KnowGraph: Knowledge-Enabled Anomaly Detection via Logical Reasoning on Graph Data

Andy Zhou
UIUC
Champaign, IL, USA
andyz3@illinois.edu

Xiaojun Xu
Bytedance Research
San Jose, CA, USA
xiaojun.xu@bytedance.com

Ramesh Raghunathan
eBay
Austin, TX, USA
raraghunathan@ebay.com

Alok Lal
eBay
San Jose, CA, USA
allal@ebay.com

Xinze Guan
eBay
San Jose, CA, USA
xiguan@ebay.com

Bin Yu
UC Berkeley
Berkeley, CA, USA
binyu@berkeley.edu

Bo Li
UIUC
Champaign, IL, USA
lbo@illinois.edu

Abstract

Graph-based anomaly detection is pivotal in diverse security applications, such as fraud detection in transaction networks and intrusion detection for network traffic. Standard approaches, including Graph Neural Networks (GNNs), often struggle to generalize across shifting data distributions. For instance, we observe that a real-world eBay transaction dataset revealed an over 50% decline in fraud detection accuracy when adding data from only a single new day to the graph due to data distribution shifts. This highlights a critical vulnerability in purely data-driven approaches. Meanwhile, real-world domain knowledge, such as "simultaneous transactions in two locations are suspicious," is more stable and a common existing component of real-world detection strategies. To explicitly integrate such knowledge into data-driven models such as GCNs, we propose KnowGraph, which integrates domain knowledge with data-driven learning for enhanced graph-based anomaly detection. KnowGraph comprises two principal components: (1) a statistical learning component that utilizes a main model for the overarching detection task, augmented by multiple specialized knowledge models that predict domain-specific semantic entities; (2) a reasoning component that employs probabilistic graphical models to execute logical inferences based on model outputs, encoding domain knowledge through weighted first-order logic formulas. In addition, KnowGraph has leveraged the Predictability-Computability-Stability (PCS) framework for veridical data science to estimate and mitigate prediction uncertainties. Empirically, KnowGraph has been rigorously evaluated on two significant real-world scenarios: collusion detection in the online marketplace eBay and intrusion detection within enterprise networks. Extensive experiments on these large-scale real-world datasets show that KnowGraph consistently outperforms state-of-the-art baselines in both transductive and inductive settings, achieving substantial gains in average precision when generalizing to completely unseen test graphs. Further ablation studies demonstrate the effectiveness of the proposed reasoning component in improving detection performance, especially under extreme class imbalance. These results highlight the potential of integrating domain knowledge into data-driven models for high-stakes, graph-based security applications.

CCS Concepts

• **Computing methodologies** → **Statistical relational learning**; **Statistical relational learning**; **Neural networks**; • **Security and privacy** → **Intrusion/anomaly detection and malware mitigation**.

Keywords

Graph neural networks, anomaly detection, intrusion detection, collusion detection, learning with reasoning

ACM Reference Format:
Andy Zhou, Xiaojun Xu, Ramesh Raghunathan, Alok Lal, Xinze Guan, Bin Yu, and Bo Li. 2024. KnowGraph: Knowledge-Enabled Anomaly Detection via Logical Reasoning on Graph Data. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690354

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690354

1 Introduction

Graphs are ubiquitous data structures with applications in various domains, such as social networks, biological networks, and communication networks. In recent years, graph neural networks (GNNs) [75, 93] and other graph representation learning techniques, such as node2vec [20], have emerged as powerful techniques for learning on graph-structured data. These methods have achieved remarkable success in applications that involve graphs such as recommendation [16], drug discovery [90], NLP [53], traffic forecasting [34],

and social network analysis [51]. In addition, graph data-based attacks have led to various real-world scenarios, such as fraudulent transactions in financial networks [8, 56, 72, 74], fake reviews or accounts in social networks [7, 26, 29], and network intrusions in computer networks [17, 39, 43]. Due to the large economic cost of these anomalies [1, 48], it is crucial to develop automated and robust methods to detect them.

Anomaly detection on graphs aims to identify these abnormal entities, connections, or substructures that deviate significantly from the expected patterns, summarized in Fig. 1. However, real-world graphs are often larger and more complex, holding properties such as extreme label imbalance and heterogeneity. For example, a single graph might have both locally homophilous and locally heterophilous regions [94], making generalization, especially to unseen graphs in inductive settings, challenging. While recent techniques leverage the structural nature of network or marketplace data through GNNs [5, 39, 44, 45, 72], several challenges remain when applied to large-scale real-world graphs, hindering their effectiveness in these critical domains. For instance, when a GCN model is deployed on a real-world eBay marketplace transaction dataset to detect fraudulent activities, its accuracy drops by 50% when only a single day's worth of new transactions is added to the graph, highlighting the challenge of maintaining performance on constantly evolving graphs.

This issue arises largely because current machine learning systems for anomaly detection are often purely data-driven, relying on the availability of large amounts of labeled data to learn patterns and anomalies [46]. However, this approach faces several challenges. One major challenge is information heterogeneity [71], as these techniques must leverage diverse sources of information, such as zip codes, item details, account data, authentication types, IP addresses, and event timestamps. Another challenge is label imbalance, as anomalous events or transactions are often extremely rare compared to benign ones, making it difficult to train ML systems that can reliably distinguish between them while maintaining a low false positive rate [47]. Collecting labels for every graph component is also costly and challenging, especially for large-scale graphs with millions of nodes. These factors make inductive generalization especially challenging, which is crucial as real-world graphs constantly evolve with millions of new events daily.

In contrast, traditional intrusion detection approaches often rely on human-defined rules and heuristics [4, 6, 11, 58]. While these rules may not be as flexible as data-driven methods, they can help address some of the challenges faced by current ML systems. For instance, domain experts can provide insights into the most informative features for fraud detection or lateral movement, reducing the reliance on costly labeled data. They can also define logical rules that character the relationships between different entities in the graph, enhancing the interpretability and robustness of the model. Furthermore, incorporating domain knowledge can guide the model to focus on the most relevant subgraphs or patterns, improving its ability to generalize to unseen graphs. However, standard GNN architectures, such as GCN [40], rely on global graph properties and cannot incorporate valuable human knowledge. Intuitively, integrating domain-specific knowledge into these architectures can significantly improve their generalization and interpretability.

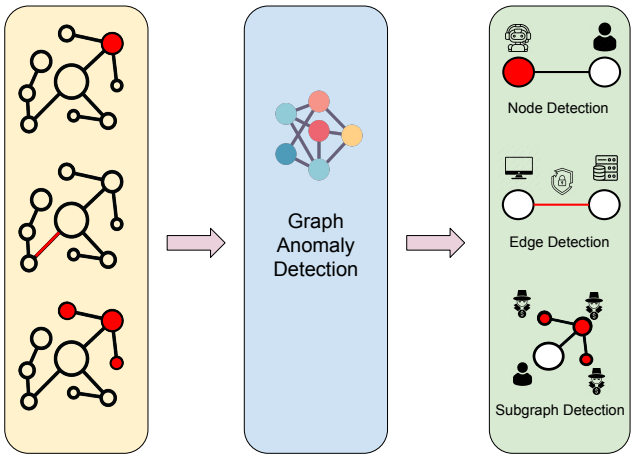

Figure 1: Examples of anomaly detection on graph-structure data with GNN models. Data-driven learning approaches have been successful on node-level, edge-level, and subgraph-level tasks but tend to consider different levels of the graph separately, focusing on a single level.

In this paper, we propose a framework, KnowGraph, that integrates expert knowledge into graph neural networks to bridge the gap between purely data-driven and rule-based approaches. Our framework consists of two key components: (1) a *learning* component that utilizes a *main model* for the overarching detection task, augmented by multiple specialized *knowledge models* that are trained with diverse objectives and can predict domain-specific semantic entities; (2) a *reasoning component* that utilizes probabilistic graphical models to perform logical inferences based on the outputs of each knowledge model. For statistical learning, we develop a set of complex models that can predict various aspects of graph data, such as node attributes, edge properties, and subgraph patterns. This expands the representational capacity of the main model, which is otherwise limited to a single overall facet. For reasoning, domain knowledge is encoded through first-order logic formulas with learned weights that organize each model's output, allowing the framework to learn how to leverage ground-truth information and constraints during inference. Due to the high-stakes nature of our experimental setting, we follow the Predictability, Computability, and Stability (PCS) framework [81–84] for our evaluation and introduce weight noise ensembling to all models, improving inductive generalization and ensuring prediction stability and reliability.

We demonstrate that our approach can be applied to both graph edges and subgraph detection, enabling it to capture both local and global patterns in the graph. We evaluate KnowGraph on two large-scale, real-world graph datasets: an eBay dataset containing over 40 million transactions from 40 days for collusion detection and the Los Alamos National Laboratory (LANL) network event dataset [37] consisting of 1.6 billion authentication events over 58 days for intrusion detection. These datasets present significant challenges due to their size, heterogeneity, label imbalance, and low frequency of malicious to benign events. Our experiments demonstrate that KnowGraph consistently outperforms state-of-the-art GNN models and baselines in both transductive and inductive settings. We summarize our contributions below

- We propose the first framework that integrates purely data-driven models with knowledge-enabled reasoning for enhanced anomaly detection on graph data. Our method, KnowGraph, combines multiple GNN models operating on different graph structures with a probabilistic logical reasoning component, which encodes human knowledge.
- We employ the Predictability, Computability, Stability (PCS) framework for veridical data science and introduce a weight noise ensembling approach to mitigate uncertainty and improve the inductive generalization ability of the framework.
- We evaluate KnowGraph on two large real-world datasets: an eBay dataset for collusion detection and the Los Alamos National Laboratory (LANL) network event dataset for intrusion detection. Our results demonstrate that KnowGraph consistently outperforms state-of-the-art baselines in both transductive and inductive settings with as few as three rules.

2 Related Work

Graph Neural Networks. Graph neural networks (GNNs) have become a cornerstone in machine learning for graph data, effectively encapsulating local graph structures and feature information by transforming and aggregating representations from neighbors. Common architectures include graph convolutional networks (GCN) [40], which uses convolutions directly on graph structures to capture dependencies, GraphSAGE [23], which learns inductive representations through sampling and aggregating neighborhood information, and GAT [66], which introduces an attention mechanism to aggregate neighborhood information. To represent long-range dependencies in disassortative graphs, Geom-GCN [54] introduces a geometric aggregation scheme that enhances the convolution operation by leveraging the continuous space underlying the graph. GraphSAINT [86] proposes a graph sampling-based training method that allows for efficient and scalable learning on large graphs by iteratively sampling small subgraphs and performing GNN computations on them. DR-GCN [61] employs a class-conditioned adversarial network to mitigate bias in node classification tasks with imbalanced class distributions. More recently, Graphormer [78] proposes a graph transformer model employing attention mechanisms to leverage structural information well.

Generalization of GNNs. While some standard GNN architectures [23, 86] have been proposed for inductive settings, generalization remains challenging for graphs with class imbalance or diverse structures. Standard techniques such as resampling [9, 25] or reweighting [32, 42] are often sensitive to overfitting on minority classes and less effective on shifts to new graphs. Instead, many techniques aim to learn more robust and generalizable graph representations through data augmentation [24, 91] or large-scale pretraining [30, 79, 80]. In addition, GNN capacity can be enhanced by model combination techniques such as model ensemble [63, 73], where the predictions of multiple models are combined to improve performance. Mixture-of-experts (MoE) [15, 33, 60] is a similar technique where the problem space is divided by routing inputs to specialized experts. Recent work adopts MoE for GNNs to address the class imbalance issue [31, 68, 85]. We also aim to address class imbalance and inductive generalization but leverage many task-specific GNNs organized through more deliberate domain knowledge.

To evaluate the generalization and uncertainty of GNNs in a principled manner, we draw upon the Predictability, Computability, and Stability (PCS) framework [83, 84]. PCS emphasizes three key principles: predictability, which serves as a reality check for models; computability, which considers the feasibility and scalability of methods; and stability, which assesses the consistency of results under perturbations to data and models. In this work, we focus on the stability aspect and propose to incorporate weight noise ensembling [3] into KnowGraph to mitigate uncertainty and improve generalization to out-of-distribution graphs.

Graph anomaly detection. Graph anomaly detection has become an increasingly important research area, with applications spanning various domains such as network security, financial fraud detection, and social network analysis. The goal of graph anomaly detection is to identify abnormal nodes, edges, or subgraphs within a graph-structured dataset that deviates from the expected patterns or behaviors. In this paper, we focus on two representative settings: lateral movement detection and collusion detection.

In network security, graph anomaly detection is crucial in detecting and mitigating lateral movement, a key stage in the MITRE ATT&CK framework [50]. Lateral movement refers to the propagation of malware through a network to compromise new systems in search of a target, often involving pivoting through multiple systems and accounts using legitimate credentials or malware. Research on mitigating lateral movement in computer networks generally follows three main approaches: enhancing security policies, detecting malicious lateral movement, and developing forensic methods for post-attack analysis and remediation [14, 17, 22, 23, 27, 39, 43, 55, 62]. While proactive security measures can help reduce the attack surface, they cannot eliminate all potential paths an attacker might exploit. Many lateral movement detection techniques represent internal network logins as a machine-to-machine graph and employ rule-based or machine learning algorithms to identify suspicious patterns [4, 27, 37]. However, these methods often struggle with scalability, generate excessive false alarms, or fail to detect attacks that do not match predefined signatures. Recent approaches improve detection by leveraging the structural nature of network data [5, 39], formulating the problem as a temporal graph link prediction task. These methods aim to identify edges with low likelihood scores correlated with anomalous connections indicative of lateral movement. However, challenges remain in generalizing these models to new networks in inductive settings.

In the domain of financial fraud detection, graph anomaly detection has also gained significant attention due to the economic cost and prevalence of fraud in various settings, such as social networks [7], online payment systems [92], and online marketplace platforms [8, 44, 56]. Early explorations focused on rule-based methods [11] and association rules [6, 58], but these approaches often fail to adapt to evolving fraudulent behaviors over time. With the availability of large-scale transaction data, data-driven and learning-based methods have gained popularity, including SVM-based ensemble strategies [67], graph-mining-based approaches [64, 65], convolutional neural networks (CNNs) [18], and recurrent neural networks (RNNs) for sequence-based fraud detection [35, 70, 89].

More recently, GNNs have been applied to various fraud detection problems, leveraging the structural nature of transaction data [41, 44, 45, 69, 72]. Some works have also combined homogeneous and heterogeneous graphs to help information propagation through various types of nodes and edges [41, 56]. Despite these advancements, our proposed setting of detecting collusive fraud, where multiple adversaries conspire for illegal financial gain, remains particularly challenging due to the limited normal transaction history of colluders, severe label imbalance, and the diverse range of collusion mechanisms employed by fraudsters.

Knowledge-based logical reasoning. Machine learning models are often purely data-driven and cannot directly use human knowledge to improve performance. Integrating human knowledge, such as relationships between classes, has been shown to improve ImageNet [13] classification accuracy [12] or adversarial robustness [21] for deep neural networks. In addition, Bayesian logic programs [38], relational Markov networks [19], Bayesian networks [59], and Markov logic networks [57] have also been adopted for logical reasoning. Recent work has combined these methods into learning-reasoning frameworks [21, 77, 87] that organize the predictions of various classifiers with knowledge rules whose weights are learned with a logical inference model. However, these works only consider simple models and tasks, such as small classifiers and datasets. The scalability of learning-reasoning frameworks, especially on large-scale graph data, has not been explored.

3 Preliminaries

Graph Neural Networks. Given a graph $G = (V, E, Y)$, where V is the set of nodes, E is the set of edges, and Y is the set of labels associated with the graph elements (nodes, edges, or the entire graph), the overall task of graph neural networks (GNNs) is to learn a mapping $f : G \rightarrow Y$ that predicts the labels of the graph elements based on their local neighborhood structure. To achieve this, GNNs learn representations of the graph elements by repeatedly performing neighborhood aggregation or message passing across multiple layers. The learned representations can then be used for various graph-related tasks, such as node classification ($f : V \rightarrow Y$), edge prediction ($f : E \rightarrow Y$), or graph classification ($f : G \rightarrow Y$).

Let $\mathbf{x}_j \in \mathbb{R}^{F_0}$ be the input feature vector of node $j \in V$, $h_j^{(\ell)} \in \mathbb{R}^H$ be the representations or embeddings of node j learned by layer ℓ ($\ell \geq 1$), and $e_{(k,j)} \in \mathbb{R}^{F_2}$ be optional features of the edge (k, j) from nodes k to j. The message passing procedure that produces the embeddings of node j via the ℓ-th GNN layer can be described as follows:

$$h_j^{(\ell)} = \gamma^{(\ell)}\left(\bigoplus_{k \in \mathcal{N}(j)} \psi^{(\ell)}\left(h_j^{(\ell-1)}, h_k^{(\ell-1)}, e_{(k,j)}\right)\right), \tag{1}$$

Here, $t_i^{(0)}$ is set to x_j, $\mathcal{N}(j)$ is the neighborhood of node j, and $\psi(\cdot)$ is a function that extracts a message for neighborhood aggregation, which summarizes the information of the nodes j and k, as well as the optional edge features $e_{(k,j)}$ if available. $\bigoplus(\cdot)$ denotes a permutation-invariant function (e.g., mean or max) to aggregate incoming messages, and $\gamma(\cdot)$ is a function that produces updated embeddings of node j by combining node j's embeddings with aggregated messages. With multi-layer GNNs, the embeddings of a node learned via local message passing capture information from its k-hop neighborhood.

The final output of a GNN model, denoted as $t_i(\cdot)$, is obtained from the last layer's embeddings $t_i^{(L)}$, where L is the total number of layers in the GNN. The model's prediction confidence for node j is represented by z_j, which is derived from t_i using a separate output function (e.g., a softmax layer).

Markov Logic Networks. MLNs combine first-order logic and probabilistic graphical models to enable probabilistic reasoning over knowledge bases. An MLN is essentially a set of weighted first-order logic formulas, where each formula is associated with a weight that reflects its importance or confidence.

In the context of MLNs, the relationships between entities are represented using predicates. A predicate $p(\cdot)$ is a logical function that maps tuples of variables to binary truth values. Given a set of variables $\mathcal{V} = \{v_1, \ldots, v_N\}$, where each v_i represents a logical constant (e.g., an entity or an attribute), a predicate is defined as:

$$p(\cdot) : \mathcal{V} \times \ldots \times \mathcal{V} \rightarrow \{0, 1\}. \tag{2}$$

First-order logic formulas in MLNs combine predicates using logical connectives (e.g., $\wedge$, $\vee$, $\neg$). A formula $f(\cdot)$ is defined over a set of predicates and maps tuples of variables to binary truth values:

$$f(\cdot) : \mathcal{V} \times \ldots \times \mathcal{V} \rightarrow \{0, 1\}. \tag{3}$$

Each formula f_i in an MLN is associated with a weight $w_i \in \mathbb{R}$, which represents the confidence or importance of that formula.

Given a set of observed predicates (evidence) $\mathcal{E}$, an MLN defines a joint probability distribution over all possible worlds (i.e., possible assignments to all predicates):

$$P_w(\mathcal{X} = x|\mathcal{E}) = \frac{1}{Z(w)} \exp\left(\sum_{i=1}^{M} w_i n_i(x)\right), \tag{4}$$

where $\mathcal{X}$ is the set of all predicates, x is a possible world (one possible assignment for the predicates), M is the number of formulas in the MLN, $n_i(x)$ represents true/false value of formula f_i in x, and $Z(w)$ is the partition function.

Inference in MLNs involves computing the probabilities of specific predicates or formulas given the observed evidence. Various techniques, such as Markov chain Monte Carlo (MCMC) methods or lifted inference algorithms, can be used.

In the context of integrating knowledge into graph neural networks, MLNs can represent domain knowledge and perform probabilistic reasoning over the predictions of multiple GNN models. The models' outputs can be treated as observed predicates, and the MLN can be used to infer the most likely overall predictions based on the defined logical formulas and their associated weights.

4 KnowGraph

4.1 Overview

To effectively integrate domain knowledge into data-driven graph neural networks (GNNs), we propose KnowGraph. In particular, KnowGraph consists of two components: a *learning* component

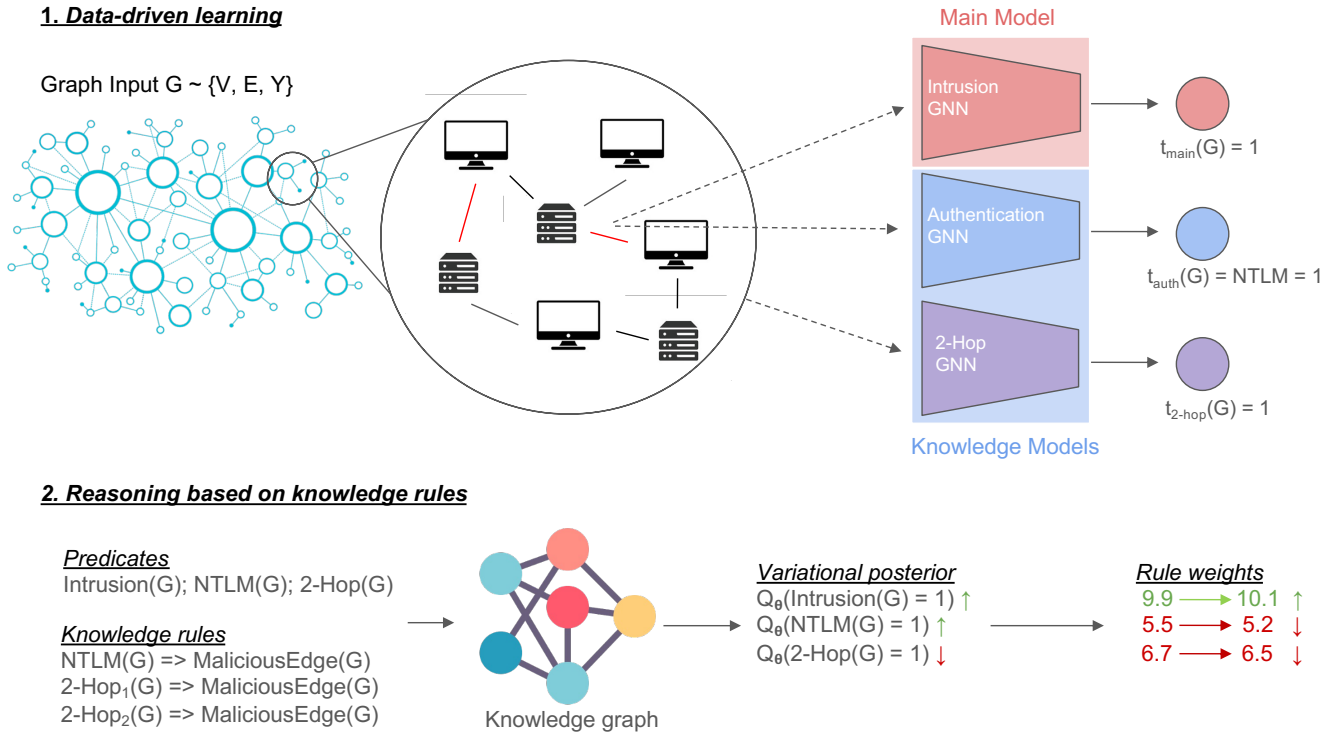

Figure 2: An overview of the learning and reasoning components of KnowGraph. KnowGraph consists of a learning component composed of a main GNN model trained on the overall task and multiple knowledge GNN models trained on separate objectives, such as predicting relevant sub-attributes. The reasoning component performs logical reasoning based on the outputs of each model, which is organized based on domain knowledge rules. These rules are assigned weights, modeled by a learned scalable reasoning model parameterized by θ, which explicitly ensures that the final predictions comply with the domain knowledge rules, improving reliability.

and a *reasoning* component. Specifically, the *learning* component comprises a main model and several knowledge models. The main model focuses on the main function (e.g., malicious edge detection in intrusion detection), delivering multi-class predictions based on graph inputs. The knowledge models take the same graph as input to predict specific semantic entities (e.g., authentication type of an edge). The internal relationships among these models are represented by expert-designed knowledge rules (e.g., a malicious edge indicates a specific authentication type), which are embedded in a reasoning component for logical inference. For example, in an edge-level intrusion detection task, the main model is trained to predict whether an edge is malicious, and a knowledge model is trained to predict the authentication type of the edge. The outputs of these models are then sent to a reasoning component to check whether they satisfy the domain knowledge rules such as "malicious edges indicate an NTLM [49] authentication type". During inference, if the knowledge is violated, the prediction of the main model will be automatically corrected, leading to a more resilient final prediction. In KnowGraph, every model in the learning component is a trained GNN model, distinguishing KnowGraph from some previous frameworks [21, 77, 87] that rely on simpler deep neural network (DNN) classifiers.

Concretely, the knowledge rules are formalized as first-order logic rules, such as "$\texttt{IsNTLM(x)} \implies \texttt{IsMalicious(x)}$" within the *reasoning* component, which learns a weight for each rule from its direct usefulness. The *reasoning* component itself is implemented using probabilistic graphical models, such as a Markov Logic Network (MLN) [57]. Due to the computationally intensive nature of MLN inference from the exponential complexity of constructing the ground Markov network, we use a scalable variational inference method [87] to learn a GNN to embed the knowledge rules. We utilize the Expectation-Maximization (EM) algorithm that iteratively enhances the GCN's accuracy in learning the GCN model weights (E step) and refines the weighting of knowledge rules (M step) within the latent MLN. During inference, the learned reasoning GCN component ensures that the prediction outputs of the main and knowledge models satisfy the defined domain knowledge rules and achieve resilient final predictions.

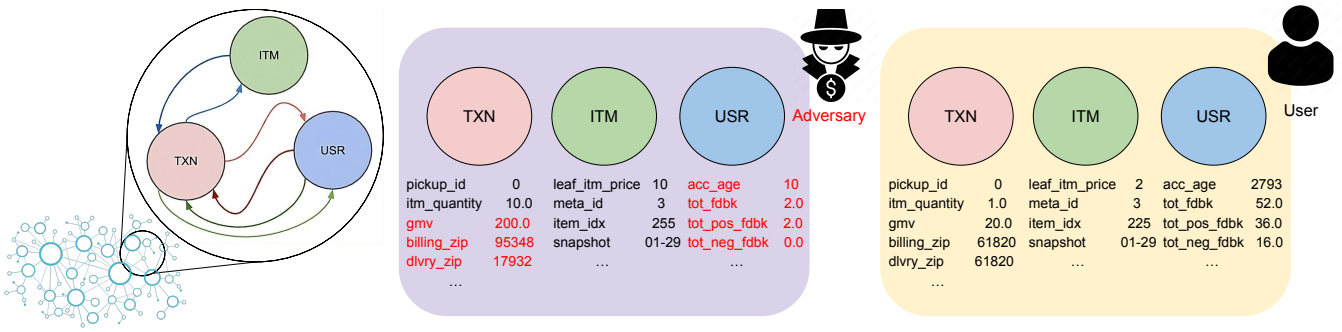

Figure 3: An illustration of the graph from real eBay marketplace data containing the core entities of a transaction, which are represented as nodes in a knowledge graph consisting of nodes of transactions (TXN), users (USR), and items (ITM). Examples of collusive and benign transactions are shown. Example node attributes are listed below the entity, such as gross merchandise value (gmv), account age (acc_age), and total feedback (tot_fdbk). Domain knowledge suggests that collusive transactions typically involve discrepancies between billing and delivery zip codes and are associated with users who have minimal feedback and newer accounts. This understanding has led to the formulation of rules such as "$[\texttt{feedback amt} < a] \wedge [(\texttt{seller\_age} < b \vee \texttt{buyer\_age} < c)] \Rightarrow \texttt{collusion}$", which help refine the main model's predictions based on these indicators.

4.2 Learning with knowledge-enabled reasoning on graph data

In the *learning* component of KnowGraph, we train a set of GNN models to predict the main task and knowledge models for assistant tasks. The knowledge models are conceptualized as predicates within this framework, outputting binary predictions for each knowledge model.

Formally, we denote the output of the i-*th* model by $t_i(\cdot)$ as t_i, with z_i representing the confidence level of its prediction. For a given input graph $G = (V, E, Y)$ with nodes V, edges E, and labels Y, the GNN model's prediction is denoted as $t_i(G)$. Upon receiving input G and the assorted model predictions $t_i(G)$, these predictions are interlinked through their logical interrelations and a Markov Logic Network (MLN) [57], enabling the *reasoning* capability in KnowGraph.

Specifically, KnowGraph involves a primary model alongside multiple knowledge models $t_i(G)$, serving as predicates within the MLN framework. Logical connections between these predicates are established to formulate different logical expressions. Assuming L models in total, an MLN defines a joint probability distribution over the pre-defined logical expressions (i.e., knowledge rules), which can be expressed as follows:

$$P_w(t_1, ..., t_L) = \frac{1}{Z(w)} \exp\left(\sum_{f \in \mathcal{F}} w_f f(t_1, ..., t_L)\right), \tag{5}$$

with $Z(w)$ symbolizing the partition function, summing across all predicate assignments.

KnowGraph's reasoning component manages logic formulas articulated as first-order logic rules. Following [87], we consider three types of logic rules:

- Attribute rule ($t_i \implies t_j \vee t_j \vee ...$): This rule leverages specific attributes associated with prediction classes to formulate knowledge-based rules.
- Hierarchy rule ($t_i \implies t_j$): Reflecting the hierarchical nature among classes, this rule aids in constructing logical expressions like $f(t_i, t_j) = \neg t_i \vee t_j$.
- Exclusion rule ($t_i \oplus t_j$): This rule addresses the inherent exclusivity among some class predictions, ensuring that an entity cannot simultaneously belong to mutually exclusive classes.

After designing the models and rules, the final step of KnowGraph is to learn and assign a weight for each rule to reflect the impact of their prediction confidence z_i for each model $t_i(\cdot)$. To achieve this, we utilize the logarithm of the odds ratio, $\log[z_i/(1 - z_i)]$, as the weight for model t_i.

4.3 Scalable reasoning with a GCN

To reduce the computational complexity of training the MLN, we employ variational inference [87] to optimize the variational evidence lower bound (ELBO) of the data log-likelihood. This approach is motivated by the intractability of directly optimizing the joint distribution $P_w(O, \mathcal{U})$, which requires computing the partition function $Z(w)$ and integrating over all observed predicates O and unobserved predicates $\mathcal{U}$. The ELBO is formulated as follows:

$$\begin{aligned} \log P_w(O) \geq \mathcal{L}_{\text{ELBO}}(Q_\theta, P_w) = \mathbb{E}_{Q_\theta(\mathcal{U}|O)}\left[\log P_w(O, \mathcal{U})\right] \\ - \mathbb{E}_{Q_\theta(\mathcal{U}|O)}\left[\log Q_\theta(\mathcal{U}|O)\right], \end{aligned} \tag{6}$$

where $Q_\theta(\mathcal{U}|O)$ is the variational posterior distribution. The representation of model outputs and knowledge rules as a graph motivates the use of Graph Convolutional Networks (GCNs) for encoding $Q_\theta(\cdot)$.

We adopt a variational EM algorithm to refine the ELBO and learn the MLN weights w. In the E-step, the GCN parameters Q_θ are updated to minimize the KL divergence between $Q_\theta(\mathcal{T})$ and $P_w(\mathcal{T})$, where $\mathcal{T} = t_1, t_2, ..., t_L$ are the model outputs. The optimization objective is enhanced with a supervised negative log-likelihood term

$\mathcal{L}_{\text{supervised}}$ to leverage available label information during training:

$$\mathcal{L}_{\text{ELBO}}(Q_\theta, P_w) := \mathbb{E}_{Q_\theta(\mathcal{T})}[\log P_w(\mathcal{T})] - \mathbb{E}_{Q_\theta(\mathcal{T})}[\log Q_\theta(\mathcal{T})]. \tag{7}$$

where η is a hyperparameter balancing the importance of the ELBO and the supervised term. This approach ensures that the class embedding vectors $\vec{\mu}$ are refined during the optimization of the GCN, enhancing the model's expressiveness.

Conversely, the M-step fixes Q_θ while updating the weights w. Since directly integrating over all variables is intractable, we follow [87] and optimize the pseudo-likelihood [2] instead, which is formulated as:

$$P_w^*(t_1, ..., t_L) := \prod_{i=1}^{L} P_w\left(t_i | MB\left(t_i\right)\right), \tag{8}$$

where $f\left(t_i = 0\right)$ and $f\left(t_i = 1\right)$ reflect the formula f's ground truth values under the hypothetical scenarios of t_i being 0 or 1, respectively, with other variables held constant.

This strategy addresses the intractability of directly optimizing the complex log-likelihood by maximizing the expectation of the pseudo-log-likelihood, which is estimated through multiple samplings from the variational distribution Q_θ.

For a more detailed description of the variational inference procedure, including the derivation of the equations and the specific formulations of the ELBO terms, please refer to [52, 87].

4.4 Uncertainty mitigation with PCS

To further improve inductive generalization, we draw upon the Predictability, Computability, and Stability (PCS) framework [83, 84], which is a comprehensive approach to ensure the reliability, reproducibility, and transparency of data-driven results when employing complex modeling techniques. The PCS framework emphasizes three key principles: predictability, which serves as a reality check for models; computability, which considers the feasibility and scalability of methods; and stability, which assesses the consistency of results under perturbations to data and models. When applying KnowGraph to critical real-world tasks, uncertainty quantification helps to assess and mitigate the uncertainty associated with our results. Adding uncertainty information to KnowGraph can be valuable for decision-making, as it allows us to identify cases where the model is less confident and may require additional investigation or human intervention.

In particular, the stability principle in PCS emphasizes the importance of assessing the stability of data results with respect to data and model perturbations. We propose incorporating a weight noise ensembling technique into the stability principle of the PCS framework. Weight noise ensembling introduces random perturbations to the weights of the neural network during training and inference, allowing us to capture the uncertainty in the model's parameters by considering a distribution over the weights rather than a single point estimate. This approach acts as a regularization technique, reducing overfitting in our setting with extreme label imbalance and promoting the learning of robust features that can generalize to new graphs.

To integrate weight noise ensembling into our pipeline within the PCS framework, we modify the statistical learning and reasoning components of KnowGraph. Within the learning component, we train multiple instances of the GNN model, each with different random perturbations added to the weights. These perturbations are sampled from a predetermined probability distribution (e.g., Gaussian distribution with mean 0 and a specified variance). In the reasoning component, we perform inference using each perturbed model and collect the predictions from each instance. We then compute summary statistics (e.g., mean and variance) of the ensemble predictions to mitigate the uncertainty associated with each collusion detection result, allowing us to make more robust updates to the learned rule weights and GCN posterior.

Discussion. Conceptually, KnowGraph has several intuitive advantages as a framework for graph-based anomaly detection: (1) *Flexibility*: KnowGraph improves performance with even a small number of well-designed rules. The usefulness of each rule is encoded in its learned weight, allowing the framework to default to the original base model prediction if the rule is not helpful. (2) *Scalability*: The learning component trains knowledge models in parallel, scaling linearly with the number of models. The reasoning component's complexity depends on the number of rules, where each rule adds only a few nodes and edges to the reasoning component, allowing for easy expansion of knowledge models and rules without significant computational overhead. (3) *Adaptability*: By combining data-driven learning with knowledge-based reasoning, KnowGraph demonstrates strong generalization capabilities. This hybrid approach allows the framework to adapt to new patterns and evolving scenarios, making it particularly suited for dynamic environments where attack strategies may change over time. Both new knowledge models and rules can also be added, only requiring the reasoning component to be retrained.

5 Experiments

5.1 Intrusion detection on LANL

In this section, we describe our results on intrusion detection on the public Los Alamos National Labs (LANL) [36] dataset.

Dataset. We conduct experiments on a open-sourced intrusion detection dataset[1] from Los Alamos National Labs (LANL) [36]. The LANL dataset contains a 58-day log within their internal computer network, among which the malicious authentication events are identified. The dataset comprises 12,425 users, 17,684 computers, and 1.6B authentication events. We model the dataset as a graph so that a node represents a server in the network and an edge represents an authentication event (e.g. log-in / log-out) from one server to another. We will remove the repeated edges in the graph so that the overall number of edges is 45M. Our goal is to detect whether an edge is malicious or not.

Labels. Each edge in the graph has a ground-truth label by the dataset, indicating whether the authentication event is malicious or not. Only 518 malicious edges are among the 45M edges, making the dataset highly unbalanced.

Settings and baselines. We mainly follow the data pre-processing procedure as in the Euler [39] work. We use the training and validation set that consists of all the information before the first anomalous edge appears, with 5% of them being the validation set and the

[1]Detailed dataset information can be found at https://csr.lanl.gov/data/cyber1/.

Table 1: An overview of the learning models and rules designed for lateral movement detection on LANL [36], which we formulate as a link prediction problem to identify malicious edges. We design three models and three knowledge rules to ensure resilient inference of the base model. The rules are designed from domain knowledge and verified in the dataset.

Model	Description	Rule
Main	Main model to predict if an edge is malicious	Used in all rules as the overall base model
Auth	Binary edge classifier to determine if the authentication type of an edge is NTLM	Authentication rule: "[main = Mal] ⇒ [auth = NTLM]"
2-hop	Model to predict if a edge is malicious using a 2-hop enclosing graph	Subgraph rules: "[EncG = Mal] ⇒ [main = Mal]", "[EncG = Benign] ⇒ [main = Benign]"

Table 2: Detection Performance of the baselines and our KnowGraph framework. We observe better performance for KnowGraph in both transductive and inductive settings. Transductive settings involve learning and testing on the same graph, leveraging its entire structure, whereas inductive settings require the model to generalize to new graphs or unseen parts of a graph based on learned patterns.

Transductive					
Method	AUC	AP	TP@0.5FP	TP@1FP	TP@2FP
Euler [39]	0.9946	0.0433	0.7777	0.9444	**1.0**
EncG [88]	0.9995	0.3249	**1.0**	**1.0**	**1.0**
KnowGraph	**0.9999**	**0.8886**	**1.0**	**1.0**	**1.0**
Inductive					
Method	AUC	AP	TP@0.5FP	TP@1FP	TP@2FP
Euler [39]	0.8973	0.0193	0.0	0.0	0.2534
EncG [88]	0.8269	0.0338	0.1648	0.2755	0.3950
KnowGraph	**0.9112**	**0.0852**	**0.3554**	**0.4643**	**0.5910**

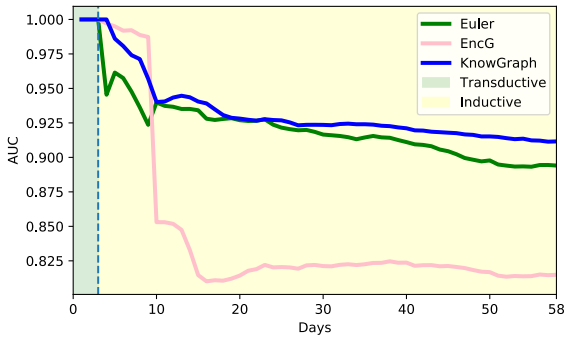

Figure 4: Detection AUC of baselines and KnowGraph given different time shifts on the LANL dataset. In the challenging inductive setting (yellow), the baseline performance drops significantly while KnowGraph still maintains high detection performance.

rest being the training set. We split the dataset into multiple graphs so that each graph consists of the edges within a time window of 1,800 seconds. After training the model, we will evaluate it on the test set, which consists of the edges after the first anomalous edge appears. To compare the detection performance, we will evaluate the AUC, Average Precision (AP), and the True Positive rate at a certain False Positive rate (e.g., TP@0.5FP denotes the true positive rate when the false positive rate is 0.5%).

We adopt two baselines. The first baseline is the Euler [39] work, where the authors use graph neural networks (with or without a recurrent neural network header) to detect the malicious edges. The inputs to the model include the node and edges within a time window. The GNN processes the information and returns a value for each edge, indicating the probability that it is malicious. The model will be trained with negative sampling. In each training step, the model will be trained to give a low malicious probability for the existing edges and a high malicious probability for a randomly sampled set of non-connected node pairs.

The second baseline uses link prediction with an Enclosing Graph (EncG) [88]. EncG extracts the K-hop enclosing subgraph for each edge and trains a subgraph classification model to determine whether the corresponding edge is malicious. We use $K = 2$ considering the tradeoff between efficiency and effectiveness.

Knowledge models. We have three models for this task. First, we have the `main` model to determine whether the edge is malicious, with the same model architecture as in the Euler work. Second, we have the `auth` model, which is also a binary edge classifier to determine whether the authentication type of the model is NTLM or not. Finally, we have a `EncG` model that follows the same methodology in the EncG [88] work to judge the edge maliciousness based on the enclosing graphs.

Knowledge rules. We designed three rules for the task, which are as follows: (1) *Authentication rule*, which states that malicious authentications are likely to be in type NTLM, "[main = Mal] ⇒ [auth = NTLM]". This rule incorporates the human knowledge that NTLM authentication is usually a less secured authentication protocol (compared with, for example, Kerberos), and most malicious authentications are using NTLM authentication. (2) *Subgraph rule 1* which states that the main model should be malicious if 2hop model predicts a malicious, "[EncG = Mal] ⇒ [main = Mal]". (3) *Subgraph rule 2* which states that the main model should be benign if 2hop model predicts a benign, "[EncG = Benign] ⇒ [main = Benign]".

Table 3: Detection performance (AUC) of the models and KnowGraph framework with partial knowledge. We observe that KnowGraph with the final reasoning framework indeed achieves the best detection performance.

Model	Transductive	Inductive
`main`	0.9946	0.8973
`auth`	0.9440	0.7950
`EncG`	0.9995	0.8269
`main+auth`	0.9947	0.9003
`main+EncG`	0.9997	0.9006
`main+auth+EncG`	**0.9999**	**0.9112**

Note that rules (2) and (3) combined indicate that the prediction outputs of the `main` model and the `2hop` model should be consistent.

Main Results. Based on our intensive evaluations, we show the model and detection performance of the final reasoning pipeline in Table 2. We observe that in the transductive setting, all models perform well with close performance given similar training and testing data distributions. In particular, the detection performance of the baselines can achieve over 0.99 AUC in the inductive setting. By integrating the model information, we can improve the performance and achieve a close-to-perfect detection performance. However, the model performance drops to below 0.9 under the inductive learning setting with out-of-distribution data. By integrating knowledge rules and reasoning, KnowGraph can better aggregate the information from the models in this OOD scenario and significantly improve detection performance. This showcases the robustness of the reasoning framework against OOD cases and matches our intuition of including human knowledge in the pipeline.

In addition, we also show the detection AUC and prediction logits change on the time domain for both Euler and our method in Figure 4 and Figure 5, respectively. We can observe that, as the timestamp goes to the later days, the detection AUC will gradually drop due to the OOD issues. For both Euler and EncG, many benign OOD cases are viewed as malicious by the model, which is not seen in the transductive setting. Moreover, many malicious accesses receive a low malicious score in EncG. By comparison, the distribution of benign and malicious accesses is generally similar in both transductive and inductive settings. We attribute it to the stability of the reasoning framework with human knowledge.

Ablation Studies. We show the ablation study of model performance in Table 3. We can observe that every model has a relatively good performance (with transductive AUC larger than 0.9 and inductive AUC larger than 0.8). After combining the models with our reasoning framework, we can further improve the performance in both transductive and inductive settings. Note that the reasoning framework is ubiquitously better than each of its components, showing its superiority over simple methods, which, for example, average the results. In addition, we observe a trend that combining better-performing models can yield better reasoning results, which shows the importance of training better models and designing the reasoning framework.

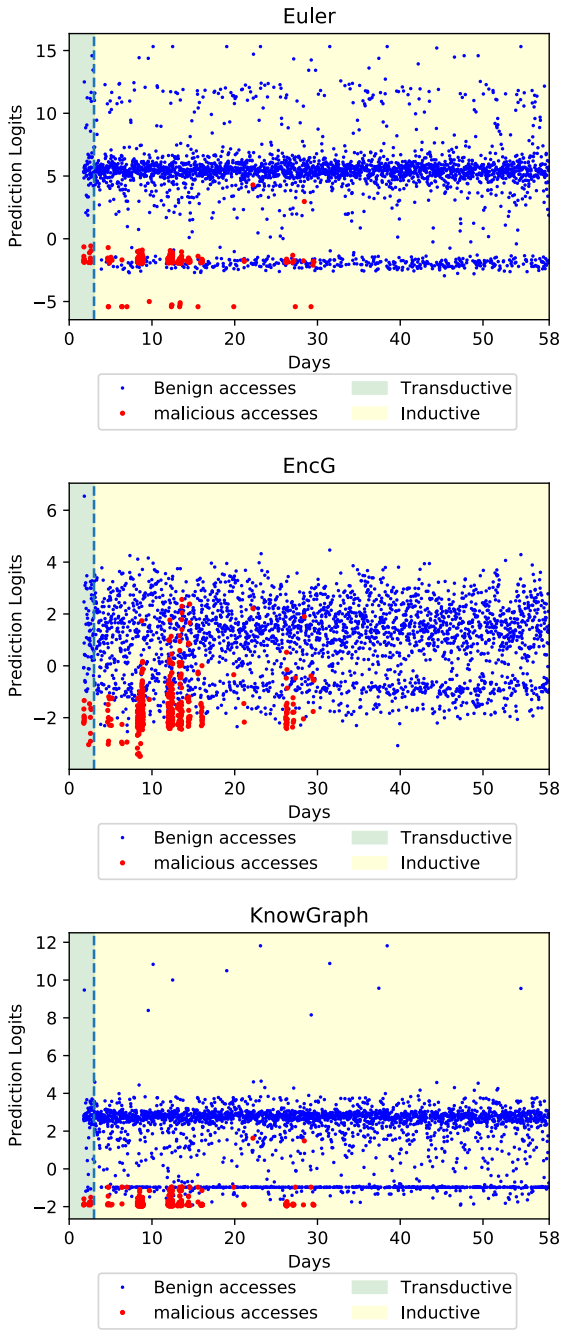

Figure 5: Model prediction logits on the different time shifts for (top) Euler, (middle) EncG, and (bottom) KnowGraph. Low prediction logits indicate that the access is considered malicious. We can observe that all approaches perform well for the transductive setting; for the inductive setting, it is easier for KnowGraph to separate the malicious accesses by setting a threshold at around -1.5. However, it is difficult to classify benign and malicious accesses based on the prediction logits of Euler and EncG.

Table 4: An overview of the models and knowledge rules designed for collusion detection on the real-world eBay dataset, a heterogeneous graph of marketplace transactions. We design a total of eight data-driven learning models and six knowledge rules to enhance the inference of the main task model. These rules are designed by experts with domain knowledge.

Model	Description	Knowledge Rule
Main	Main model to predict if a user, transaction, or item node is an instance of collusion	Used in all rules as the overall base model
Secondary	Same objective as the main model, but trained on a larger graph snapshot with balanced sampling	Used in the Ensemble Rules: "[main = collusion] ⇒ [secondary = collusion]" and "[secondary = collusion] ⇒ [main = collusion]."
Account Takeover	Model to predict collusion subclass where one of the colluding parties fraudulently takes control of another account	Used in Hierarchy Rule: "[ATO = collusion] ⇒ [main = collusion]."
Sign-In	Model to predict collusion subclass accompanied by some problem indicator related to sign-in	Used in Hierarchy Rule: "[sign-in = collusion] ⇒ [main = collusion]."
One Day Reg	Model to predict collusion subclass where buyers register and purchase an item on the same day	Used in Hierarchy Rule: "[one-day-reg = collusion] ⇒ [main = collusion]."
US eBay Decline	Model to predict collusion subclass where the transaction is declined by eBay but overridden by the colluding party	Used in Hierarchy Rule, "[ebay-decline = collusion] ⇒ [main = collusion]."
Account Age	Domain knowledge using labeled attributes on the age and amount of user feedback of an account	Used in Business Rule, "[feedback amt $< a$] ∧ [(seller_age $< b \vee$ buyer_age $< c$)] ⇒ collusion"
Business	Domain knowledge using labeled attributes of the zip code, price, and gross value of a transaction	Used in Business Rule, "[(gmv − price $> d$] ∧ [billing_zip ≠ delivery_zip] ⇒ collusion"

5.2 Collusion detection on real-world eBay marketplace dataset

In this section, we describe the real-world eBay marketplace dataset and our collusion detection results and analysis in detail.

Dataset. We use a large-scale proprietary dataset on real-world marketplace transactions from the popular online shopping website eBay, which has more than 135 million users [76]. The dataset contains transactions from 40 days total, each with around 4 million transactions, collected from January to February of 2022. Each transaction consists of three entities: a seller, a buyer, and an item. These transactions and entities are organized into a knowledge graph, where each entity is a node with bidirectional edge relationships with other entities in the same transaction. The corresponding task aims to train models that predict if a transaction indicates buyer-seller collusion, a type of fraud in which buyers and sellers conspire for illegal financial gain. This fraud can involve manipulating prices or exchanging fake feedback to deceive the marketplace, leading to significant financial losses. Due to the scale of the marketplace, this has a financial cost of millions of dollars a year based on proprietary estimates, making automated detection crucial.

Labels. Each transaction has ground-truth collusion labels from real collusion cases and additional data based on various buyer, seller, and item features. These include relevant knowledge features such as transaction zip code, item price, and shipping cost, comprising 26 features across the three entities. This is summarized in Fig. 2, which contains example differences between benign and anomalous transactions. The ground-truth collusion labels include a single overall binary classification label and multiple fine-grained class or collusion subclass labels. Subclasses are based on specific instances of collusion, such as collusion from a particular group or those exhibiting specific collusive modus operandi. The number of positive labels is extremely sparse, with only around 1330 new collusion cases daily. This makes the exact ratio of positive to negative labels 1 : 3269.371, a case of extreme label imbalance.

Settings and baselines. To manage label sparsity in our data, we segment the graph into temporal subgraphs, each spanning 20 consecutive days. The initial model training is conducted on a subgraph containing the first 20 days of transactions, a portion of which is reserved as a test set for evaluation. We use additional 20-day snapshots from subsequent periods to test time-shift generalization in the inductive setting. These later snapshots vary in their degree of temporal overlap with the training snapshot; 'overlap' here refers to days that are included in both the training snapshot and testing snapshots. In the most challenging setting, the test snapshot comprises the latter 20 days and has no overlapping days with the training set. We evaluate the AUC, Average Precision (AP), and the

Table 5: Overall performance of baselines and KnowGraph. KnowGraph outperformes the baseline DGI [66] in both the transductive and inductive settings. The performance is further improved with PCS framework. We report AUC and average precision with $k = 0.5$, where k is the proportion of the total dataset considered when evaluating average precision. We bold results with an improvement larger than 1%.

Transductive					
Method	AUC	AP	TP@0.5FP	TP@1FP	TP@2FP
GCN [40]	0.953	0.723	0.719	0.742	0.790
DGI [66]	0.892	0.587	0.717	0.738	0.788
KnowGraph	0.978	**0.755**	0.714	0.745	0.788
KnowGraph+PCS	0.968	0.731	**0.731**	**0.757**	**0.814**
Inductive					
Method	AUC	AP	TP@0.5FP	TP@1FP	TP@2FP
GCN [40]	0.501	0.0001	0.0	0.0096	0.0096
DGI [66]	0.498	0.0001	0.0	0.0096	0.0096
KnowGraph	0.609	0.121	0.0048	0.0096	0.0144
KnowGraph+PCS	**0.649**	**0.167**	**0.0287**	**0.0383**	**0.0431**

True Positive rate at a certain False Positive rate (e.g., TP@0.5FP denotes the true positive rate when the false positive rate is 0.5%).

The data is organized into a Heterogeneous Knowledge Graph [28] comprising three core entities in the eBay e-commerce setting – users, items, and transactions. Edges define transactional relationships between buyers and sellers (users) for specific items. The heterogeneous knowledge graph with GCN layers is trained on node graph features with Deep Graph Infomax (DGI) [66] to learn label agnostic node embeddings. Inductive prediction generates heterogeneous node embeddings for the four key entities within the transactional sub-graph: transaction, buyer, seller, and item. These embeddings are combined to form a feature set for the subgraph, which is then input into an XGBoost [10] classifier trained to perform node-level collusion classification. This configuration serves as the primary baseline and main model within KnowGraph, and is also used in the GNN architecture for the knowledge models. It is also the main component in the current automated collusion detection system at eBay, and our overall goal is to improve its performance and generalization. We also compare using only domain knowledge and a simpler GCN [40] baseline trained with supervised learning, which has strong transductive performance but weaker generalization. Since we focus on the inductive setting, it is not used as a model in KnowGraph.

eBay Dataset Challenges. For the transductive setting, the model has access to the entire graph during training but not the collusion labels for nodes in the test set. The model must generalize to a new snapshot with an unseen graph for the inductive setting. We control the difficulty of this setting by changing the amount of overlapping days with the training snapshot. We observe that generalization is challenging in this context due to several reasons:

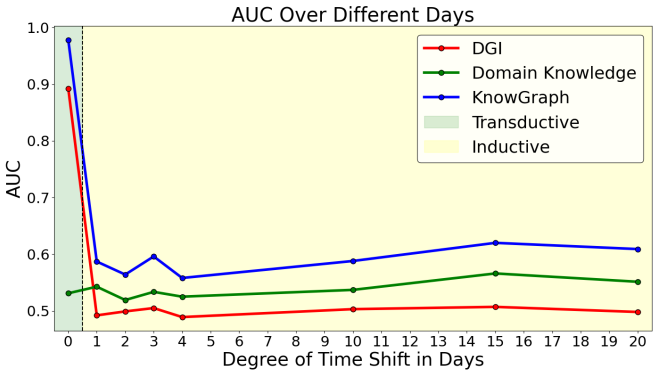

Figure 6: AUC of test graph snapshots with decreasing overlap with the training snapshot. KnowGraph consistently outperforms baselines. We observe similar performance regardless of how different the new graph is, indicating KnowGraph is robust to the magnitude of the time shifts.

- The high volumes of eBay transactions (4-5 million per day) provide substantial changes to the graph structure and label distribution across graph snapshots.
- The noise induced by the inductive sub-graph sampling for graph inference can be significant as the entire graph is too large for tractable message passing.
- The extreme label imbalance coupled with collusive participants who are mostly new to the platform, often lacking prior purchasing or selling history, composes fewer anomalous links in the graph.
- The dataset does not model dynamic time-varying features for users and items, instead relying on static input node features for each such reference entity within a single snapshot.

Due to these challenges, baselines cannot generalize to the inductive setting and have performance close to random guessing. However, while not as effective for the transductive setting, expert-designed knowledge rules have higher performance due to invariance to this distribution shift.

Knowledge models. For the learning component of the framework, we augment the unsupervised embedding model trained on the main classification task with several knowledge models centered around more specific tasks for a total of eight models. Utilizing the fine-grained collusion labels, we train a model for each fine-grained class for four models. These labels are derived from specific instances or cases of collusion. Next, we train a second binary collusion model on upsampled data. These models are independently trained XGBoost classifiers on their respective tasks using the same unsupervised graph embeddings. Finally, we use graph features on buyer and seller account age, zip code, and price to construct two additional models. Unlike the other models, these are matrices directly derived from the graph.

Knowledge rules. For the reasoning component of the framework, we organize the model's predictions into logical rules. The models are predicates of the main model and are organized into the following six rules.

Table 6: Final rule weights of KnowGraph applied to collusion detection on the real-world eBay dataset. We observe similar weights in similar rules, such as the hierarchy rule. Rules with higher magnitude have a larger overall effect on inference.

	ATO	Sign-in	One Day Reg	US Ebay Decline	Exclusion	Ensemble	Account	Price/Shipping
Weight	-1.2227e-02	-1.2221e-02	-1.2221e-02	-1.2221e-02	-2.2177e-10	-1.5172e-02	1.6767e-03	1.5844e-03

Table 7: Knowledge model performance under the transductive and inductive settings for the main task (first two rows) and their own semantic tasks (third row). We report average precision with $k = 0.5$. The knowledge models perform well on their own semantic tasks in general and perform poorly when directly mapped to the main task.

Methods	Main	Secondary	ATO	Sign-in	One Day Reg	US Ebay Decline	Account Age	Business
Transductive	0.587	0.5805	0.0002	0.0011	0.0047	0.0014	0.0434	0.0378
Inductive	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0526	0.0692
Knowledge Task	0.587	0.5805	0.681	0.521	0.353	0.251	1.0	1.0

- *Hierarchy rule* that denotes the hierarchical relation between the binary and collusion subclasses, "$[\texttt{subclass} = \texttt{collusion}] \Rightarrow [\texttt{main} = \texttt{collusion}]$."
- *Exclusion rule* that utilizes the mutually exclusive nature of fine-grained classes, formulated as "$[\texttt{collusion}_i = \texttt{true}] \Rightarrow [(\texttt{collusion}_0 = \texttt{false}) \wedge \cdots \wedge (\texttt{collusion}_{i-1} = \texttt{false}) \wedge (\texttt{collusion}_{i+1} = \texttt{false}) \wedge \cdots \wedge (\texttt{collusion}_n = \texttt{false})]$." where i is the class index up to $n = 4$
- *Ensemble rules* that ensembles the predictions of the two models trained on the overall classification task by ensuring consistency, "$[\texttt{main} = \texttt{collusion}] \Rightarrow [\texttt{secondary} = \texttt{collusion}]$" and "$[\texttt{secondary} = \texttt{collusion}] \Rightarrow [\texttt{main} = \texttt{collusion}]$.", where secondary refers to the model trained on a larger graph with balanced sampling.
- *Business rule* that uses expert knowledge of existing features to pinpoint likely cases of collusion, "$[\texttt{feedback\_amt} < a] \wedge [(\texttt{seller\_age} < b \vee \texttt{buyer\_age} < c)] \Rightarrow \texttt{collusion}$" and "$[(\texttt{gmv} - \texttt{price} > d] \wedge [\texttt{billing\_zip} \neq \texttt{delivery\_zip}] \Rightarrow \texttt{collusion}$", where a, b, c, d are hyperparameters. The business rules and hyperparameters are based on domain expertise and have been verified in the graph.

PCS implementation. To improve inductive generalization and mitigate uncertainty, we implement weight noise ensembling into KnowGraph, following the PCS framework. During the statistical learning component, we add random perturbations sampled from a zero-mean Gaussian distribution with a noise scale of 0.1 to the weights of each GNN model. In the reasoning component, we perform ten forward passes through the GCN for each batch during the E-step of the variational EM algorithm, accumulating the loss and rule scores from each sample and computing their average to obtain a more robust estimate of the variational posterior. As demonstrated by our empirical results, incorporating weight noise ensembling within the PCS framework improves KnowGraph's generalization performance and calibration. We measure calibration with the Expected Calibration Error (ECE), which quantifies the difference between the model's predicted probabilities and the observed frequencies of correct predictions. A lower ECE indicates better calibration and reliability.

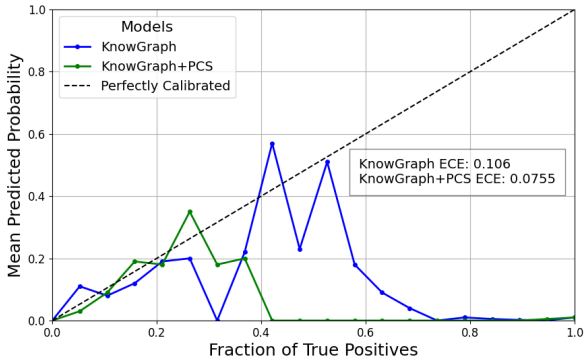

Figure 7: Model calibration for the KnowGraph framework with and without weight noise ensembling for the data-driven models. KnowGraph shows a more volatile calibration curve, deviating significantly from the dashed line representing perfect calibration. In contrast, KnowGraph+PCS maintains a closer alignment to the perfectly calibrated line, suggesting more reliable probability estimates. The ECE values for KnowGraph (0.106) and KnowGraph +PCS (0.0755) quantify these observations, with the latter indicating a more accurately calibrated model.

Main Results. Combining model predictions with the main model in the reasoning framework improves performance in both the transductive and inductive settings. Notably, in Tab. 5 we observe a 0.086 AUC gain and 0.032 AP gain in the transductive setting over the individual main model DGI [66]. This is also comparable to the highest performing baseline for the transductive setting, a GCN [40] trained with supervised learning. There is a more significant gain in the inductive setting, where we observe a 0.111 AUC and 0.120 AP improvement. Furthermore, adding weight noise ensembling from PCS further improves inductive generalization performance to 0.649

AUC and 0.167 AP, a 0.151 and 0.166 gain over the baseline, and the highest performance on this dataset. However, using PCS slightly lowers transductive AUC and AP but still outperforms baselines while improving TP at lower FP rates.

In addition, we find in Fig. 6 that this performance improvement is consistent over various overlapping days, including the fully inductive setting where the model is evaluated on a new graph at a 20-day time shift. Performance appears to be *independent* from the degree the graph shifts, indicating the learned features and rules are robust to changes in the graph in future days. Since it is prohibitively expensive to label new nodes, this is the first time a fully automated pipeline can classify new transactions on this dataset with the potential to assist human experts. However, due to the low performance of the main model, there is still a large gap between transductive and inductive performance, which we leave for future work.

Ablation Studies. The reasoning component of KnowGraph learns a weight for each rule corresponding to their significance on the final classification decision of the framework. We list the associated weight for each rule in Tab. 6. Rules with associated weights with higher magnitudes have a larger effect on inference.

Besides inductive generalization ability, it is also important for models to be well-calibrated; the probability associated with the predicted class label should reflect its ground truth correctness likelihood. We conduct a calibration analysis shown in Fig. 7. In this diagram, the x-axis represents the model's predicted probability (confidence), divided into twenty bins. The y-axis represents the actual fraction of positive instances within each bin. We find in Fig. 7 that the base KnowGraph exhibits higher variability in its predicted probabilities with a noticeable peak around the 0.6 mark on the fraction of true positives, indicating a point of overconfidence in its predictions. In contrast, KnowGraph with PCS remains closer to the perfectly calibrated line, suggesting that it is better calibrated and more consistent in its predictive probabilities across different thresholds. This is also supported by the ECE values, where adding PCS reduces ECE from 0.106 to 0.0755.

6 Discussion and Conclusion

We propose KnowGraph , the first framework that integrates knowledge reasoning with GNNs for enhanced graph-based anomaly detection. KnowGraph combines multiple GNN models operating on different graph structures with a probabilistic logical reasoning component. We employ the PCS framework and introduce weight noise ensembling to mitigate uncertainty and improve the inductive generalization ability of our model. KnowGraph is flexible to various amounts and designs of rules, highly scalable to large graphs, and demonstrates robust performance in challenging scenarios with class imbalance and heterogeneous information. Experiments on two large-scale real-world datasets, an eBay dataset for collusion detection and the LANL network event dataset for intrusion detection, demonstrate KnowGraph's superior performance over state-of-the-art GNN baselines in both transductive and inductive settings. These real-world graph datasets present significant challenges, such as extreme class imbalance, heterogeneous information, and the need for inductive generalization to new graphs. KnowGraph's ability to effectively address these challenges highlights the potential of integrating domain knowledge into data-driven models for high-stakes, graph-based security applications.

KnowGraph's modular architecture paves the way for more accurate and interpretable graph learning systems, bridging the gap between symbolic and neural approaches to graph learning. The reasoning approach introduced in this paper opens up new opportunities for leveraging domain knowledge in various real-world applications. Future research directions include exploring more advanced GNN architectures and investigating alternative inference techniques. Additionally, developing more efficient and scalable reasoning components could further enhance the applicability of KnowGraph to even larger real-world graphs.

7 Acknowledgements

This work is partially supported by the National Science Foundation under grant No. 2046726, NSF AI Institute ACTION No. IIS-2229876, DARPA GARD, the National Aeronautics and Space Administration (NASA) under grant No. 80NSSC20M0229, the Alfred P. Sloan Fellowship, the Meta research award, and the eBay research award.

References

[1] Latifa AlFalahi and Haitham Nobanee. 2019. Conceptual Building of Sustainable Economic Growth and Corporate Bankruptcy. *Economic Growth eJournal* (2019).
[2] Julian Besag. 1977. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika* 64 (1977), 616–618.
[3] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight Uncertainty in Neural Networks. *ArXiv* abs/1505.05424 (2015).
[4] Atul Bohara, Mohammad A. Noureddine, Ahmed M. Fawaz, and William H. Sanders. 2017. An Unsupervised Multi-Detector Approach for Identifying Malicious Lateral Movement. *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)* (2017).
[5] Benjamin Bowman, Craig Laprade, Yuede Ji, and H. Howie Huang. 2020. Detecting Lateral Movement in Enterprise Computer Networks with Unsupervised Graph AI. In *International Symposium on Recent Advances in Intrusion Detection*.
[6] Rüdiger W. Brause, Timm Sebastian Langsdorf, and Hans-Michael Hepp. 1999. Neural data mining for credit card fraud detection. *Proceedings 11th International Conference on Tools with Artificial Intelligence* (1999).
[7] Adam Breuer, Roee Eilat, and Udi Weinsberg. 2020. Friend or Faux: Graph-Based Early Detection of Fake Accounts on Social Networks. *Proceedings of The Web Conference 2020* (2020).
[8] Bokai Cao, Mia Mao, Siim Viidu, and Philip S. Yu. 2017. HitFraud: A Broad Learning Approach for Collective Fraud Detection in Heterogeneous Information Networks. *2017 IEEE International Conference on Data Mining (ICDM)* (2017), 769–774.
[9] N. Chawla, K. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *ArXiv* abs/1106.1813 (2002).
[10] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM. https://doi.org/10.1145/2939672.2939785
[11] William W. Cohen. 1995. Fast Effective Rule Induction. In *International Conference on Machine Learning*. https://api.semanticscholar.org/CorpusID:6492502
[12] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. 2014. Large-Scale Object Classification Using Label Relation Graphs. In *European Conference on Computer Vision*. https://api.semanticscholar.org/CorpusID:10559817
[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPR.2009.5206848
[14] John Dunagan, Alice X. Zheng, and Daniel R. Simon. 2009. Heat-ray: combating identity snowball attacks using machinelearning, combinatorial optimization and attack graphs. In *Symposium on Operating Systems Principles*.
[15] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. 2013. Learning Factored Representations in a Deep Mixture of Experts. *CoRR* abs/1312.4314 (2013).
[16] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. *The World Wide Web Conference* (2019).
[17] Scott Freitas, Andrew Wicker, Duen Horng Chau, and Joshua Neil. 2020. D2M: Dynamic Defense and Modeling of Adversarial Movement in Networks.

arXiv:2001.11108 [cs.SI]
[18] Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. 2016. Credit Card Fraud Detection Using Convolutional Neural Networks. In *International Conference on Neural Information Processing*.
[19] Lise Getoor and Ben Taskar. 2007. Relational Markov Networks. https://api.semanticscholar.org/CorpusID:5600613
[20] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).
[21] Nezihe Merve Gürel, Xiangyu Qi, Luka Rimanic, Ce Zhang, and Bo Li. 2022. Knowledge Enhanced Machine Learning Pipeline against Diverse Adversarial Attacks. arXiv:2106.06235 [cs.LG]
[22] Aric A. Hagberg, Nathan Lemons, Alexander D. Kent, and Joshua Neil. 2014. Connected Components and Credential Hopping in Authentication Graphs. *2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems* (2014).
[23] William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive Representation Learning on Large Graphs. arXiv:1706.02216 [cs.SI]
[24] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. 2022. G-Mixup: Graph Data Augmentation for Graph Classification. *ArXiv* abs/2202.07179 (2022).
[25] Haibo He and Edwardo A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21 (2009).
[26] Knut Hinkelmann, Sajjad Ahmed, and Flávio Corradini. 2022. Combining Machine Learning with Knowledge Engineering to detect Fake News in Social Networks - A Survey. *ArXiv* abs/2201.08032 (2022).
[27] Grant Ho, Mayank Dhiman, Devdatta Akhawe, Vern Paxson, Stefan Savage, Geoffrey M. Voelker, and David A. Wagner. 2021. Hopper: Modeling and Detecting Lateral Movement (Extended Report). *ArXiv* abs/2105.13442 (2021).
[28] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge Graphs. *Comput. Surveys* 54, 4 (July 2021), 1–37. https://doi.org/10.1145/3447772
[29] Bryan Hooi, Neil Shah, Alex Beutel, Stephan Günnemann, Leman Akoglu, Mohit Kumar, Disha Makhija, and Christos Faloutsos. 2015. BIRDNEST: Bayesian Inference for Ratings-Fraud Detection. *ArXiv* abs/1511.06030 (2015).
[30] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, C. Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2022).
[31] Fenyu Hu, Liping Wang, Shu Wu, Liang Wang, and Tieniu Tan. 2021. Graph Classification by Mixture of Diverse Experts. arXiv:2103.15622 [cs.LG]
[32] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. Learning Deep Representation for Imbalanced Classification. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).
[33] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive Mixtures of Local Experts. *Neural Computation* 3 (1991), 79–87.
[34] Weiwei Jiang and Jiayun Luo. 2021. Graph Neural Network for Traffic Forecasting: A Survey. *ArXiv* abs/2101.11174 (2021). https://api.semanticscholar.org/CorpusID:231718752
[35] Johannes Jurgovsky, Michael Granitzer, Konstantin Ziegler, Sylvie Calabretto, Pierre-Edouard Portier, Liyun He-Guelton, and Olivier Caelen. 2018. Sequence classification for credit-card fraud detection. *Expert Syst. Appl.* 100 (2018), 234–245.
[36] Alexander D Kent. 2016. Cyber security data sources for dynamic network research. In *Dynamic Networks and Cyber-Security*. World Scientific, 37–65.
[37] Alexander D. Kent, Lorie M. Liebrock, and Joshua Neil. 2015. Authentication graphs: Analyzing user behavior within an enterprise network. *Comput. Secur.* 48 (2015).
[38] Kristian Kersting and Luc De Raedt. 2001. Towards Combining Inductive Logic Programming with Bayesian Networks. In *International Conference on Inductive Logic Programming*. https://api.semanticscholar.org/CorpusID:15085278
[39] Isaiah J King and H Howie Huang. 2023. Euler: Detecting network lateral movement via scalable temporal link prediction. *ACM Transactions on Privacy and Security* (2023).
[40] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.
[41] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. 2019. Spam Review Detection with Graph Convolutional Networks. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019).
[42] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2017. Focal Loss for Dense Object Detection. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017).
[43] Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. 2019. Log2vec: A Heterogeneous Graph Embedding Based Approach for Detecting Cyber Threats within Enterprise. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019).
[44] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. 2018. Heterogeneous Graph Neural Networks for Malicious Account Detection. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018).
[45] Jun Ma and Danqing Zhang. 2018. GraphRAD : A Graph-based Risky Account Detection System.
[46] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z. Sheng, and Hui Xiong. 2021. A Comprehensive Survey on Graph Anomaly Detection With Deep Learning. *IEEE Transactions on Knowledge and Data Engineering* 35 (2021), 12012–12038.
[47] Yihong Ma, Yijun Tian, Nuno Moniz, and N. Chawla. 2023. Class-Imbalanced Learning on Graphs: A Survey. *ArXiv* abs/2304.04300 (2023).
[48] Domician Máté, Rabeea Sadaf, Judit Oláh, József Popp, and Edit Szűcs. 2019. THE EFFECTS OF ACCOUNTABILITY, GOVERNANCE CAPITAL, AND LEGAL ORIGIN ON REPORTED FRAUDS. *Technological and Economic Development of Economy* (2019).
[49] Microsoft. 2010. Introduction to NT LAN Manager (NTLM) Authentication Protocol Specification. Online. Retrieved August 15, 2010.
[50] MITRE. 2020. Mitre att@ck. https://attack.mitre.org/.
[51] Mark Newman, Duncan J. Watts, and Steven H. Strogatz. 2002. Random graph models of social networks. *Proceedings of the National Academy of Sciences of the United States of America* 99 (2002), 2566 – 2572. https://api.semanticscholar.org/CorpusID:7415348
[52] Shu Kay Ng, Thriyambakam Krishnan, and Geoffrey J McLachlan. 2012. The EM algorithm. *Handbook of computational statistics: concepts and methods* (2012), 139–172.
[53] Namyong Park, Fuchen Liu, Purvanshi Mehta, Dana Cristofor, Christos Faloutsos, and Yuxiao Dong. 2022. EvoKG: Jointly Modeling Event Time and Network Structure for Reasoning over Temporal Knowledge Graphs. *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining* (2022). https://api.semanticscholar.org/CorpusID:246828738
[54] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. arXiv:2002.05287 [cs.LG]
[55] Emilie Purvine, John R. Johnson, and Chaomei Lo. 2016. A Graph-Based Impact Metric for Mitigating Lateral Movement Cyber Attacks. *Proceedings of the 2016 ACM Workshop on Automated Decision Making for Active Cyber Defense* (2016).
[56] Susie Xi Rao, Shuai Zhang, Zhichao Han, Zitao Zhang, Wei Min, Zhiyao Chen, Yinan Shan, Yang Zhao, and Ce Zhang. 2020. xFraud: Explainable Fraud Transaction Detection. *Proc. VLDB Endow.* 15 (2020), 427–436.
[57] Matthew Richardson and Pedro M. Domingos. 2006. Markov logic networks. *Machine Learning* 62 (2006), 107–136.
[58] Saharon Rosset, Uzi Murad, Einat Neumann, Yizhak Idan, and Gadi Pinkas. 1999. Discovery of fraud rules for telecommunications—challenges and solutions. In *Knowledge Discovery and Data Mining*.
[59] Paola Sebastiani, María M. Abad-Grau, and Marco Ramoni. 1998. Bayesian networks. *Nature Methods* 12 (1998), 799 – 800. https://api.semanticscholar.org/CorpusID:45016284
[60] Noam M. Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *ArXiv* abs/1701.06538 (2017).
[61] Min Shi, Yufei Tang, Xingquan Zhu, David A. Wilson, and Jianxun Liu. 2020. Multi-Class Imbalanced Graph Convolutional Network Learning. In *International Joint Conference on Artificial Intelligence*.
[62] Hossein Siadati and Nasir D. Memon. 2017. Detecting Structurally Anomalous Logins Within Enterprise Networks. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (2017).
[63] Ke Sun, Zhanxing Zhu, and Zhouchen Lin. 2021. AdaGCN: Adaboosting Graph Convolutional Networks into Deep Models. arXiv:1908.05081 [cs.LG]
[64] Tian Tian, Jun Zhu, Fen Xia, Xin Zhuang, and T. Zhang. 2015. Crowd Fraud Detection in Internet Advertising. *Proceedings of the 24th International Conference on World Wide Web* (2015).
[65] Vincent S. Tseng, Jia-Ching Ying, Che-Wei Huang, Yimin Kao, and Kuan-Ta Chen. 2015. FrauDetector: A Graph-Mining-based Framework for Fraudulent Phone Call Detection. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015).
[66] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. In *International Conference on Learning Representations*.
[67] G. Wang and Jian Ma. 2012. A hybrid ensemble approach for enterprise credit risk assessment based on Support Vector Machine. *Expert Syst. Appl.* 39 (2012), 5325–5331. https://api.semanticscholar.org/CorpusID:18223080
[68] Haotao Wang, Ziyu Jiang, Yuning You, Yan Han, Gaowen Liu, Jayanth Srinivasa, Ramana Rao Kompella, and Zhangyang Wang. 2023. Graph Mixture of Experts: Learning on Large-Scale Graphs with Explicit Diversity Modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*.
[69] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. 2019. FdGars: Fraudster Detection via Graph Convolutional Networks in Online App Review

System. *Companion Proceedings of The 2019 World Wide Web Conference* (2019).
[70] Shuhao Wang, Cancheng Liu, Xiang Gao, Hongtao Qu, and Wei Xu. 2017. Session-Based Fraud Detection in Online E-Commerce Transactions Using Recurrent Neural Networks. In *ECML/PKDD*.
[71] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, Philip S. Yu, and Yanfang Ye. 2019. Heterogeneous Graph Attention Network. *The World Wide Web Conference* (2019).
[72] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. 2019. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. *ArXiv* abs/1908.02591 (2019).
[73] Wenqi Wei, Mu Qiao, and Divyesh Jadav. 2023. GNN-Ensemble: Towards Random Decision Graph Neural Networks. arXiv:2303.11376 [cs.LG]
[74] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. 2022. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *2022 ieee symposium on security and privacy (sp)*. IEEE, 2005–2024.
[75] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32 (2019), 4–24.
[76] Yaguara. 2023. 46+ eBay Statistics For 2024 (Users, Data & Latest Trends). https://www.yaguara.co/ebay-statistics/ Accessed: 2024-04-24.
[77] Zhuolin Yang, Zhikuan Zhao, Boxin Wang, Jiawei Zhang, Linyi Li, Hengzhi Pei, Bojan Karlaš, Ji Liu, Heng Guo, Ce Zhang, and Bo Li. 2022. Improving Certified Robustness via Statistical Learning with Logical Reasoning. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.).
[78] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Bad for Graph Representation? arXiv:2106.05234 [cs.LG]
[79] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph Contrastive Learning Automated. In *International Conference on Machine Learning*.
[80] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph Contrastive Learning with Augmentations. In *Neural Information Processing Systems*.
[81] Bin Yu. 2013. Stability. *Bernoulli* 19(4) (2013), 1484–1500.
[82] Bin Yu. 2023. What is uncertainty in a data science life cycle? *Journal of Econometrics* 237 (2023), 105529.
[83] Bin Yu and Rebecca Barter. 2024. *Veridical Data Science: The Practice of Responsible Data Analysis and Decision Making*. MIT Press, Cambridge, MA.
[84] Bin Yu and Karl Kumbier. 2020. Veridical data science. *Proceedings of the National Academy of Sciences of the United States of America* 117 (2020), 3920 – 3929.
[85] Hanqing Zeng, Hanjia Lyu, Diyi Hu, Yinglong Xia, and Jiebo Luo. 2023. Mixture of Weak & Strong Experts on Graphs. arXiv:2311.05185 [cs.LG]
[86] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. arXiv:1907.04931 [cs.LG]
[87] Jiawei Zhang, Linyi Li, Ce Zhang, and Bo Li. 2022. CARE: Certifiably Robust Learning with Reasoning via Variational Inference. arXiv:2209.05055 [cs.LG]
[88] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems* 31 (2018).
[89] Ruinan Zhang, Fanglan Zheng, and Wei Min. 2018. Sequential Behavioral Data Processing Using Deep Learning and the Markov Transition Field in Online Fraud Detection. *ArXiv* abs/1808.05329 (2018).
[90] Zehong Zhang, Lifan Chen, Feisheng Zhong, Dingyan Wang, Jiaxin Jiang, Sulin Zhang, Hualiang Jiang, Mingyue Zheng, and Xutong Li. 2022. Graph neural network approaches for drug-target interactions. *Current opinion in structural biology* 73 (2022), 102327. https://api.semanticscholar.org/CorpusID:246173100
[91] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver J. Woodford, Meng Jiang, and Neil Shah. 2020. Data Augmentation for Graph Neural Networks. In *AAAI Conference on Artificial Intelligence*.
[92] Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Financial Defaulter Detection on Online Credit Payment via Multi-view Attributed Heterogeneous Information Network. *Proceedings of The Web Conference 2020* (2020).
[93] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph Neural Networks: A Review of Methods and Applications. *ArXiv* abs/1812.08434 (2018). https://api.semanticscholar.org/CorpusID:56517517
[94] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. arXiv:2006.11468 [cs.LG]