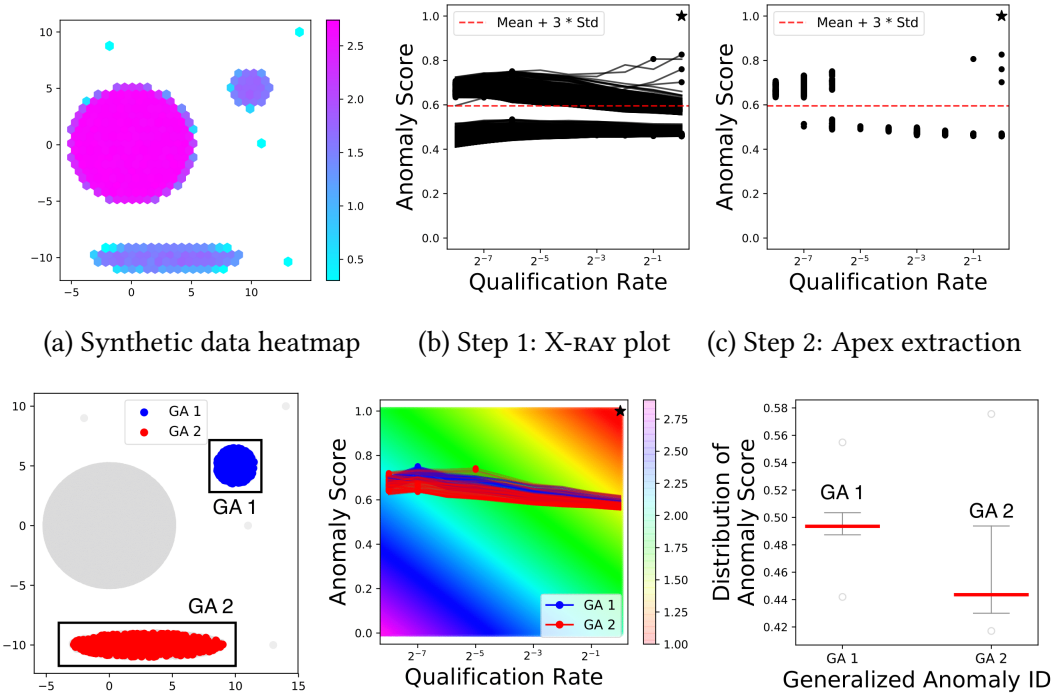
For a cloud of n points, the X-RAY plot is the 2-d plot of all the n X-RAY-lines (one for each data point)

See Figure 11.6b for an example.

Definition 11.4: APEX

Apex of point $\boldsymbol{a}$ is the point $(score, qr)$ with the highest anomaly score.

See Figure 11.6c for an example.



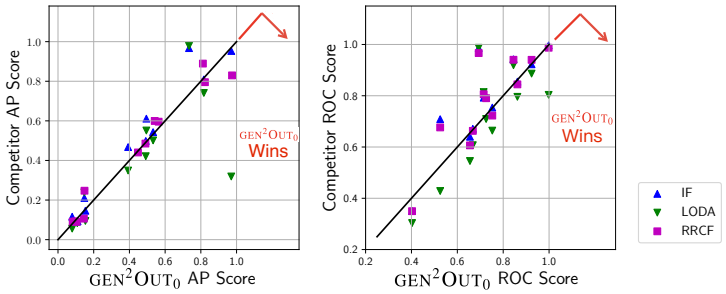
(a) Synthetic data heatmap (b) Step 1: X-RAY plot (c) Step 2: Apex extraction

(d) Step 3: Outlier grouping (e) Step 4: Anomaly iso-curves (f) Step 5: Scoring

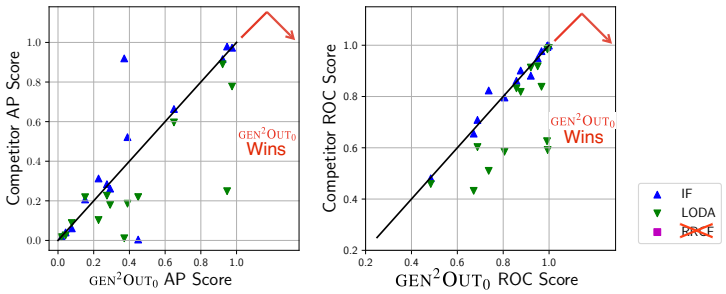
Figure 11.6: **GEN^{2}OUT correctly detects group anomalies.** Illustration of GEN^{2}OUT on synthetic dataset.

Algorithm 11.4 describes the steps of the proposed GEN^{2}OUT. In summary, we find the X-RAY plot (Step 1) and then find the apex point for every data point $\boldsymbol{a}$ (Step 2); keep the ones with high apex and then cluster the corresponding data points (Step 3); and then assign scores to the each group (Step 4 and Step 5).

Figure 11.6 illustrates the steps in GEN^{2}OUT on a synthetic dataset that has two anomalous groups along with several point anomalies. Figure 11.6b finds the X-RAY plot and Figure 11.6c shows the apex with the red threshold line. We find two groups after applying clustering (DBSCAN [SSE+17] in our implementation) shown in color red, and blue in Figure 11.6d. Then we compute the similarity of points in X-RAY plot representation in each cluster to the theoretically most anomalous point at score=1, qr=1 (see iso curves in Figure 11.6e), and then assign generalized anomaly score using the median of the similarity scores as shown in Figure 11.6f. GEN^{2}OUT correctly assigns higher score to GA1 (blue cluster in Figure 11.6f) which contains 1000 points as compared to GA2 (red cluster in Figure 11.6f) containing 2000 points (also see Axiom A5). For ease of visualization, we do not show point-anomalies in this plot.



(a) Dataset size $<$ 3000



(b) Dataset size $\geq$ 3000

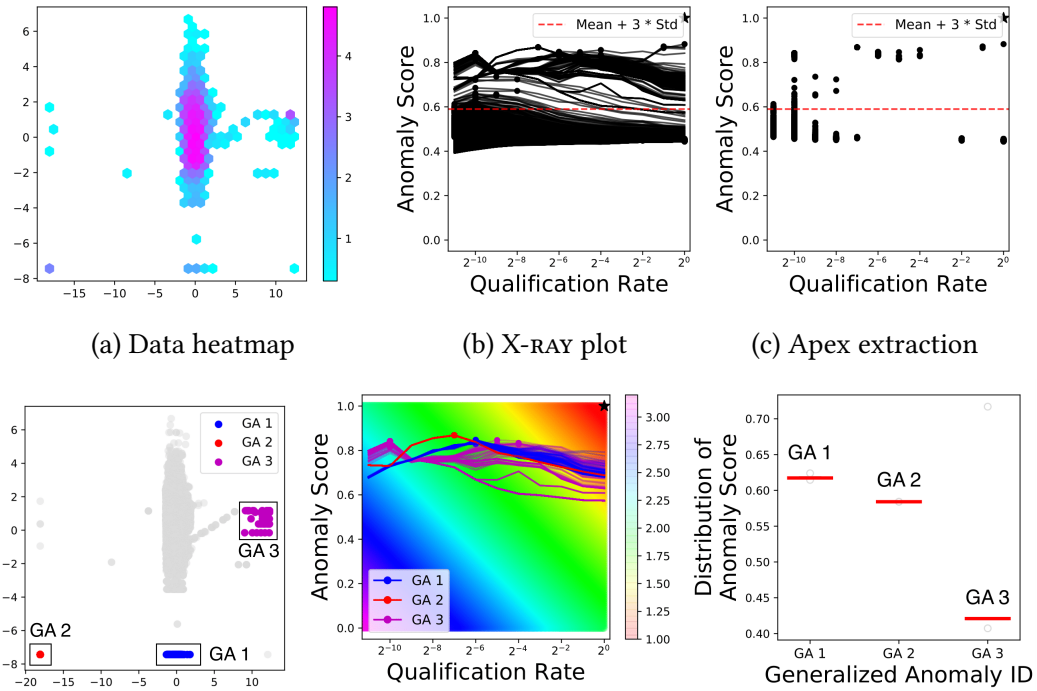
Figure 11.7: **GEN²OUT₀ wins in point anomaly detection**. We plot average precision (AP) and area under the ROC curve for GEN²OUT₀ against the same metric of the competitors (none of which obey all our axioms). Points representing benchmark datasets are below the line for the majority of datasets. RRCF does not scale to datasets with size greater than 3000.

11.5 Experiments

We evaluate our method through extensive experiments on a set of real-world datasets. We provide dataset details and the experimental setup, followed by the experimental results. We aim to answer following research questions (RQ):
RQ1. **Point Anomalies:** How well does GEN²OUT detect point anomalies?
RQ2. **Group Anomalies:** How well does GEN²OUT detect group anomalies?
RQ3. **Scalability:** How scalable is GEN²OUT to large point-cloud datasets?

Epilepsy Dataset We analyzed intracranial electroencephalographic (EEG) signals recorded at the Epilepsy Monitoring Unit of a large public university from one patient with refractory epilepsy. Electrodes were placed in the brain and EEG signals were then recorded across 122 electrode contacts at a sampling rate of 2 KHz with focal region in the right temporal lobe.

Benchmark Datasets Our benchmark set consist of 26 real-world outlier detection datasets from ODDS repository [Ray16]. The datasets cover diverse application domains and have diverse range dimensionality and outlier percentage. The ODDS datasets provide ground truth outliers that we use for the quantitative evaluation of the methods.



(a) Data heatmap (b) X-RAY plot (c) Apex extraction

(d) Outlier grouping (e) Anomaly iso-curves (f) Scoring

Figure 11.8: **GEN²OUT detects DDoS attacks** on intrusion detection `http` dataset.

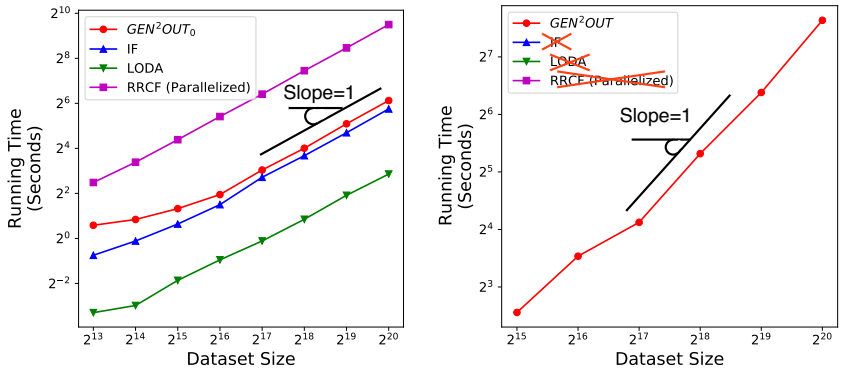
11.5.1 RQ1 – Point Anomalies

We compare GEN²OUT$_0$ to the following state-of-the-art ensemble baselines:

1. **IF**: Isolation Forest [LTZ08] uses an ensemble of randomized trees to flag anomalies.
2. **LODA**: Lightweight on-line detector of anomalies [Pev16] is a projection based histogram ensemble of weak estimators.
3. **RRCF**: Robust Random Cut Forest [GMRS16] are tree ensembles that use sketch based anomaly detector.

To evaluate effectiveness, we compare GEN²OUT$_0$ to state-of-the-art ensemble baselines on a set of real-world point-cloud benchmark outlier detection datasets. We use average precision (AP) and receiver operating characteristic (ROC) scores as our evaluation metrics. We plot the scores (AP and ROC score) for each competing method on all the benchmark datasets in Figure 11.7.

If the points are below the 45 degree line where each point represents a dataset, then it indicates that GEN²OUT$_0$ outperforms the competition in those datasets. As shown in Figure 11.7, for both the evaluation metrics, GEN²OUT$_0$ beats or at least ties with all baselines on majority of the datasets (see Figure 11.1c). The quantitative evaluation demonstrates that GEN²OUT$_0$ is superior to its competitors in terms of evaluation performance as well as obeys all the proposed axioms while none of the competition obeys the axioms.



(a) GEN^{2}OUT$_0$ (b) GEN^{2}OUT

Figure 11.9: (a) **GEN^{2}OUT$_0$ is fast and scalable**: Evaluation on benchmark datasets show that GEN^{2}OUT$_0$ (in red) scales linearly (eventual slope=1 in log-log scales). Note that none of the competitors obeys the axioms, and RRCF is much slower. (b) **GEN^{2}OUT is fast and scalable**, linear in size of input.

11.5.2 RQ2 – Group Anomalies

We evaluate the effectiveness of GEN^{2}OUT on real-world intrusion dataset that has attributes describing duration of attack, source and destination bytes. Note that we do not include group anomaly detection methods for comparison as they require group structure information, hence do not apply to our setting. Figure 11.8a shows source bytes plotted against destination bytes for the points. Figures 11.8b – 11.8f shows the X-RAY plot with scores trajectory, APEX with candidate points above the threshold (set at mean + 3 standard deviation of scores in full dataset), identified groups and the generalized anomaly score for each detected group. GEN^{2}OUT matches ground truth as it detects the three anomalous groups as shown in Figure 11.8d. In short, GEN^{2}OUT is able to detect groups that correspond to distributed-denial-of-service attack.

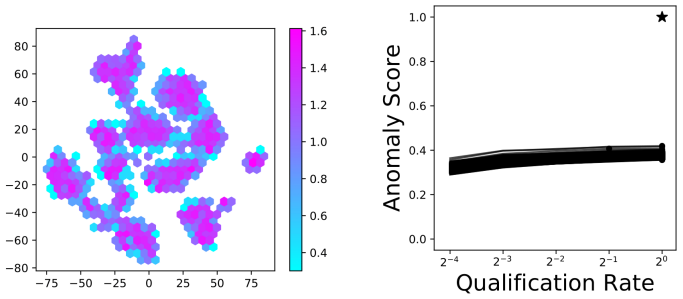
11.5.3 RQ3 – Scalability

To quantify the scalability, we empirically vary the number of observations in the chosen dataset and plot against the wall-clock running time (on 3.2 GHz 36 core CPU with 256 GB RAM) for the methods. First we compare GEN^{2}OUT$_0$ against the competitors in Figure 11.9a for point-anomalies. The running time curve of GEN^{2}OUT$_0$ is parallel to the running time curve of IF, which shows that GEN^{2}OUT$_0$ does not increase time complexity except adding a small constant overhead for estimating the depth function $H(.)$. The running time of RRCF is much higher than others even after implementing the trees in parallel. Note that only GEN^{2}OUT$_0$ obeys the axioms. For generalized anomalies, Figure 11.9b reports the wall-clock running time of GEN^{2}OUT as we vary the data size. Notice that GEN^{2}OUT scales linearly with input size. Importantly, competitors do not apply as they require additional information.

11.6 GEN^{2}OUT at Work

11.6.1 No False Alarms

When applied to datasets containing only normal groups that are relatively equal in size, GEN^{2}OUT correctly identifies them as normal groups i.e. does not flag any set of points as anomalous group. To illustrate this phenomenon, we apply GEN^{2}OUT to `optdigits` dataset which contains the feature representation of numerical digits.



(a) Data heatmap (b) X-Ray plot

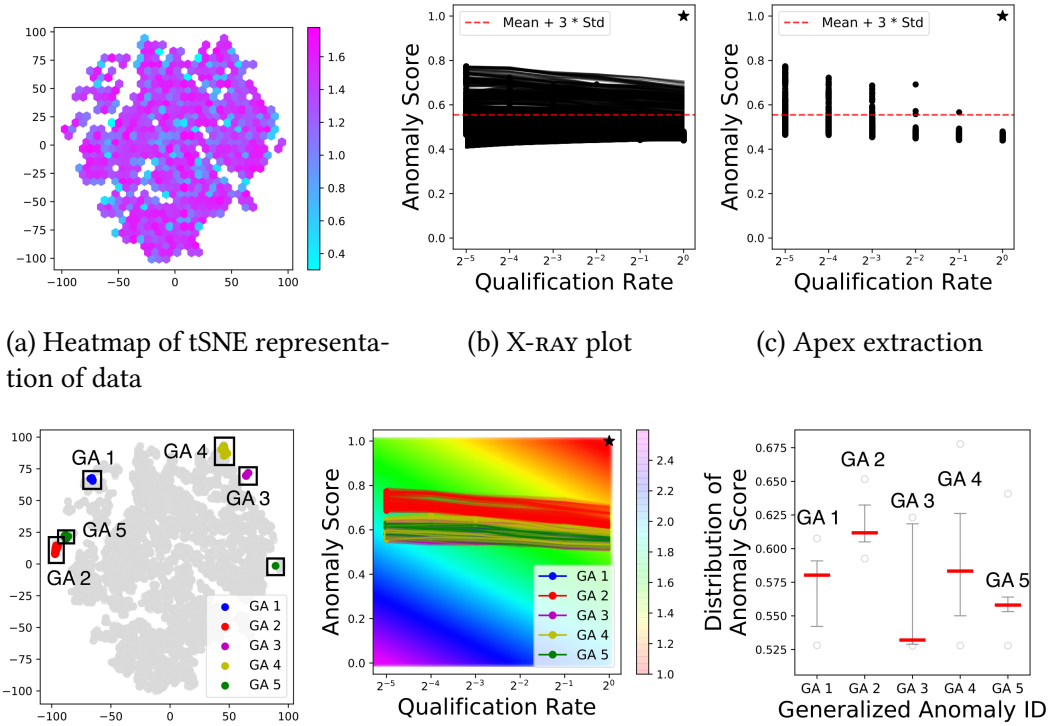
Figure 11.10: **GEN^{2}OUT raises no false alarms.** It correctly flags no anomalies in the `optdigits` dataset.

To visualize the dataset, we embed the points in two dimensional space using tSNE [dMH08] as shown in Figure 11.10a. It is a balanced dataset, where we have equal number of points for each digit, hence no group is present. X-RAY plot (Figure 11.10b) shows that all the score trajectories are below 0.5 (scores close to 1 are anomalous) with mean score at 0.36 in full dataset. Hence, we do not find group and correctly so.

11.6.2 Attention Routing in Medicine

We apply GEN^{2}OUT on EEG recordings for the epileptic patient (PT1) – PT1 suffered through onset of two seizures in our recording clips; our motivating application. We extract four simple statistical measures from the subsequences of the time series features, namely mean, variance, skewness and kurtosis, by sliding a thirty minute window with two minutes overlap. Figure 11.11a shows 2-dimensional tSNE representation of the data.

We then compute the generalized anomaly scores over time (within each window) for each detected group. Since the scores generated by GEN^{2}OUT are comparable, we draw attention to the most anomalous time point, where the seizures occurred as the detected groups correspond to seizure time period. The steps of GEN^{2}OUT are illustrated in Figure 11.11 when applied to multi-variate EEG data. Note that we find, several groups as shown in Figure 11.11. Of the detected groups, the group receiving highest score (GA2) is plotted over the raw voltage recordings over time for the patient. The group corresponds to the ground truth seizure duration (see Figure 11.1a). These time points that we direct attention to would assist the domain expert in decision making by alleviating cognitive load of examining all time points.



(a) Heatmap of tSNE representation of data

(b) X-RAY plot

(c) Apex extraction

(d) Outlier grouping

(e) Anomaly iso-curves

(f) Scoring

Figure 11.11: **GEN^{2}OUT detects seizures in real-world EEG data.** It assigns highest anomaly score to group anomaly GA2 that corresponds to seizures as we show in Figure 11.1a.

11.7 Conclusion

We presented GEN^{2}OUT, a principled anomaly detection algorithm with following properties:

- **Principled and Sound:** We propose five axioms that GEN^{2}OUT obeys them, in contrast to top competitors.
- **Doubly-General:** Propose doubly general – simultaneously detects point and group anomalies – GEN^{2}OUT. It does not require information on group structure, and ranks detected groups of varying sizes in order of their anomalousness.
- **Scalable:** Linear on the input size; requires minutes on 1M dataset on a stock machine.
- **Effective**: Applied on real-world data (Figure 11.1 and 11.7), GEN^{2}OUT wins in most cases over 27 benchmark datasets for point anomaly detection, and agrees with ground truth on seizure detection as well as group detection tasks.

Chapter 12

Conclusion and Future Work

In this thesis, we propose various explainable methods to address the limitations of black-box machine learning methods. These methods are either inherently explainable, or provide explanations of the data or decision-making process to users. Specifically, we focus on developing algorithms and solving applications related to graph and time series data. The algorithms we developed aim to address fundamental ML problems, and the applications we solved leverage domain-specific insights. We summarize the contributions of our work below.

12.1 Contributions

Part I: *Node*-Level Graph Mining

In Part I, we focus on the problem where only a single graph is given and solve node-level graph tasks such as node classification and link prediction. Our proposed methods perform well, and explain their performance by identifying which information (e.g., graph structure or node features) is useful to the task.
We propose three algorithms:

- **Network Effects Detection (Chapter 3):** NetEffect identifies whether network effects exist in the given graph (i.e., homophily, heterophily, both, or none) and precisely estimates the underlying compatibility matrix. In node classification, NetEffect is *12.9%* more accurate and *3.4*× faster than its competitors.
- **Robust and Interpretable Node Classification (Chapter 4):** SlimG is a linear graph neural network (GNN) that addresses pain points of existing GNNs and is robust across all scenarios in attributed graphs (e.g. no network effects and useless features). SlimG is *10.3%* more accurate and *2.5*× faster than its competitors.
- **Network Usable Information (Chapter 5):** NetInfoF quantitatively measures usable information in attributed graphs for both node classification and link prediction. NetInfoF is the first linear GNN that generalizes to link prediction and achieves an *average rank 1.1* among its competitors.

We solve an application:

- **"Hybrid" Question Answering (Chapter 6):** HybGRAG is a retrieval-augmented generation (RAG) method designed to handle questions that require hybrid information, i.e. both relational and textual. HybGRAG achieves an average relative improvement of *51%* over the best competitor.

Part II: *Graph*-Level Graph Mining

In Part II, we focus on the problem where a graph database with multiple graphs is given and solve graph-level graph tasks such as graph anomaly detection and graph regression. Our proposed methods detect explainable substructures that are frequently shared among graphs in the database and leverage them to solve the downstream task.
We propose two algorithms:

- **Anomaly Detection with Frequent Substructures (Chapter 7):** GAWD identifies anomalies in a graph database by compressing the graphs with frequent substructures using the minimum description length (MDL) principle. GAWD is up to *58*× faster, while being *1.3*× better in average precision.
- **Learnable Graph Kernel for Descriptive Features (Chapter 8):** RWK$^+$ enhances the random walk graph kernel and extracts descriptive substructure features from the graph database. A model using the features extracted by RWK$^+$ outperforms the baseline by *14.3%* in mean absolute error on a large graph regression benchmark.

We solve an application:

- **Human Trafficking Detection (Chapter 9):** DeltaShield incrementally detects templates among text documents by representing them as graphs and applying the MDL principle. DeltaShield detects human-trafficking advertisements with *84%* precision, while requiring only *8 hours* for *4 million* documents.

Part III: Time Series Mining

In Part III, we focus on the problem of detecting sequence-level and point-level anomalies in time series data. Our proposed methods not only detect anomalies in the time series, but also explain them by identifying the group behavior shared among the anomalies.
We propose an algorithm:

- **Self-Supervised Anomaly Detection (Chapter 10):** TSAP automatically fine-tunes the best hyperparameters for creating pseudo-time-series anomalies, to learn the anomaly detector in a self-supervised manner. TSAP achieves an average rank *2.2* in F1 score and identifies the true hyperparameters of anomalies.

We solve an application:

- **Seizure Detection in EEG Recordings (Chapter 11):** gen^{2}Out detects group anomalies (such as seizures) and point anomalies (such as noise) in time series EEG recordings. gen^{2}Out is the first to detect both group and point anomalies and takes only *2 minutes* to run on *1 million* data points.

12.2 Future Work

Explainable Graph Algorithms for Temporal Graphs

While linear GNNs have shown great effectiveness and explainability in solving node-level graph tasks such as node classification and link prediction, none of them focus on graphs that evolve over time. While tensor decomposition [PFS17, dARF17] is able to address the time-evolving graph structure, it is not yet clear how to handle temporal node features. Moreover, as the behavior of the nodes changes over time, some of their features may shift from being useful to becoming useless. Although many nonlinear GNNs have been proposed for temporal graphs [RCF+20, ZZN+22], they generally lack the explainability offered by linear GNNs. Extending linear GNNs to handle temporal graphs is thus an important direction for future research.

Learnable Graph Kernels with Edge Features

While learnable graph kernels are powerful feature extractors for graph databases, most of them do not take edge features into account. However, edge features can be crucial for graph-level graph tasks. For example, in a database containing protein-protein association graphs [SGL+19], edge features contain meaningful association information between proteins. Although some graph kernels support edge features [KJM20], incorporating learnability into them has not yet been studied. We believe that addressing this limitation can further improve the performance and interpretability of learnable graph kernels.

Multi-Modal Explainable Anomaly Detection in Time Series

While explaining anomalies in time series using abnormal patterns is straightforward, explaining them using text is even simpler. For example, in the real world, an EEG signal can be paired with a physician report [OP16]. The physician describes the normal or abnormal periods observed in the signal in the report. In this case, an algorithm capable of pairing physician descriptions with specific time periods in the signal is desired. This algorithm has the potential to significantly reduce the time required for physicians to diagnose patients. Although multi-modal data has been utilized to improve downstream performance [QHZ+23], how it can be leveraged to enhance explainability has not been well explored. Therefore, leveraging text to explain detected anomalies in time series is an impactful future direction.

12.3 Closing Thoughts

In this thesis, we focus on developing explainable algorithms, as well as solving applications while providing insightful explanations, for graph and time series data. While developing machine learning algorithms lays the foundation for explainable artificial intelligence (XAI), solving real-world applications through multidisciplinary collaboration with domain experts has the potential to directly advance human well-being. Although this thesis focuses on graph and time series data, the significance of these directions can be generalized to other data types, such as text and images. We believe that these directions will drive impactful future research.

Bibliography

[AA13] Faraz Ahmed and Muhammad Abulaish. A generic statistical approach for spam detection in online social networks. *Comput. Commun.*, 36(10-11):1120–1129, 2013.

[ABKS99] Mihael Ankerst et al. OPTICS: ordering points to identify the clustering structure. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 49–60. ACM Press, 1999.

[ABLS07] Noga Alon et al. Non-backtracking random walks mix faster. *Communications in Contemporary Mathematics*, 9(04):585–603, 2007.

[ACB17] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 2017.

[ACL06] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Local graph partitioning using pagerank vectors. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 475–486. IEEE Computer Society, 2006.

[Agg13] Charu C. Aggarwal. *Outlier Analysis*. Springer, 2013.

[AJ18] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7786–7795, 2018.

[ALB23] Mohammad Sadegh Akhondzadeh, Vijay Lingam, and Aleksandar Bojchevski. Probing graph representations. In *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, volume 206 of *Proceedings of Machine Learning Research*, pages 11630–11649. PMLR, 2023.

[AMF10] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. oddball: Spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II*, volume 6119 of *Lecture Notes in Computer Science*, pages 410–421. Springer, 2010.

[AMG+20] Julien Audibert et al. USAD: unsupervised anomaly detection on multivariate time series. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3395–3404. ACM, 2020.

[APK+19] Sami Abu-El-Haija et al. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 21–29. PMLR, 2019.

[APZ19] Rami Al-Rfou, Bryan Perozzi, and Dustin Zelle. DDGK: learning graph representations for deep divergence graph kernels. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 37–48. ACM, 2019.

[ARL+18] Nesreen K. Ahmed et al. Learning role-based graph embeddings. *CoRR*, abs/1802.02896, 2018.

[ASS17] Hamidreza Alvari, Paulo Shakarian, and J. E. Kelly Snyder. Semi-supervised learning for detecting human trafficking. *Secur. Informatics*, 6(1):1, 2017.

[AST+24] Abdul Fatir Ansari et al. Chronos: Learning the language of time series. *CoRR*, abs/2403.07815, 2024.

[ATK15] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Min. Knowl. Discov.*, 29(3):626–688, 2015.

[AWW+24] Akari Asai et al. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[BAY22] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[BBR+18] Mohamed Ishmael Belghazi et al. Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 530–539. PMLR, 2018.

[BCML22] Ane Blázquez-García et al. A review on outlier/anomaly detection in time series data. *ACM Comput. Surv.*, 54(3):56:1–56:33, 2022.

[BDG95] Sergey Brin, James Davis, and Hector Garcia-Molina. Copy detection mechanisms for digital documents. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA, May 22-25, 1995*, pages 398–409. ACM Press, 1995.

[BFZB23] Giorgos Bouritsas et al. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(1):657–668, 2023.

[BGJM17] Piotr Bojanowski et al. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146, 2017.

[BH21] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[BHA+21] Rishi Bommasani et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

[BHX+22] Alexei Baevski et al. data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 1298–1312. PMLR, 2022.

[BJR17] Robert A. Bridges, Jessie D. Jamieson, and Joel W. Reed. Setting the threshold for high throughput detectors: A mathematical approach for ensembles of dynamic, heterogeneous, probabilistic anomaly detectors. In *2017 IEEE International Conference on Big Data (IEEE BigData 2017), Boston, MA, USA, December 11-14, 2017*, pages 1071–1078. IEEE Computer Society, 2017.

[BJRL15] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[BK05] Karsten M. Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA*, pages 74–81. IEEE Computer Society, 2005.

[BKEF12] Michele Berlingerio et al. Netsimile: A scalable approach to size-independent network similarity. *CoRR*, abs/1209.2684, 2012.

[BKK18] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.

[BKNS00] Markus M. Breunig et al. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 93–104. ACM, 2000.

[BKS93] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient processing of spatial joins using r-trees. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*, pages 237–246. ACM Press, 1993.

[BL03] Regina Barzilay and Lillian Lee. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003, Edmonton, Canada, May 27 - June 1, 2003*. The Association for Computational Linguistics, 2003.

[BMR+20] Tom B. Brown et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information*

Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[BS87] Geoffrey J. Barton and Michael J.E. Sternberg. A strategy for the rapid multiple alignment of protein sequences: Confidence levels from tertiary structure comparisons. *Journal of Molecular Biology*, 198(2):327–337, 1987.

[BS88] Michael F. Barnsley and Alan D. Sloan. A better way to compress images. *BYTE*, 13(1):215–223, January 1988.

[BZA21] Azzedine Boukerche, Lining Zheng, and Omar Alfandi. Outlier detection: Methods, models, and classification. *ACM Comput. Surv.*, 53(3):55:1–55:37, 2021.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.

[CCG+24] Jie Chen et al. Exploiting neighbor effect: Conv-agnostic GNN framework for graphs with heterophily. *IEEE Trans. Neural Networks Learn. Syst.*, 35(10):13383–13396, 2024.

[CCVB20] Zhengdao Chen et al. Can graph neural networks count substructures? In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[CG10] Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 255–262. Omnipress, 2010.

[CH94] Diane J. Cook and Lawrence B. Holder. Substructure discovery using minimum description length and background knowledge. *J. Artif. Intell. Res.*, 1:231–255, 1994.

[CJM20] Dexiong Chen, Laurent Jacob, and Julien Mairal. Convolutional kernel networks for graph-structured data. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1576–1586. PMLR, 2020.

[CLM+17] Gari D. Clifford et al. AF classification from a short single lead ECG recording: the physionet computing in cardiology challenge 2017. In *Computing in Cardiology, CinC 2007, Rennes, France, September 24-27, 2017*. www.cinc.org, 2017.

[CMB+21] Luca Cosmo et al. Graph kernel neural networks. *CoRR*, abs/2112.07436, 2021.

[CPLM21] Eli Chien et al. Adaptive universal generalized pagerank graph neural network. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[CPMF04] Deepayan Chakrabarti et al. Fully automatic cross-associations. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 79–88. ACM, 2004.

[CPP+16] Stefano Cresci et al. Dna-inspired online behavioral modeling and its application to spambot detection. *IEEE Intell. Syst.*, 31(5):58–64, 2016.

[CPP+17] Stefano Cresci et al. The paradigm-shift of social spambots: Evidence, theories, and tools for the arms race. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017*, pages 963–972. ACM, 2017.

[CSL21] Hyunsoo Cho, Jinseok Seol, and Sang-goo Lee. Masked contrastive learning for anomaly detection. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 1434–1441. ijcai.org, 2021.

[CSTI09] Niranjan Chakravarthy et al. Controlling epileptic seizures in a neural mass model. *J. Comb. Optim.*, 17(1):98–116, 2009.

[CTC18] Raghavendra Chalapathy, Edward Toth, and Sanjay Chawla. Group anomaly detection using deep generative models. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part I*, volume 11051 of *Lecture Notes in Computer Science*, pages 173–189. Springer, 2018.

[Cut13] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2292–2300, 2013.

[CWH+20] Ming Chen et al. Simple and deep graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1725–1735. PMLR, 2020.

[CWL+18] Wei Cao et al. BRITS: bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6776–6786, 2018.

[CYL+18] Zheng Chen et al. Correlated anomaly detection from large streaming data. In *IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018*, pages 982–992. IEEE, 2018.

[CZ04] Edwin KP Chong and Stanislaw H Zak. *An Introduction to Optimization*. John Wiley & Sons, 2004.

[dARF17] Miguel Ramos de Araujo, Pedro Manuel Pinto Ribeiro, and Christos Faloutsos. Tensorcast: Forecasting with context using coupled tensors (best paper award). In *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, pages 71–80. IEEE Computer Society, 2017.

[DBH18] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3837–3845, 2016.

[DCLT19] Jacob Devlin et al. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[DFP+24] Jialin Dong et al. Don't forget to connect! improving RAG with graph-based reranking. *CoRR*, abs/2405.18414, 2024.

[DH04] Chris H. Q. Ding and Xiaofeng He. Principal component analysis and effective k-means clustering. In *Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004*, pages 497–501. SIAM, 2004.

[DHS+19] Simon S. Du et al. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5724–5734, 2019.

[DJL+23] Vijay Prakash Dwivedi et al. Benchmarking graph neural networks. *J. Mach. Learn. Res.*, 24:43:1–43:48, 2023.

[DK23] Mingze Dong and Yuval Kluger. Towards understanding and reducing graph structural noise for gnns. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 8202–8226. PMLR, 2023.

[DLB+25] Boje Deforce et al. End-to-end self-tuning self-supervised time series anomaly detection. In *Proceedings of the 2025 SIAM International Conference on Data Mining*, pages 568–577. SIAM, 2025.

[DLMR11] Michael Davis et al. Detecting anomalies in graphs with numeric labels. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 1197–1202. ACM, 2011.

[dMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

[DVF+16] Clayton Allen Davis et al. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 273–274. ACM, 2016.

[DWCL22] Kaize Ding et al. Meta propagation networks for graph few-shot semi-supervised learning. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 6524–6531. AAAI Press, 2022.

[DZA24] Xueying Ding, Yue Zhao, and Leman Akoglu. Fast unsupervised deep outlier model selection with hypernetworks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 585–596. ACM, 2024.

[ECP+19] Saeideh Shahrokh Esfahani et al. Context-specific language modeling for human trafficking detection from online advertisements. In *ACL (1)*, pages 1180–1184. Association for Computational Linguistics, 2019.

[ECS22] Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding dataset difficulty with V-usable information. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR, 2022.

[EDD+15] Andrew Emmott et al. Systematic construction of anomaly detection benchmarks from real data. *CoRR*, abs/1503.01158, 2015.

[EFGM18] Dhivya Eswaran et al. Spotlight: Detecting anomalies in streaming graphs. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 1378–1386. ACM, 2018.

[EFMN19] Dhivya Eswaran et al. Intervention-aware early warning. In *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*, pages 1030–1035. IEEE, 2019.

[EH07] William Eberle and Lawrence B. Holder. Discovering structural anomalies in graph-based data. In *Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 393–398. IEEE Computer Society, 2007.

[EK13] Sara Elmanarelbouanani and Ismail Kassou. Authorship analysis studies: A survey. *Int. Journal of Computer Applications*, 86, 12 2013.

[EKF20] Dhivya Eswaran, Srijan Kumar, and Christos Faloutsos. Higher-order label homogeneity and spreading in graphs. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 2493–2499. ACM / IW3C2, 2020.

[EKSX96] Martin Ester et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231. AAAI Press, 1996.

[ETC+24] Darren Edge et al. From local to global: A graph RAG approach to query-focused summarization. *CoRR*, abs/2404.16130, 2024.

[FKP+13] Aasa Feragen et al. Scalable kernels for graphs with continuous attributes. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 216–224, 2013.

[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.

[FTW+22] Shangbin Feng et al. Twibot-22: Towards graph-based twitter bot detection. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[FWV06] Damien François, Vincent Wertz, and Michel Verleysen. The permutation test for feature selection by mutual information. In *14th European Symposium on Artificial Neural Networks, ESANN 2006, Bruges, Belgium, April 26-28, 2006, Proceedings*, pages 239–244, 2006.

[FX22] Yiwei Fu and Feng Xue. MAD: self-supervised masked anomaly detection task for multivariate time series. In *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*, pages 1–8. IEEE, 2022.

[FYWT22] Aosong Feng et al. Kergnns: Interpretable graph neural networks with graph kernels. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 6614–6622. AAAI Press, 2022.

[GC23] Azul Garza and Max Mergenthaler Canseco. Timegpt-1. *CoRR*, abs/2310.03589, 2023.

[GCS+15] Maria Giatsoglou et al. Nd-sync: Detecting synchronized fraud activities. In *Advances in Knowledge Discovery and Data Mining - 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings, Part II*, volume 9078 of *Lecture Notes in Computer Science*, pages 201–214. Springer, 2015.

[GDP+23] Luyu Gao et al. RARR: researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 16477–16508. Association for Computational Linguistics, 2023.

[GE18] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9781–9791, 2018.

[GES+18] Nikhil Gupta et al. Beyond outlier detection: Lookout for pictorial explanation. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part I*, volume 11051 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2018.

[GFW03] Thomas Gärtner, Peter A. Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 2003.

[GGAH14] Manish Gupta et al. Outlier detection for temporal data: A survey. *IEEE Trans. Knowl. Data Eng.*, 26(9):2250–2267, 2014.

[GGKF15] Wolfgang Gatterbauer et al. Linearized and single-pass belief propagation. *Proc. VLDB Endow.*, 8(5):581–592, 2015.

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM, 2016.

[Gle19] Nathaniel Gleicher. *How We Respond to Inauthentic Behavior on Our Platforms: Policy Update*, 2019.

[GLT+20] Kelvin Guu et al. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR, 2020.

[GMB+19] Asim Ghosh et al. Quantifying gender preferences in human social interactions using a large cellphone dataset. *EPJ Data Sci.*, 8(1):9:1–9:15, 2019.

[GMRS16] Sudipto Guha et al. Robust random cut forest based anomaly detection on streams. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2712–2721. JMLR.org, 2016.

[Grü07] Peter D. Grünwald. *The Minimum Description Length Principle*. MIT press, 2007.

[GSG+24] Zhibin Gou et al. CRITIC: large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[GSR+17] Justin Gilmer et al. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[GSW+20] Jingkun Gao et al. Robusttad: Robust time series anomaly detection via decomposition and convolutional neural networks. *CoRR*, abs/2002.09545, 2020.

[Gut84] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, USA, June 18-21, 1984*, pages 47–57. ACM Press, 1984.

[HA85] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[Har54] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[HFZ+20] Weihua Hu et al. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[HGL+11] Keith Henderson et al. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 663–671. ACM, 2011.

[HHA24] Hadi Hojjati, Thi Kieu Khanh Ho, and Narges Armanfard. Self-supervised anomaly detection in computer vision and beyond: A survey and outlook. *Neural Networks*, 172:106106, 2024.

[HHS+21] Qian Huang et al. Combining label propagation and simple models out-performs graph neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[HJ15] Tamir Hazan and Tommi S. Jaakkola. Steps toward deep kernel methods from infinite neural networks. *CoRR*, abs/1508.05133, 2015.

[HK09] Shohei Hido and Hisashi Kashima. A linear-time graph kernel. In *ICDM 2009, The Ninth IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, pages 179–188. IEEE Computer Society, 2009.

[HLT+19] Yu Huang et al. Robust sensor-based human activity recognition with snippet consensus neural networks. In *16th IEEE International Conference on Wearable and Implantable Body Sensor Networks, BSN 2019, Chicago, IL, USA, May 19-22, 2019*, pages 1–4. IEEE, 2019.

[HLZ+24] Yuntong Hu et al. GRAG: graph retrieval-augmented generation. *CoRR*, abs/2405.16506, 2024.

[HPPI18] Timothy Hutson et al. Predictability and resetting in a case of convulsive status epilepticus. *Frontiers in Neurology*, 9:172, 2018.

[HTS+24] Xiaoxin He et al. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[HXD03] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognit. Lett.*, 24(9-10):1641–1650, 2003.

[HYL17] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[IMN+18] Vassilis N Ioannidis et al. Kernel-based inference of functions over graphs. In *Adaptive Learning Methods for Nonlinear System Modeling*, pages 173–198. Elsevier, 2018.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[JLJZ16] Biao Jie et al. Sub-network based kernels for brain network classification. In *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB 2016, Seattle, WA, USA, October 2-5, 2016*, pages 622–629. ACM, 2016.

[Jon04] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502, 2004.

[JSR+21] Alon Jacovi et al. Contrastive explanations for model interpretability. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1597–1611. Association for Computational Linguistics, 2021.

[JXZ+24] Bowen Jin et al. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 163–184. Association for Computational Linguistics, 2024.

[JYST22] Yang Jiao et al. Timeautoad: Autonomous anomaly detection with self-supervised contrastive loss for multivariate time series. *IEEE Trans. Netw. Sci. Eng.*, 9(3):1604–1619, 2022.

[KBG19] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[KDS+17] Mayank Kejriwal et al. Flagit: A system for minimally supervised human trafficking indicator mining. *CoRR*, abs/1712.03086, 2017.

[KG16] Vani Kanjirangat and Deepa Gupta. A study on extrinsic text plagiarism detection techniques and tools. *Journal of Engineering Science and Technology Review*, 9:150–164, 10 2016.

[KJM20] Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris. A survey on graph kernels. *Appl. Netw. Sci.*, 5(1):6, 2020.

[KKK+11] Danai Koutra et al. Unifying guilt-by-association approaches: Theorems and fast algorithms. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2011, Athens, Greece, September 5-9, 2011, Proceedings, Part II*, volume 6912 of *Lecture Notes in Computer Science*, pages 245–260. Springer, 2011.

[KLF+17] Sun-Hee Kim et al. Daps: Mining using change-point detection of epileptic activity time series data. *J. Inf. Sci. Eng.*, 33(2):517–536, 2017.

[KLKF14] U Kang et al. Net-ray: Visualizing and mining billion-scale graphs. In *Advances in Knowledge Discovery and Data Mining - 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I*, volume 8443 of *Lecture Notes in Computer Science*, pages 348–361. Springer, 2014.

[KLR04] Eamonn J. Keogh, Stefano Lonardi, and Chotirat (Ann) Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 206–215. ACM, 2004.

[KMH+20] Jared Kaplan et al. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[KO21] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[KOM+20] Vladimir Karpukhin et al. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics, 2020.

[Kri22] Nils M. Kriege. Weisfeiler and leman go walking: Random walk kernels revisited. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[KRT22] Rayan Krishnan, Pranav Rajpurkar, and Eric J Topol. Self-supervised learning in medicine and healthcare. *Nature Biomed. Eng.*, 6(12):1346–1352, 2022.

[KSG04] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E*, 69(6):066138, 2004.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.

[KSZ08] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD Inter-*

national Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008, pages 444–452. ACM, 2008.

[KTI03] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 321–328. AAAI Press, 2003.

[KTS12] U Kang, Hanghang Tong, and Jimeng Sun. Fast random walk graph kernel. In *Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012*, pages 828–838. SIAM / Omnipress, 2012.

[Kul21] Aayushi Kulshrestha. *Detection of Organized Activity in Online Escort Advertisements*. McGill University (Canada), 2021.

[KVF13] Danai Koutra, Joshua T. Vogelstein, and Christos Faloutsos. DELTACON: A principled massive-graph similarity function. In *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA*, pages 162–170. SIAM, 2013.

[KVW+15] B. Krishnan et al. Epileptic focus localization based on resting state interictal meg recordings is feasible irrespective of the presence or absence of spikes. *Clinical Neurophysiology*, 126(4):667–674, 2015.

[KW16] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *CoRR*, abs/1611.07308, 2016.

[KW17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[KWG19] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13333–13345, 2019.

[KY04] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004, Proceedings*, volume 3201 of *Lecture Notes in Computer Science*, pages 217–226. Springer, 2004.

[LB21] Derek Lim and Austin R. Benson. Expertise and dynamics within crowdsourced musical knowledge curation: A case study of the genius platform. In *Proceedings of the Fifteenth International AAAI Conference on Web and Social Media, ICWSM 2021, held virtually, June 7-10, 2021*, pages 373–384. AAAI Press, 2021.

[LCW+18] Jundong Li et al. Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6):94:1–94:45, 2018.

[LGS02] Christopher J. Lee, Catherine S. Grasso, and Mark F. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinform.*, 18(3):452–464, 2002.

[LGW+22] Mingjie Li et al. G^2cn: Graph gaussian convolution networks with concentrated graph filters. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 12782–12796. PMLR, 2022.

[LHG+24] Shilong Li et al. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 12758–12786. Association for Computational Linguistics, 2024.

[LHJ+23] Yunshi Lan et al. Complex knowledge base question answering: A survey. *IEEE Trans. Knowl. Data Eng.*, 35(11):11196–11215, 2023.

[LHL+21] Derek Lim et al. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 20887–20902, 2021.

[LHL+22] Sitao Luan et al. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[LHX+23] Sitao Luan et al. When do graph neural networks help with node classification? investigating the homophily principle on node distinguishability. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[LHY+19] Meng-Chieh Lee et al. Deepidentifier: A deep learning-based lightweight approach for user identity recognition. In *Advanced Data Mining and Applications - 15th International Conference, ADMA 2019, Dalian, China, November 21-23, 2019, Proceedings*, volume 11888 of *Lecture Notes in Computer Science*, pages 389–405. Springer, 2019.

[Lip18] Zachary C. Lipton. The mythos of model interpretability. *ACM Queue*, 16(3):30, 2018.

[LJBJ17] Tao Lei et al. Deriving neural architectures from sequence and graph kernels. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2024–2033. PMLR, 2017.

[LJS13] Mingfeng Lin, Henry C. Lucas Jr., and Galit Shmueli. Research commentary - too big to fail: Large samples and the p-value problem. *Inf. Syst. Res.*, 24(4):906–917, 2013.

[LJWS21] Qingqing Long et al. Theoretically improving graph neural networks via anonymous walk graph kernels. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 1204–1214. ACM / IW3C2, 2021.

[LK14] Jure Leskovec and Andrej Krevl. Snap datasets: Stanford large network dataset collection, 2014.

[LL17] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[LLH+24] Nelson F. Liu et al. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173, 2024.

[LLZ+24] Zhuowan Li et al. Retrieval augmented generation or long-context llms? A comprehensive study and hybrid approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024*, pages 881–893. Association for Computational Linguistics, 2024.

[LM14] Quoc V. Le and Tomás Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org, 2014.

[LMTG19] Guohao Li et al. Deepgcns: Can gcns go as deep as cnns? In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9266–9275. IEEE, 2019.

[LNB+21] Meng-Chieh Lee et al. GAWD: graph anomaly detection in weighted directed graph databases. In *ASONAM '21: International Conference on Advances in Social Networks Analysis and Mining, Virtual Event, The Netherlands, November 8 - 11, 2021*, pages 143–150. ACM, 2021.

[Loa00] Charles F Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1-2):85–100, 2000.

[LP11] Lei Li and B. Aditya Prakash. Time series clustering: Complex is simpler! In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 185–192. Omnipress, 2011.

[LPF10] Lei Li, B. Aditya Prakash, and Christos Faloutsos. Parsimonious linear fingerprinting for time series. *Proc. VLDB Endow.*, 3(1):385–396, 2010.

[LPP+20] Patrick S. H. Lewis et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[LR94] Ming-Ling Lo and Chinya V. Ravishankar. Spatial joins using seeded trees. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, USA, May 24-27, 1994*, pages 209–220. ACM Press, 1994.

[LSF+21] Meng-Chieh Lee et al. Gen^{2}out: Detecting and ranking generalized anomalies. In *2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021*, pages 801–811. IEEE, 2021.

[LSL+18] Lin Li et al. Detection and characterization of human trafficking networks using unsupervised scalable text template matching. In *IEEE International Conference on Big Data (IEEE BigData 2018), Seattle, WA, USA, December 10-13, 2018*, pages 3111–3120. IEEE, 2018.

[LSYF24] Meng-Chieh Lee et al. NETEFFECT: discovery and exploitation of generalized network effects. In *Advances in Knowledge Discovery and Data Mining - 28th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2024, Taipei, Taiwan, May 7-10, 2024, Proceedings, Part I*, volume 14645 of *Lecture Notes in Computer Science*, pages 299–312. Springer, 2024.

[LSYP21] Chun-Liang Li et al. Cutpaste: Self-supervised learning for anomaly detection and localization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 9664–9674. Computer Vision Foundation / IEEE, 2021.

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 413–422. IEEE Computer Society, 2008.

[LTZ12] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1):3:1–3:39, 2012.

[LVK+21] Meng-Chieh Lee et al. INFOSHIELD: generalizable information-theoretic human-trafficking detection. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, pages 1116–1127. IEEE, 2021.

[LW68] Andrei Leman and Boris Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968.

[LWJ22] Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(12):10270–10276, 2022.

[LYZ+24] Meng-Chieh Lee et al. Netinfof framework: Measuring and exploiting network usable information. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[LZA24] Meng-Chieh Lee, Lingxiao Zhao, and Leman Akoglu. Descriptive kernel convolution network with improved random walk kernel. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 457–468. ACM, 2024.

[LZH+23] Xiao Liu et al. Self-supervised learning: Generative or contrastive. *IEEE Trans. Knowl. Data Eng.*, 35(1):857–876, 2023.

[LZM+25] Meng-Chieh Lee et al. Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases. In *Proceedings of the 63nd Annual Meeting of the Association for Computational Linguistics*, 2025.

[LZW+20] Meng-Chieh Lee et al. Autoaudit: Mining accounting and time-evolving graphs. In *2020 IEEE International Conference on Big Data (IEEE BigData 2020), Atlanta, GA,*

USA, December 10-13, 2020, pages 950–956. IEEE, 2020.

[LZX+21] Peiyuan Liao et al. Information obfuscation of graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6600–6610. PMLR, 2021.

[Mai16] Julien Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1399–1407, 2016.

[MAR96] Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. SLIQ: A fast scalable classifier for data mining. In *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings*, volume 1057 of *Lecture Notes in Computer Science*, pages 18–32. Springer, 1996.

[MBSL19] Haggai Maron et al. Provably powerful graph networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2153–2164, 2019.

[MCCD13] Tomás Mikolov et al. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[MCJ+23] Haitao Mao et al. Demystifying structural disparity in graph neural networks: Can one size fit all? In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[MFHdC14] Vítor T. Martins et al. Plagiarism detection: A tool survey and comparison. In *3rd Symposium on Languages, Applications and Technologies, SLATE 2014, June 19-20, 2014 - Bragança, Portugal*, volume 38 of *OASIcs*, pages 143–158. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

[MHA17] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017.

[Mil21] Tim Miller. Contrastive explanation: a structural-model approach. *Knowl. Eng. Rev.*, 36:e14, 2021.

[MK24] Costas Mavromatis and George Karypis. GNN-RAG: graph neural retrieval for large language model reasoning. *CoRR*, abs/2405.20139, 2024.

[MKB+20] Christopher Morris et al. Tudataset: A collection of benchmark datasets for learning with graphs. *CoRR*, abs/2007.08663, 2020.

[MKHS14] Julien Mairal et al. Convolutional kernel networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2627–2635,

2014.

[MKKM16] Christopher Morris et al. Faster kernels for graphs with continuous attributes via hashing. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 1095–1100. IEEE Computer Society, 2016.

[MLST22] Yao Ma et al. Is homophily a necessity for graph neural networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[MMA16] Emaad A. Manzoor, Sadegh M. Milajerdi, and Leman Akoglu. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1035–1044. ACM, 2016.

[MRA+16] Pankaj Malhotra et al. Lstm-based encoder-decoder for multi-sensor anomaly detection. *CoRR*, abs/1607.00148, 2016.

[MRF+19] Christopher Morris et al. Weisfeiler and leman go neural: Higher-order graph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4602–4609. AAAI Press, 2019.

[MS13] Krikamol Muandet and Bernhard Schölkopf. One-class support measure machines for group anomaly detection. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*. AUAI Press, 2013.

[MS21] Octavio César Mesner and Cosma Rohilla Shalizi. Conditional mutual information estimation for mixed, discrete and continuous data. *IEEE Trans. Inf. Theory*, 67(1):464–484, 2021.

[MSF14] Yasuko Matsubara, Yasushi Sakurai, and Christos Faloutsos. Autoplait: automatic mining of co-evolving time sequences. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 193–204. ACM, 2014.

[MT19] Jan Mielniczuk and Pawel Teisseyre. Stopping rules for mutual information-based feature selection. *Neurocomputing*, 358:255–274, 2019.

[MTG+23] Aman Madaan et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[MV09] Pierre Mahé and Jean-Philippe Vert. Graph kernels based on tree patterns for molecules. *Mach. Learn.*, 75(1):3–35, 2009.

[MZZA23] Martin Q. Ma et al. The need for unsupervised outlier model selection: A review and evaluation of internal evaluation strategies. *SIGKDD Explor.*, 25(1):19–35, 2023.

[NAa] NA. Human trafficking & online prostitution advertising. https://wagner.house.gov/Human%20Trafficking%20%26%20Online%20Prostitution%20Advertising. backapage.

[NAb] NA. Survivor survey r5. http://www.thorn.org/wp-content/uploads/2015/02/Survivor_Survey_r5.pdf. trafficking survivors.

[NC03] Caleb C. Noble and Diane J. Cook. Graph-based anomaly detection. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 631–636. ACM, 2003.

[NCV+17] Annamalai Narayanan et al. graph2vec: Learning distributed representations of graphs. *CoRR*, abs/1707.05005, 2017.

[NMBD17] Chirag Nagpal et al. An entity resolution approach to isolate instances of human trafficking online. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, pages 77–84. Association for Computational Linguistics, 2017.

[NMN+23] Supriya Nagesh et al. Explaining a machine learning decision to physicians via counterfactuals. In *Conference on Health, Inference, and Learning, CHIL 2023, Broad Institute of MIT and Harvard (Merkin Building), 415 Main Street, Cambridge, MA, USA*, volume 209 of *Proceedings of Machine Learning Research*, pages 556–577. PMLR, 2023.

[NMT+18] Giannis Nikolentzos et al. Kernel graph convolutional neural networks. In *Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I*, volume 11139 of *Lecture Notes in Computer Science*, pages 22–32. Springer, 2018.

[NSV21] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: A survey. *J. Artif. Intell. Res.*, 72:943–1027, 2021.

[NTS18] Nicolò Navarin, Dinh Van Tran, and Alessandro Sperduti. Pre-training graph neural networks with kernels. *CoRR*, abs/1811.06930, 2018.

[NV20] Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[Off] International Labour Office. Ilo global estimate of forced labor. https://www.ilo.org/sites/default/files/wcmsp5/groups/public/@ed_norm/@declaration/documents/publication/wcms_182004.pdf. 2012.

[OP09] Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Soc. Networks*, 31(2):155–163, 2009.

[OP16] Iyad Obeid and Joseph Picone. The temple university hospital eeg data corpus. *Frontiers in neuroscience*, 10:196, 2016.

[Ope23] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[Pan03] Liam Paninski. Estimation of entropy and mutual information. *Neural Comput.*, 15(6):1191–1253, 2003.

[Pev16] Tomás Pevný. Loda: Lightweight on-line detector of anomalies. *Mach. Learn.*, 102(2):275–304, 2016.

[PFS17] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Trans. Intell. Syst. Technol.*, 8(2):16:1–16:44, 2017.

[PHD+17] Rebecca S. Portnoff et al. Backpage and bitcoin: Uncovering human traffickers. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 1595–1604. ACM, 2017.

[Pin64] Mark S Pinsker. Information and information stability of random variables and processes. *Holden-Day*, 1964.

[PIP+24] Debjit Paul et al. REFINER: reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024 - Volume 1: Long Papers, St. Julian's, Malta, March 17-22, 2024*, pages 1100–1126. Association for Computational Linguistics, 2024.

[PKBP23] Oleg Platonov et al. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[Prz07] Natasa Przulj. Biological network comparison using graphlet degree distribution. *Bioinform.*, 23(2):177–183, 2007.

[PSLF24] Lalithsai Posam et al. Difffind: Discovering differential equations from time series. In *Advances in Knowledge Discovery and Data Mining - 28th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2024, Taipei, Taiwan, May 7-10, 2024, Proceedings, Part VI*, volume 14650 of *Lecture Notes in Computer Science*, pages 175–187. Springer, 2024.

[PWC+20] Hongbin Pei et al. Geom-gcn: Geometric graph convolutional networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[PZL+24] Boci Peng et al. Graph retrieval-augmented generation: A survey. *CoRR*, abs/2408.08921, 2024.

[QHZ+23] Jielin Qiu et al. Can brain signals reveal inner alignment with human languages? In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 1789–1804. Association for Computational Linguistics, 2023.

[QPK+21] Chen Qiu et al. Neural transformation learning for deep anomaly detection beyond images. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8703–8714. PMLR, 2021.

[QZF+24] Shuofei Qiao et al. Autoact: Automatic agent learning from scratch for QA via self-planning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3003–3021. Association for Computational Linguistics, 2024.

[RAS21] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *J. Complex Networks*, 9(2), 2021.

[Ray16] Shebuti Rayana. Odds library, 2016.

[RBD18] Reihaneh Rabbany, David Bayani, and Artur Dubrawski. Active search of connections for case building and combating human trafficking. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2120–2129. ACM, 2018.

[RCF+20] Emanuele Rossi et al. Temporal graph networks for deep learning on dynamic graphs. *CoRR*, abs/2006.10637, 2020.

[RG03] Jan Ramon and Thomas Gärtner. Expressivity versus efficiency of graph kernels. In *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74, 2003.

[RG24] Matthew Renze and Erhan Guven. Self-reflection in LLM agents: Effects on problem-solving performance. *CoRR*, abs/2405.06682, 2024.

[RH07] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 410–420. ACL, 2007.

[Ris78] Jorma Rissanen. Modeling by shortest data description. *Autom.*, 14(5):465–471, 1978.

[Ris83] Jorma Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

[Ros14] Brian C Ross. Mutual information between discrete and continuous data sets. *PloS One*, 9(2):e87357, 2014.

[RPG+21] Aditya Ramesh et al. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual*

Event, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 2021.

[RRS00] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 427–438. ACM, 2000.

[RS10] Matthias Rupp and Gisbert Schneider. Graph kernels for molecular similarity. *Molecular Informatics*, 29(4):266–273, 2010.

[RS21] Benedek Rozemberczki and Rik Sarkar. Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings. *CoRR*, abs/2101.03091, 2021.

[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*, pages 1135–1144, 2016.

[RSG18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[Rud19] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5):206–215, 2019.

[RXW+19] Hansheng Ren et al. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 3009–3017. ACM, 2019.

[Sat20] Ryoma Sato. A survey on the expressive power of graph neural networks. *CoRR*, abs/2003.04078, 2020.

[SCG+23] Noah Shinn et al. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[SCM+23] Freda Shi et al. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR, 2023.

[SCSC03] Mei-Ling Shyu et al. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop*, pages 172–179. IEEE Press Piscataway, NJ, USA, 2003.

[SDS17] Nenad Stojanovic, Marko Dinic, and Ljiljana Stojanovic. A data-driven approach for multivariate contextualized anomaly detection: Industry use case. In *2017 IEEE*

International Conference on Big Data (IEEE BigData 2017), Boston, MA, USA, December 11-14, 2017, pages 1560–1569. IEEE Computer Society, 2017.

[SDS+21] Alessandro Sordoni et al. Decomposed mutual information estimation for contrastive representation learning. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9859–9869. PMLR, 2021.

[SF83] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Syst. Man Cybern.*, 13(3):353–362, 1983.

[SGL+19] Damian Szklarczyk et al. STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Res.*, 47(Database-Issue):D607–D613, 2019.

[She87] Roger N Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987.

[Sho09] S. Shorvon. *Epilepsy*. Oxford Neurology Library. OUP Oxford, 2009.

[SK03] Alexander J. Smola and Risi Kondor. Kernels and regularization on graphs. In *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777 of *Lecture Notes in Computer Science*, pages 144–158. Springer, 2003.

[SLBF17] Neil Shah et al. The many faces of link fraud. In *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*, pages 1069–1074. IEEE Computer Society, 2017.

[SMBG18] Oleksandr Shchur et al. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018.

[SNB+08] Prithviraj Sen et al. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008.

[SQJ+19] Karishma Sharma et al. Combating fake news: A survey on identification and mitigation techniques. *ACM Trans. Intell. Syst. Technol.*, 10(3):21:1–21:42, 2019.

[SRPE06] Siwei Shen et al. Adding syntax to dynamic programming for aligning comparable texts for the generation of paraphrases. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*. The Association for Computer Linguistics, 2006.

[SSE+17] Erich Schubert et al. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Trans. Database Syst.*, 42(3):19:1–19:21, 2017.

[SSvL+11] Nino Shervashidze et al. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12:2539–2561, 2011.

[SVP+09] Nino Shervashidze et al. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and*

Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009, volume 5 of *JMLR Proceedings*, pages 488–495. JMLR.org, 2009.

[SWP22] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.*, 15(9):1779–1797, 2022.

[SWS+99] Bernhard Schölkopf et al. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pages 582–588. The MIT Press, 1999.

[SXT+24] Jiashuo Sun et al. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[TC18] Edward Toth and Sanjay Chawla. Group deviation detection methods: A survey. *ACM Comput. Surv.*, 51(4):77:1–77:38, 2018.

[TI06] Kostas Tsakalis and Leonidas D. Iasemidis. Control aspects of a theoretical model for epileptic seizures. *Int. J. Bifurc. Chaos*, 16(7):2013–2027, 2006.

[TKG+22] Rylee Thompson et al. On evaluation metrics for graph generative models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[TMJS20] Jihoon Tack et al. CSI: novelty detection via contrastive learning on distributionally shifted instances. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[TMP11] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. Social structure of facebook networks. *CoRR*, abs/1102.2166, 2011.

[TSdC+19] Mariya Toneva et al. An empirical study of example forgetting during deep neural network learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[TSWY09] Jie Tang et al. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 807–816. ACM, 2009.

[TZJM17] Edmund Tong et al. Combating human trafficking with multimodal deep models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1547–1556. Association for Computational Linguistics, 2017.

[VCC+18] Petar Velickovic et al. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[VDH20] Sahil Verma, John P. Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *CoRR*, abs/2010.10596, 2020.

[VKT+17] Ioannis Vlachos et al. The concept of effective inflow: Application to interictal localization of the epileptogenic focus from ieeg. *IEEE Trans. Biomed. Eng.*, 64(9):2241–2252, 2017.

[VLK+23] Catalina Vajiac et al. Deltashield: Information theory for human- trafficking detection. *ACM Trans. Knowl. Discov. Data*, 17(2):28:1–28:27, 2023.

[WG20] Borui Wang and Geoffrey Gordon. Learning general latent-variable graphical models with predictive belief propagation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 6118–6126. AAAI Press, 2020.

[WGKM18] Tal Wagner et al. Semi-supervised learning on data streams via temporal label propagation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5082–5091. PMLR, 2018.

[WJZ+19] Felix Wu et al. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 2019.

[WMR17] Sandra Wachter, Brent D. Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *CoRR*, abs/1711.00399, 2017.

[WPC+21] Zonghan Wu et al. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021.

[WRB20] Larry Wasserman, Aaditya Ramdas, and Sivaraman Balakrishnan. Universal inference. *Proceedings of the National Academy of Sciences*, 117(29):16880–16890, 2020.

[WSY+21] Qingsong Wen et al. Time series data augmentation for deep learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4653–4660. ijcai.org, 2021.

[WWS+22] Jason Wei et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[WWYL21] Yifei Wang et al. Dissecting the diffusion process in linear graph convolutional networks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 5758–5769, 2021.

[WZH+24] Shirley Wu et al. Avatar: Optimizing LLM agents for tool-assisted knowledge retrieval. *CoRR*, abs/2406.11200, 2024.

[WZY+24] Shirley Wu et al. Stark: Benchmarking LLM retrieval on textual and relational knowledge bases. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[XHLJ19] Keyulu Xu et al. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[XPS+11] Liang Xiong et al. Hierarchical probabilistic models for group anomaly detection. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 789–797. JMLR.org, 2011.

[XWJ+24] Hongzuo Xu et al. Calibrated one-class classification for unsupervised time series anomaly detection. *IEEE Trans. Knowl. Data Eng.*, 36(11):5723–5736, 2024.

[XZS+20] Yilun Xu et al. A theory of usable information under computational constraints. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[YCA+18] Wenchao Yu et al. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2672–2681. ACM, 2018.

[YCHY08] Xifeng Yan et al. Mining significant graph patterns by leap search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 433–444. ACM, 2008.

[YCS16] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 40–48. JMLR.org, 2016.

[YGZL24] Shi-Qi Yan et al. Corrective retrieval augmented generation. *CoRR*, abs/2401.15884, 2024.

[YH02] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 721–724. IEEE Computer Society, 2002.

[YHC+18] Rex Ying et al. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 974–983. ACM, 2018.

[YHG13] Chao Yang, Robert Chandler Harkreader, and Guofei Gu. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Trans. Inf. Forensics*

Secur., 8(8):1280–1293, 2013.

[YHL15] Rose Yu, Xinran He, and Yan Liu. GLAD: group anomaly detection in social media analysis. *ACM Trans. Knowl. Discov. Data*, 10(2):18:1–18:22, 2015.

[YJK19] Jaemin Yoo, Hyunsik Jeon, and U Kang. Belief propagation network for hard inductive semi-supervised learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4178–4184. ijcai.org, 2019.

[YLSF23] Jaemin Yoo et al. Less is more: Slimg for accurate, robust, and interpretable graph mining. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 3128–3139. ACM, 2023.

[YRB+21] Michihiro Yasunaga et al. QA-GNN: reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 535–546. Association for Computational Linguistics, 2021.

[YSX+24] Xiao Yang et al. CRAG - comprehensive RAG benchmark. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[YZA23a] Jaemin Yoo, Lingxiao Zhao, and Leman Akoglu. End-to-end augmentation hyperparameter tuning for self-supervised anomaly detection. *CoRR*, abs/2306.12033, 2023.

[YZA23b] Jaemin Yoo, Tiancheng Zhao, and Leman Akoglu. Data augmentation is a hyperparameter: Cherry-picked self-supervision for unsupervised anomaly detection is creating the illusion of success. *Trans. Mach. Learn. Res.*, 2023, 2023.

[YZU+16] Chin-Chia Michael Yeh et al. Matrix profile I: all pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 1317–1322. IEEE Computer Society, 2016.

[YZW+22] Haoteng Yin et al. Algorithm and system co-design for efficient subgraph-based graph representation learning. *Proc. VLDB Endow.*, 15(11):2788–2796, 2022.

[YZY+23] Shunyu Yao et al. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[YZZA23] Jaemin Yoo et al. DSV: an alignment validation loss for self-supervised outlier model selection. In *Machine Learning and Knowledge Discovery in Databases: Research Track - European Conference, ECML PKDD 2023, Turin, Italy, September 18-22, 2023, Proceedings, Part I*, volume 14169 of *Lecture Notes in Computer Science*, pages 254–269. Springer, 2023.

[ZA22] Yue Zhao and Leman Akoglu. Towards unsupervised HPO for outlier detection. *CoRR*, abs/2208.11727, 2022.

[ZC18] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5171–5181, 2018.

[ZCH+20a] Jie Zhou et al. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[ZCH+20b] Jie Zhou et al. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[ZJAS22] Lingxiao Zhao et al. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[ZK21] Hao Zhu and Piotr Koniusz. Simple spectral graph convolution. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[ZLH+19] Bin Zhou et al. Beatgan: Anomalous rhythm detection using adversarially generated time series. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4433–4439. ijcai.org, 2019.

[ZLL+19] Li Zheng et al. Addgraph: Anomaly detection in dynamic graph using attention-based temporal GCN. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4419–4425. ijcai.org, 2019.

[ZNL19] Yue Zhao, Zain Nasrullah, and Zheng Li. Pyod: A python toolbox for scalable outlier detection. *J. Mach. Learn. Res.*, 20:96:1–96:7, 2019.

[ZRA21] Yue Zhao, Ryan A. Rossi, and Leman Akoglu. Automatic unsupervised outlier model selection. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 4489–4502, 2021.

[ZSA22] Lingxiao Zhao, Neil Shah, and Leman Akoglu. A practical, progressively-expressive GNN. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[ZWC+23] Yuxin Zhang et al. Adaptive memory networks with self-supervised learning for unsupervised anomaly detection. *IEEE Trans. Knowl. Data Eng.*, 35(12):12068–12080, 2023.

[ZWX+18] Zhen Zhang et al. Retgk: Graph kernels based on return probabilities of random walks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018,*

Montréal, Canada, pages 3968–3978, 2018.

[ZWZ+24] Kexin Zhang et al. Self-supervised learning for time series analysis: Taxonomy, progress, and prospects. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(10):6775–6794, 2024.

[ZYW+22] Ge Zhang et al. Dual-discriminative graph neural network for imbalanced graph-level anomaly detection. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[ZYZ+20] Jiong Zhu et al. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[ZZ21] Xinyi Zhou and Reza Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Comput. Surv.*, 53(5):109:1–109:40, 2021.

[ZZA22] Yue Zhao, Sean Zhang, and Leman Akoglu. Toward unsupervised outlier model selection. In *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, pages 773–782. IEEE, 2022.

[ZZN+22] Hongkuan Zhou et al. TGL: A general framework for temporal GNN training on billion-scale graphs. *CoRR*, abs/2203.14883, 2022.