Effective Fault Scenario Identification for Communication Networks via Knowledge-Enhanced Graph Neural Networks

Haihong Zhao, Bo Yang, Jiaxu Cui, Qianli Xing, Jiaxing Shen, Fujin Zhu, and Jiannong Cao, *Fellow, IEEE*

Abstract—Fault Scenario Identification (FSI) is a challenging task that aims to automatically identify the fault types in communication networks from massive alarms to guarantee effective fault recoveries. Existing methods are developed based on rules, which are not accurate enough due to the mismatching issue. In this paper, we propose an effective method named Knowledge-Enhanced Graph Neural Network (KE-GNN), the main idea of which is to integrate the advantages of both the rules and GNN. This work is the first work that employs GNN and rules to tackle the FSI task. Specifically, we encode knowledge using propositional logic and map them into a knowledge space. Then, we elaborately design a teacher-student scheme to minimize the distance between the knowledge embedding and the prediction of GNN, integrating knowledge and enhancing the GNN. To validate the performance of the proposed method, we collected and labeled three real-world 5G fault scenario datasets. Extensive evaluation conducted on these datasets indicates that our method achieves the best performance compared with other representative methods, improving the accuracy by up to 8.10%. Furthermore, the proposed method achieves the best performance against a small dataset setting and can be effectively applied to a new carrier site with a different topology structure.

Index Terms—Communication Networks, Fault Scenario Identification, Knowledge, Propositional Logic, Graph Neural Network.

1 Introduction

Recently, the rapid development of communication networks impressively revolutions our lives. At the same time, new advanced communication networks deepen the difficulty of network fault management [1]. In advanced communication networks (e.g., 5G networks), the high density at both the hardware level and software level leads to large amounts of alarms [2], which are the reflection of network faults. Such large amounts of faults are hard to be accurately identified by human operators effectively, which will significantly impact the network performance.

Fault Scenario Identification (FSI) is an emerging task for better processing network faults/alarms and keeping networks running healthily [3], [4], [5], [6], [7], [8], [9]. It aims to effectively identify concrete fault scenario types from large amounts of alarm sets. In terms of its benefits, it can help network operators complete fast fault recoveries by giving definite fault scenario types in some situations without experiencing the fault localization step. It can also improve existing fault localization methods by helping focus on potential devices. Due to the complexity and heterogeneity of advanced communication networks [10], it is challenging to design an effective FSI method. In academic fields, there is no extensive research on the FSI task. In industry, network practitioners have accumulated a lot of expert knowledge that can easily and directly transform into rules. Thus, most of the general FSI methods are based on rules that are regarded as the way to represent knowledge [3], [4], [5].

However, when employing the rule-based methods to deal with real-world fault scenarios, the methods usually encounter two problems. The first problem is that sometimes the rules fail to match the alarm. The second problem is that some rules are conflicted with each other. There are two reasons that lead to the first problem. On the one hand, some network alarms are delayed or stopped to report due to several reasons, such as network congestion and communication interruption. On the other hand, designing a set of general rules to cover all fault scenario instances is impractical due to the network complexity and heterogeneity. The second problem is derived from the first problem. When general rules cannot tackle some specific fault scenario instances, many specific rules are designed. With more and more specific rules generating, it is hard to guarantee that the designed rules are not conflicted with each other.

To address these issues, we consider introducing deep learning technology into FSI method design. Since a deep learning method is based on the constructed neural net-

(Corresponding author: Bo Yang and Qianli Xing.)

- *H. Zhao is with the Key Laboratory of Symbolic Computation and Knowledge Engineer of Ministry of Education, Jilin University, China, and with the School of Artificial Intelligence, Jilin University, Changchun, Jilin 130012, China. And he is also with the Hong Kong University of Science and Technology (Guangzhou), Guangzhou, Guangdong 510650, China. E-mail: zhaohh19@mails.jlu.edu.cn/hzhaobf@connect.ust.hk.*
- *B. Yang, J. Cui, and Q. Xing are with the Key Laboratory of Symbolic Computation and Knowledge Engineer of Ministry of Education, Jilin University, China, and with the School of Computer Science and Technology, Jilin University, Changhun, Jilin 130012, China. E-mail: {ybo, cjx, qianlixing}@jlu.edu.cn.*
- *J. Shen is with the Department of Computing and Decision Sciences, Lingnan University. E-mail: jiaxingshen@LN.edu.hk.*
- *F. Zhu is with the ICT P&S General Development Dept. Huawei Technologies Co., Ltd., Shenzhen 518129, China. E-mail: zhufujin@huawei.com.*
- *J. Cao is with The Hong Kong Polytechnic University, Hung Hom, Hong Kong. E-mail: csjcao@comp.polyu.edu.hk.*

works to learn a function, which is similar to simulating a large number of rules, they can avoid confronting the two problems of rule-based methods to some extent. Additionally, deep learning methods are likely to infer fault scenarios accurately, where the rules cannot cover. Moreover, they can easily integrate diverse information, such as alarm types, occurrence locations, attribute information, and physical topology. However, deep learning-based methods are difficult to distinguish similar scenarios with slight differences. By contrast, rule-based methods can accurately identify similar fault scenarios through activating different rules. Therefore, to guarantee identification effectiveness, we consider integrating knowledge into deep learning methods to combine the advantages of both fields.

In this work, we propose an effective method named Knowledge-Enhanced Graph Neural Network (KE-GNN). We believe that this is the first work that employs GNN and rules to tackle the FSI task. There are two main challenging problems for integrating knowledge into deep learning methods to deal with the FSI task: (1) How to express complex knowledge related to identifying fault scenarios in the communication networks; (2) How to integrate knowledge and deep learning models effectively? For the first problem, we change the knowledge into propositional formulae (rules) by propositional logic. And then, we project the rules onto a manifold by an encoder to get continuous vector representations, which can help enhance the deep learning model in later steps. For the second problem, we employ the graph neural networks (GNNs) [11], [12], [13], [14] as the deep learning model to identify faults by capturing the natural network topologies. Then, we design a teacher-student scheme to inject knowledge into deep learning. The teacher-student scheme uses knowledge rules (as the teacher) to modify the weights in GNN (as the student) through adding a regularization term to the training objective. The main contributions of this paper are summarized as follows:

1) We propose a hierarchical knowledge expression method, especially for complex knowledge. It can divide knowledge into three levels and adopt propositional logic to generate a series of rules with the purpose of preparing the essential support for integrating GNN.
2) We design a teacher-student scheme to connect knowledge and GNN, which can help GNN capture certain knowledge and increase fault scenario identification accuracy.
3) We collect three real-world fault scenario datasets from the 5G networks and build extensive experiments on them. The experimental results show that KE-GNN outperforms other representative methods. Significantly, it can improve the accuracy by 5.41%$\sim$9.83% compared with others. Furthermore, the proposed method achieves the best performance when the size of the dataset is small and applied to a new carrier site with a different topology structure.

The rest of the paper is organized as follows. Section 2 describes preliminaries. Section 3 describes the concrete structure of KE-GNN. Section 4 evaluates the performance of our proposed method. Section 5 describes related work. Finally, Section 6 and Section 7 concludes the paper and presents future works. Note that in this paper, we focus on the 5G communication networks.

2 PRELIMINARIES

In this section, we introduce the necessary preliminaries, including Network Topology, Alarm, Fault Scenario, Fault Scenario Graph, and Logical Formulae. The network topology represents the actual physical links in communication networks and provides topology information for identifying faults. The alarm is the basic element of a fault scenario. A fault scenario is composed of alarms, topology and some extra information. A fault scenario graph is constructed based on a fault scenario. A logical formula is used to express related knowledge. We will introduce them in detail below.

2.1 Network Topology

The network topology reflects the physical links among devices and components in communication networks, in which three types of devices and four link relationships are mainly considered in this paper. The three types of devices are *Network Element*, *Card*, and *Port*, respectively. Network Element is a basic device with independent transmission function. Card belongs to the Network Element and is the circuit board that carries the functional modules. Port is the I/O port of Card for communication with the outside world. The four link relationships are *Network Element has a Card*, *Card has a Port*, *Port link to Port*, *Network Element link to Network Element*, respectively. Note that a network element usually has several cards, and a card has several ports in physical links. The ports are usually linked to the ports of other network elements for photoelectric signal or information transmission, but there are also a large number of non-enabled ports in the topology. We also add links between two network elements connected by their respective ports to reduce the physical distance between them.

2.2 Alarm

2.2.1 Alarm Type

When the communication network devices or components encounter failures, their core functions are limited or damaged, resulting in alarms. We call these devices or components that generate alarms as reporting positions, including three types of devices, i.e., *Network Element*, *Card*, and *Port*. As alarms are reported at different reporting positions and reflect different failure functions, they usually have various types. Here, we introduce the six representative alarm types related to the subsequent examples in this paper, i.e., *PW Abnormal*, *N N Login*, *N COM*, *TEMP Abnormal*, *E LOS*, *Port Abnormal*, while the rest of alarm types used in experiments can be found in Appendix B. The alarms with types *PW Abnormal*, *N N Login* and *N COM* are reported by *Network Element* when encountering power supply problem, offline status, and gateway switching, respectively. The alarms with *TEMP Abnormal* are emitted by *Card* and reflect the abnormal temperature, and those with *E LOS* are reported by *Port* when it cannot receive any photoelectric signal from the *port* on the other side. The alarms with *Port Abnormal* are also reported by *Port* when the port devices cannot run normally.

TABLE 1
Alarm Data Definition.

Field Name	Information type	Explanation	Concrete Value
Name	Attribute	Alarm Name	E LOS
Network Element Type	Attribute	Product name	PTN 7900
Reporting Time	Attribute	-	44326.80786
Alarm Number	Attribute	Unique alarm number	2021051014340
Alarm Set Number	Attribute	Unique number of different alarm sets	34982
Network Element Name	Location	-	11366-Route(66)
Full Location	Location	Device position that reports the alarm	11366-Route(66)-3-1
Chain Information	Extra	Assert if the network element is in the chain	True
Gateway	Extra	Gateway device that controls the network element	213-BankBoard
Fault Scenario Label	Label	The type of current alarm set labeled by experts	Chain Network Element Log Out

2.2.2 *Alarm Data*

As the topology information is essential to identify the fault, the devices generating alarms usually package the location information into alarm data as the Network Element Name, and Full Location fields. Based on the two pieces of information, we can locate the reporting positions of alarms in network topology and then obtain the concrete topology information. Name is the name of a reported alarm. Additionally, every alarm type has a unique alarm name. Network Element Type is the serial number of a device. Reporting Time includes the time information of an alarm. Chain and Gateway Information belongs to extra information, which can further help FSI in some cases. Alarm Number corresponds to a specific reported alarm. Every alarm belongs to a unique alarm set after the fault detection task that can cluster alarms into different alarm sets. Alarm Set Number corresponds to a unique alarm set that contains the current alarm. Note that the *Fault Scenario Label* field is manually tagged by experts for offline training FSI models. Tab. 1 shows an example of alarm data in our collected three real-world fault scenario datasets. Note that not all the collected alarm data have the extra information because if the collected alarm data has no related fault scenarios that need extra information to identify, it is unnecessary for network operators to offer us extra information.

2.3 Fault Scenario

Different fault scenarios have different combinations of alarms, topologies, and some extra information. The combinations of different information are various, leading to different levels of identification complexity. In this paper, we divide the fault scenarios into three levels:

- **Unique Alarm:** A fault scenario of this level can be easily identified according to a specific alarm. For example, once experts find an alarm, whose type is *TEMP Abnormal*, they can overlook the other information to give the fault scenario, *Equipment Temperature High*, which means that the current device need to be rectified because of its high temperature.
- **Fixed Combination:** If some devices, which report specific alarms, have specific position relations in topology, a fault scenario can be identified. For example, when two linked port devices report two *E LOS* alarms, the fault scenario *Port Double No Light* is determined. The *Port Double No Lights* means that the current link between the two ports encounters a fault.
- **Complex Combination:** The rest fault scenarios of this level have large amounts of combinations of alarms and topologies, which may require extra information, leading to the difficulty of identifying. For example, Identifying *Network Element Out Of Service* needs to judge if the device which reports an *E LOS* has the extra information, *In Chain*. This extra information can help exclude the fault scenario type, *Chain Network Element Log Out*. Note that the *Network Element Out of Service* means that one or several network elements cannot work normally, and *Chain Network Element Log Out* further indicates that the abnormal devices lie in the chain in network topology.

A more detailed description of fault scenarios is shown in Appendix B

2.4 Fault Scenario Graph

A fault scenario graph is denoted as an attributed graph $G = (V, E)$ with node attributes $\boldsymbol{x}_v$ for $v \in V$ and edge attributes $\boldsymbol{e}_{uv}$ for $(u, v) \in E$, where V is a set of vertices and $E \subseteq (V \times V)$ is a set of edges. Nodes represent possible fault locations in communication networks. A node represents a piece of device (i.e., *Network Element*, *Card*, and *Port*) or an alarm (e.g., *E LOS*, *N N Login*, ...). And the attributes represent characteristics of the node. Edges describe physical and logical connections between them. An edge represents a physical connection between two devices (i.e., *Network Element has a Card*, *Card has a Port*, *Network Element link to Network Element*, *Port link to Port*) or a logical connection between a device and an alarm (i.e., *device generate an alarm*). And the attributes represent characteristics of the edge. Fig. 1 gives some examples of fault scenario graphs. In some real-world cases, for better distinguishing the fault scenarios whose level is *Complex Combination*, related extra information is added to every node as attributes, such as *In Chain* information. If the alarm data have no extra information, these features are not added.

2.5 Logical Formulae

Logical statements provide a flexible declarative language for expressing structured knowledge. In this paper, we focus on ***propositional logic***, where a ***proposition*** $\mathbf{p}$ is a statement which is either **True** or **False** [15]. A statement (proposition) consists of subject, predicate and object. It can also be regarded as a ground clause that does not contain

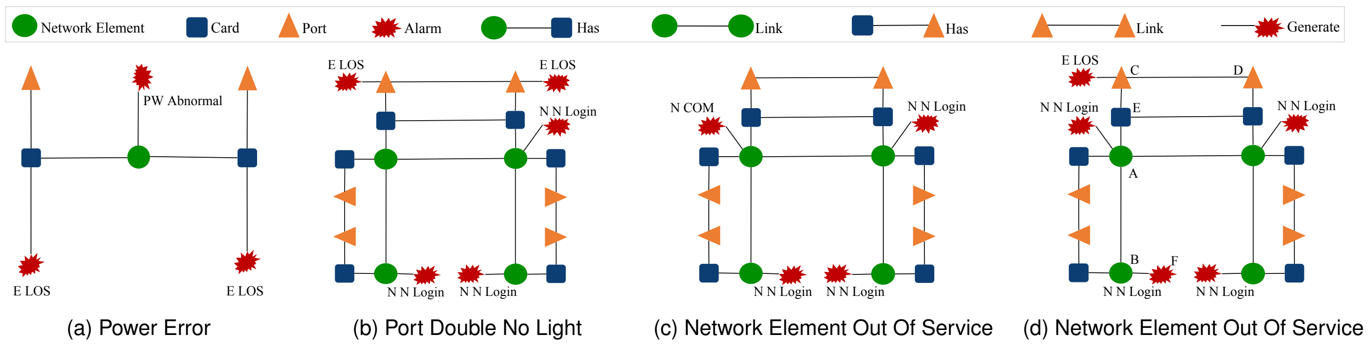


Fig. 1. Fault scenario graph examples for three levels of knowledge. In subfigure (d), *Network Element A* links to *Network Element B*, and *Port C* links to *Port D* in topology. And the edge between an device and an alarm, like *Network Element B* and *N N Login F*, indicates an device generate the alarm. Besides, *Network Element A* has a *Card E* device. *Card E* has a *Port C*.

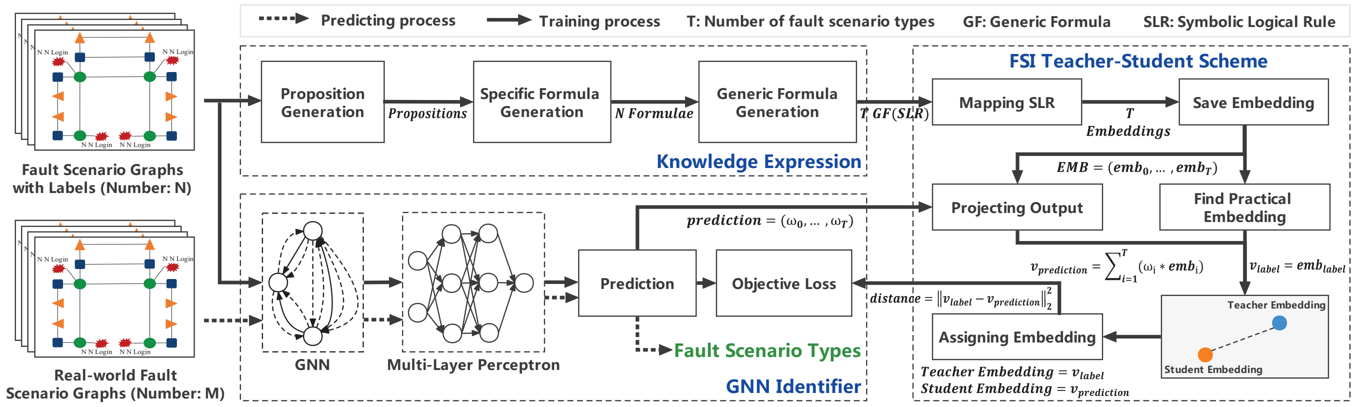

Predicting process
Training process
T: Number of fault scenario types
GF: Generic Formula
SLR: Symbolic Logical Rule
Fault Scenario Graphs with Labels (Number: N)
Real-world Fault Scenario Graphs (Number: M)
Proposition Generation
Propositions
Specific Formula Generation
N Formulae
Generic Formula Generation
T GF(SLR)
Knowledge Expression
FSI Teacher-Student Scheme
Mapping SLR
T Embeddings
Save Embedding
EMB = (emb_0, ..., emb_T)
Projecting Output
Find Practical Embedding
prediction = (ω_0, ..., ω_T)
v_prediction = Σ_{i=1}^T (ω_i * emb_i)
v_label = emb_label
Prediction
Objective Loss
distance = ||v_label − v_prediction||_2^2
Assigning Embedding
Teacher Embedding
Student Embedding
Teacher Embedding = v_label
Student Embedding = v_prediction
GNN
Multi-Layer Perceptron
Fault Scenario Types
GNN Identifier

Fig. 2. An overview of the KE-GNN method: *Knowledge Expression* and *FSI Teacher-Student Scheme* modules are used to guide the training of *GNN Identifier* module. Obtaining the trained *GNN Identifier* module, real-world fault scenario graphs can pass through the module and be assigned predicted fault scenario types.

any variables [16]. A *propositional formula* **F** is a compound of propositions connected by logical connectives [17], [18], e.g., $\neg$, $\wedge$, $\vee$, $\Rightarrow$. Also, a propositional formula is equal to a grounding first-order logic formula. The concrete proposition formats designed for FSI are introduced in Section 3.

3 Knowledge-Enhanced Graph Neural Network for Fault Scenario Identification

In this section, we focus on three specific problems: (1) How to design a reasonable knowledge expression module to express the communication network expert knowledge; (2) How to design a scheme for combining the knowledge and the deep learning method; (3) How to design and train an effective deep learning method based on knowledge. To tackle these problems, we propose a method called *KE-GNN*, which includes three modules *Knowledge Expression*, *FSI Teacher-Student Scheme*, and *GNN Identifier*. The overview is shown in Fig. 2. In *Knowledge Expression* module, we use propositional logic to express the FSI knowledge. Then, in the *FSI Teacher-Student Scheme*, we encode the expressed knowledge and then design a scheme to inject the knowledge into *GNN Identifier*. The FSI knowledge is regarded as the teacher, and the GNN Identifier is regarded as the student. Through this scheme, the FSI knowledge embedding and the output of the GNN Identifier are in the same space and becoming comparable. This scheme builds the bridge between *Knowledge Expression* and *GNN Identifier* and enhances the performance of *GNN Identifier*. *GNN Identifier* module is composed of a GNN model and a multi-layer perceptron, which takes the fault scenario graphs with labels as input and output the predictions of fault scenario types. Specifically, the training loss is adjusted by *FSI Teacher-Student Scheme* module. In the following, we will provide the details of the *KE-GNN* method, including *Knowledge Expression*, *FSI teacher-student scheme*, and *GNN Identifier*.

3.1 Knowledge Expression

Based on the propositional logic mentioned in Section 2, we design the Knowledge Expression module, which can generate the formula for fault scenario types. This module has three steps: (1) **Proposition Generation from Fault Scenario Graphs**, (2) **Specific Formula Generation for Fault Scenario Graphs**, and (3) **Generic Formula Generation for Fault Scenario Types**. In the first step, the key information can be expressed by propositional logic according to different propositions. In the second step, we use logical connectives to link different propositions. And the formulae for fault scenario graphs are generated accordingly. Especially, each fault scenario graph corresponds to a specific formula. In the last step, generic formulae for fault scenarios are obtained. Specifically, the formulae for fault scenario graphs are further classified into different groups based on the fault scenario types. Then, a generic formula for a fault scenario is obtained by refining the formulae in the same group through disjunction logical connectives (i.e., $\vee$). The

TABLE 2
Relation Predicate Definition.

Predicate	Function	Example Tuple
Ne Next To Ne	Two alarms are reported respectively by two Network Element devices, which are linked to each other.	Ne Next To Ne (N N Login, N N Login)
Twice Ne Next To NE	Two alarms are reported respectively by two Network Element devices, which are twice-order neighbors in the graph..	Twice Ne Next To Ne (N N Login, N N Login)
Port Next To Port	Two ports are linked to each other report two alarms, respectively.	Port Next To Port (E LOS, Port Abnormal)
Same Ne	Two alarms in the same Network Element device.	Same Ne (N N Login, E LOS)
Same Card	Two alarms in the same Card devices.	Same Card (TEMP Abnormal, E LOS)
Same Port	Two alarms in the same Port devices.	Same Port (E LOS, Port Abnormal)
In Chain	An alarm is reported by a Network Element device that lies in the network chain.	In Chain (E LOS)
Same Gateway	Two alarms are reported respectively by two Network Element devices belonging to the same gateway.	Same Gateway (N N Login, N N Login)

rest contents of this subsection detailed describe these three steps.

3.1.1 Proposition Generation from Fault Scenario Graphs

The key information includes the topology (or extra) relations between alarms, the behavior of reporting alarms, and the phenomenon of happening fault scenarios. It is firstly obtained from network operational experts. Then, we express it as different propositions by propositional logic. There are overall three types of propositions to illustrate the key information. Note that Section 2 mentions that a proposition consists of three components (a predicate, a subject, and an object). The subject and object can be alarm names or fault scenario names, and a predicate can combine them to a proposition. Therefore, we defined three types of predicates as the following. **Relation Predicate:** It describes the position relation in topology or the environment relation between the devices that report alarms. The environment relation corresponds to the extra information (i.e., Chain Information and Gateway) mentioned in Section 2.2.2. An example of ground relation predicate, *Ne Next To Ne (E LOS, N N Login)* , describes that the Network Element devices are linked to each other. They or their inside devices respectively report *E LOS* and *N N Login*. Any alarm type can act as the object when grounding the relation predicate. Tab. 2 shows all the defined relation predicates. **Happen Predicate:** It indicates that a fault scenario instance of a specific type happens. For example, *Happen (Power Error)* describes that a network device happens a fault scenario, Power Error. The objects of this predicate are composed of fault scenario types. **Report Predicate:** It aims to describe an alarm reported by a device. When only a fault scenario graph has one alarm, we adopt Report Predicate. For example, *Report (E LOS)* means an *E LOS* alarm is reported by a device in a fault scenario graph.

According to predicate definitions, the three types of propositions (relation proposition, happen proposition, and report proposition) are naturally defined. Therefore, applying the related definitions to express the extracted key information can generate propositions accordingly for fault scenario graphs.

3.1.2 Specific Formulae Generation for Fault Scenario Graphs

After the proposition generation, specific formulae are generated by processing the fault scenario graphs. There are three ways to form these formulae according to the three-level identification complexity defined in Section 2.3. The following three examples show the detailed description. **Unique Alarm:** If an $PW\ Abnormal$ alarm is reported, the fault scenario can be directly identified as $Power\ Error$. The formula can be denoted as $Report\ (PW\ Abnormal) \implies Happen\ (Power\ Error)$. **Fixed Combination:** If there is a specific ground relation predicate, *Port Next To Port (E LOS, E LOS)*, the type of the fault scenario, *Port Double No Light*, is determined. This can be formed as $Port\ Next\ To\ Port\ (E\ LOS, E\ LOS) \implies Happen\ (Port\ Double\ No\ Light)$. **Complex Combination:** The formulae of two fault scenario graphs shown in Fig. 1c and Fig. 1d, are formed by Formula. (1) and Formula. (2).

$$\begin{aligned} &Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ N\ Login) \wedge \\ &Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ COM) \\ &\implies Happen\ (Network\ Element\ Out\ Of\ Service) \end{aligned} \tag{1}$$

$$\begin{aligned} &Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ N\ Login) \wedge \\ &Same\ Ne\ (E\ LOS,\ N\ N\ Login) \wedge \\ &Ne\ Next\ To\ Ne\ (N\ N\ Login,\ E\ LOS) \\ &\implies Happen\ (Network\ Element\ Out\ Of\ Service) \end{aligned} \tag{2}$$

3.1.3 Generic Formula Generation For Fault Scenario Types

After the specific formula generation, we firstly classify these specific formulae into different groups based on their fault scenario types. Then, in each group, we use a formula signed as f_{ij} to express the $j-th$ formula in the $i-th$ group. For better understanding, the f_{ij} can be regarded as the specific formula generated from the $j-th$ fault scenario instance of the $i-th$ fault scenario type. After that, we compose the formulae in groups. Finally, for each fault scenario type, we generate a unique composed formula $f_i = \vee_j f_{ij}$.

When the level of a fault scenario type i is *Unique Alarm*, different fault scenario instances of this type form the same formulae. So the composed formula can be extracted from any fault scenario instance j, i.e., $f_i = \vee_j f_{ij} = f_{ij}$. For example, $Report\ (PW\ Abnormal) \implies Happen\ (Power\ Error)$ shown in Fig. 1a form the unique formula for identifying *Power Error* fault scenario.

For the *Fixed Combination* level, the formula is composed in the same way with *Unique Alarm*. For example, $Port\ Next\ To\ Port\ (E\ LOS, E\ LOS) \implies Happen\ (Port\ Double\ No\ Light)$ define the unique formula for $Port\ Double\ No\ Light$ fault scenario identification, shown by Fig. 1b.

The rest formulae for the fault scenario types, whose level is *Complex Combination*, are composed of all the specific formulae formed by different fault scenarios. The *Complex Combination* level's fault scenario types are the most complicated among the three levels. And one of the main contributions of this proposed method is to deal with such kinds of fault scenarios. Designing artificial rules for these types requires network experts to summarize related knowledge from large amounts of fault scenario graphs. However, experts often overlook some essential match items, which leads to the fact that the designed rules cannot cover some fault scenario instances that belong to the same type. On the contrary, the process of generic formula generation is completely automatic. Although the automatically extracted generic formula is complicated for people to understand, it is easy for deep learning methods to learn from them. Besides, they can represent related FSI knowledge inside and out in a more comprehensive way.

In this paper, we mainly focus on three kinds of complex fault scenarios (i.e., *Network Element Out Of Service*, *Chain Network Element Log Out*, and *Inside Network Element Log Out*). For example, Formula. (1) and Formula. (2) are composed by $\vee$ operator. And they can be changed into the form as Formula. (3). If only two specific formulae belong to *Netowrk Element Out Of Service*, the composed formula is Formula. (4). This formula is perceptibly more complex than the other rules. Some composed formulae may contain hundreds of ground clauses because of large amounts of specific formulae generated for fault scenario graphs.

$$\begin{aligned}
&\neg Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ N\ Login) \vee \\
&\neg Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ COM) \vee \\
&Happen\ (Network\ Element\ Out\ Of\ Service) \\
&\vee \\
&\neg Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ N\ Login) \vee \\
&\neg Same\ Ne\ (E\ LOS,\ N\ N\ Login) \vee \\
&\neg Ne\ Next\ To\ Ne\ (N\ N\ Login,\ E\ LOS) \vee \\
&Happen\ (Network\ Element\ Out\ Of\ Service)
\end{aligned} \tag{3}$$

$$\begin{aligned}
&Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ N\ Login) \wedge \\
&Same\ Ne\ (E\ LOS,\ N\ N\ Login) \wedge \\
&Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ COM) \wedge \\
&Ne\ Next\ To\ Ne\ (N\ N\ Login,\ E\ LOS) \\
&\implies Happen\ (Network\ Element\ Out\ Of\ Service)
\end{aligned} \tag{4}$$

The overall processes of knowledge expression are summarized in Algorithm 1. Through Algorithm 1, we finally get a generic formula for each type of fault scenario.

Algorithm 1 Knowledge Expression Module

```
Input:
    $G$: Fault scenario graph set;
    $L$: Fault scenario graph label set;
    $t$: The number of fault scenario types;
Output:
    $F$: Generic formulae for fault scenario types;
    $SPF$: The basic training data applied by the module Mapping SLR
1: initialize $f_1, ..., f_t$ as Empty Formula to represent the generic formulae of $t$ fault scenario types, respectively;
2: initialize $F$ as Empty Formula Set to include all generic formulae;
3: initialize $SPF$ as Empty Formula Set to include the training data of $Encoder(\cdot)$;
4: for all each $l \in L$ and $g \in G$ do
5:     $Observe_p \leftarrow$ Extract all relation and report propositions from $g$
6:     $label_p \leftarrow$ Extract the happen propositions according to $l$;
       {Proposition Generation from Fault Scenario Graphs Section 3.1.1}
7:     $f_{observe} \leftarrow$ Link all the propositions in $Observe_p$ by the logical connective $\wedge$
8:     $temp_f \leftarrow (f_{observe} \implies label_p)$ ;
       {Specific Formula Generation for Fault Scenario Graphs in Section 3.1.2}
9:     $f_l \leftarrow f_l \vee temp_f$;
10:    $SPF \leftarrow SPF \cup temp_f$
11: end for
    {Generic Formula Generation for Fault Scenario Types in Section 3.1.3}
12: $F \leftarrow \{f_1, ..., f_t\}$;
13: return $F, SPF$.
```

3.2 FSI Teacher-Student Scheme

The teacher-student scheme is designed for enabling symbolic logical rules to guide the training of GNN. Note that these rules are represented as formulae by propositional logic in **Knowledge Expression** module. However, there is a gap between symbolic logical rules and deep neural networks because they are two dramatically different things. To eliminate the gap, we project the logical rules and the predictions of deep neural networks (GNN Identifier) to the same logic embedding space. Then, we propose an FSI Teacher-Student Scheme to improve the training process of the GNN Identifier. We assign three parts to detail the scheme. Firstly, we map the symbolic logical rules to an embedding space. And then, we project the output of *GNN Identifier* to the embedding that represents rules. Finally, we can use the rules to guide the training of *GNN Identifier* by the approach of adding a regularization term.

3.2.1 *Mapping the Symbolic Logical Rules (the Teacher) to An Embedding Space and Save the Embedding*

We utilize a graph structure to represent a formula f_i and then construct a multi-layer Graph Convolutional Network [19] as an encoder to project the formula, $Encoder(f_i)$. Training the $Encoder(\cdot)$, we need the specific formulae generated from fault scenario graphs. The training data of encoder is referred to the second output, SPF, of Algorithm 1. Note that $Encoder(\cdot)$ is trained before the training of *GNN Identifier*. The details of training data and the model detail are shown in Appendix C. Next, we input its generic

Algorithm 2 Compute Constraint of FSI Teacher-Student Scheme

```
Input:
    $F$: N symbolic logic rules for different fault scenarios;
    $\omega$: An N-dim probability vector;
    $EMB = \{emb_i\}_{i=1\sim N}$: The logic embedding set corresponding to $N$ fault scenario type;
    $l$: The label value of the fault scenario graph;
Output:
    $d$: The distance from $v_{prediction}$ to a labeled rule $F_l$;
1: $v_{prediction} \leftarrow (v_1, ..., v_n)$, where $v_1, ..., v_n = 0$;
2: $v_{label} \leftarrow emb_l$;
3: $v_{prediction} \leftarrow (v_{prediction} + \sum_{i=1}^{N}(\omega_i * emb_i))$;
4: $d = \|v_{label} - v_{prediction}\|_2^2$;
5: return $d$;
```

formula to $Encoder(\cdot)$ for each fault scenario type and generate a logic embedding. Finally, we sign these logic embeddings by a set, $EMB = \{emb_i\}_{i=1\sim N}, emb_i = Encoder(f_i)$. f_i means the generic formula represents $i-th$ fault scenario type. emb_i is the corresponding logic embedding computed by $Encoder(\cdot)$. N is the number of fault scenario types.

3.2.2 *Projecting the output of GNN Identifier (the Student) to the Same Embedding Space with Symbolic Logical Rules (the Teacher)*

Completing the mapping process of symbolic logical rules is the basis to eliminate the mentioned gap. Next, we introduce the logic embedding space to the training of *GNN Identifier*. The most important thing to accomplish this process is to find concrete relations between rules and deep neural networks. In other words, whether there are similar insights when the two methods identify fault scenarios.

In this paper, we find the prediction result of *GNN Identifier* is a breakthrough entrance. The rule matching result implies how well the information in a fault scenario graph matches N rules of identifying fault scenario types. Driving the *GNN Identifier* to learn certain real-world FSI knowledge, we consider regarding the N rules as the symbolic logical rules. A deep neural network represents a complex mathematical function. And using the function to identify fault scenarios can be considered as a rule matching process from another perspective. Though the rules included in the function are hard for people to understand, the prediction results are understandable. The prediction is an N-dim probability vector ω. It can be regarded as the rule matching result of *GNN Identifier*. Then, we take an average of N logic embeddings in EMB weighted by the prediction shown by Eq. (5). The obtained embedding $v_{prediction}$ can be seen as the process of mapping a symbolic rule (a specific formula) to the logic embedding space. Therefore, we name it the prediction logic embedding.

$$v_{prediction} = \sum_{i=1}^{N}(\omega_i * emb_i) \tag{5}$$

3.2.3 *Assigning the Embedding of symbolic Logical Rules (the Teacher) to Guide the training of GNN Identifier (the Student)*

The previous steps project symbolic logical rules (the teacher) and the prediction results of *GNN Identifier* (the student) to the same logic embedding space. The next problem is how to inject symbolic logical rules into the prediction results of *GNN Identifier*. In other words, how the teacher teaches the student. To solve this problem, we compute the Euclidean Distance between the prediction logic embedding and the practical logic embedding ($emb_{i=label}$). The computation process is shown by Eq. (6). In the logic embedding space defined by the trained $Encoder(\cdot)$, we assume that the prediction logic embedding (the output of the student) should be close to the practical logic embedding(the embedding of the teacher). Then, the distance is added to the standard objective function (prediction loss) of *GNN Identifier* as a regularization term. When training the *GNN Identifier*, the parameters are updated based on not only the ground truth but also the symbolic logical rules. In this way, we build a bridge between knowledge of rules and *GNN Identifier*.

$$distance = \|emb_{i=label} - v_{prediction}\|_2^2 \tag{6}$$

Algorithm 2 gives the concrete process to compute $distance$ by the proposed FSI Teacher-Student Scheme.

Algorithm 3 Training Process of the KE-GNN

```
Input:
    $G$: Fault scenario graph set;
    $L$: Fault scenario graph label set;
    $t$: The number of fault scenario types
Output:
    $w^*$: Optimal Weights of GNN Identifier;
1: $F, SPF \leftarrow Algorithm1(G, L, t)$;
   /* Knowledge Expression */
2: Train $Encoder(\cdot)$ by formula set extracted from fault scenario graphs $SPF$;
3: initialize $EMB$ as empty set for saving the logic embedding corresponding to N fault scenario type;
4: for all each $f_i \in F$ do
5:     $EMB \leftarrow EMB \cup Encoder(f_i)$
       /*The first step of FSI Teacher-Student Scheme*/
6: end for
7: initialize a GNN Identifier $gnn(\cdot)$ and its weights $w$;
8: for all each $g \in G$ and $l \in L$ do
9:     $pre \leftarrow gnn(g)$;
10:    compute the prediction loss $L_{pre}$ by prediction vector $pre$ and label $l$;
11:    $distance \leftarrow Algorithm2(F, pre, EMB, l)$;
       /*The second and third step of FSI Teacher-Student Scheme.*/
12:    $loss \leftarrow L_{pre} + \lambda \cdot distance$;
13:    $w^* \leftarrow$ update the weight $w$ according to $loss$;
       /*Using the symbolic logical rules to guide the training of GNN Identifier*/
14: end for;
15: return $w^*$;
```

3.3 GNN Identifier

GNN Identifier includes a GNN and a multi-layer perceptron, which aims to identify fault scenario graphs. It takes the fault scenario graphs as input and output predicted fault scenario types. Its standard objective function is based on a prediction loss, L_{pre}. Adding the distance computed by

TABLE 3
Dataset Description.

	FSD1-15	FSD1-13	FSD2-4
Site	City-1	City-1	City-2
Year	2020	2019	2021
Month	July, October, November	November	-
Alarm Number	12548	3902	192
Alarm Type Number	31	20	2
Fault Scenario Instance Number	6335	1982	64
Fault Scenario Type Number	15	13	4
Extra Information*	Yes	No	No

* The extra information mentioned in Section 2.2.2 belongs to supplementary information. It can help experts further distinguish specific fault scenario types, such as *Chain Network Element Log Out*.

FSI Teacher-Student Scheme, the complete training objective function is shown by Eq. (7). In the objective function, λ is a trade-off factor.

$$L_{ke-gnn} = L_{pre} + \lambda \cdot distance \tag{7}$$

Algorithm 3 shows the overall process of training *GNN Identifier*. Additionally, the regularization term in our proposed method is built by the logic embeddings, while in the traditional teacher-student scheme, it is built by the prediction results [20], [21], [22], [23].

Naturally, the *Predicting* stage follows the *Training* stage. It is the stage that finally completes the real-world FSI task. During this stage, *Knowledge Expression* and *FSI Teacher-Student Scheme* modules are skipped and *GNN Identifier* module executes a forward propagation process. For real-world fault scenario graphs without labels passing through the *GNN Identifier*, a concrete fault scenario type is given.

4 Experimental Evaluation

In this section, we conduct extensive experiments to validate the effectiveness of the proposed method by answering the following four questions:

- **Q1:** Whether the performance of KE-GNN can outperform the baselines, and how much does the knowledge enhance GNN?
- **Q2:** Does the value of λ influence the experimental results, and what is the substantial influence?
- **Q3:** How about the identification ability of KE-GNN, especially on a small sample dataset?
- **Q4:** How about the identification ability of KE-GNN when we apply the trained KE-GNN on a new carrier site?

Different GNN models can be adopted by *GNN Identifier*, such as GraphSage, GCN, GAT, DGI, and GIN [11]. In this paper, we apply GAT [24] and GCN [19] to *GNN Identifier*, as they are two of the most popular models. And the corresponding methods are named KE-GAT and KE-GCN separately. Thereafter, in order to answer the four questions, we first introduce the basic *Settings* of the related experiments and then we design four experiments: **Q1 Answer:** Performance analysis and comparison of KE-GNN on FSD1-15, **Q2 Answer:** The impact of the λ value in KE-GNN on FSD1-15, **Q3 Answer:** The identification ability of KE-GNN on a small sample dataset (FSD1-13), **Q4 Answer:** The identification ability of trained KE-GNN on a new carrier site (FSD2-4).

4.1 Settings

4.1.1 Setup & Data Collection

The communication network environments that we conduct experiments on are 5G network environments, which are built based on the technology, Software-defined Packet Transport Network (SPTN) [25]. In our experiments, we focus on three types of devices. The three types are mentioned in Section 2: *Network Element*, *Card*, and *Port*. The number of fault scenario types we obtained is 15. These faults can cause devices to report 35 alarm types totally.

Since there are few public fault scenario datasets collected on the 5G network built by SPTN, we ask a network operator in China for help. The network operator offers three real-world fault scenario datasets (**FSD1-15, FSD1-13, and FSD2-4**). They are collected in two 5G carrier sites that belong to two cities (City-1 and City-2) in China. The City-1 topology without alarms has 200,336 device nodes and 512,019 edges, and the average degree of the topology is approximately 5. The City-2 topology without alarms has 74 device nodes and 117 edges, and the average degree is around 2, which is simply shown by Fig. 3. The alarm information in these datasets has been handled by the fault detection module. This means that every alarm is given an *Alarm Set Number* that is mentioned in Section 2.2.2. Table 3 gives the concrete dataset descriptions. According to the topology and alarm information in these datasets, we build the corresponding quantity of fault scenario graphs.

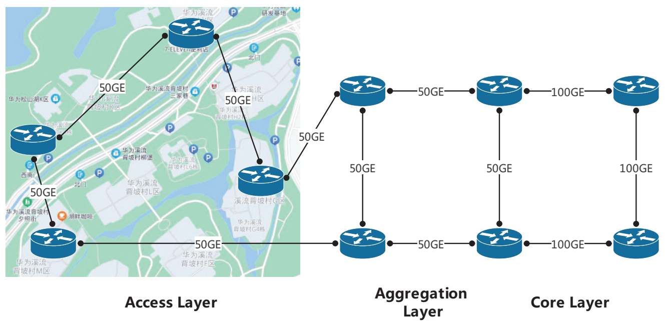


Fig. 3. A simple schematic diagram of City-2 with only showing the links among network element devices.

4.1.2 Manual Data Labeling

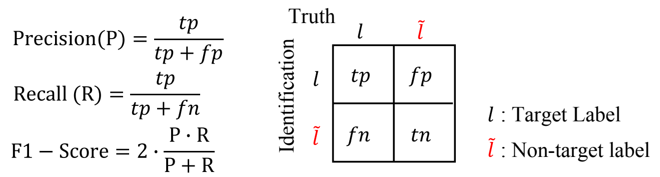
The collected datasets from the network operator have no labels, and it is challenging for us to label these datasets. Firstly, we have to learn large volumes of FSI knowledge in advance from experts. Secondly, it is extremely time-consuming to discuss with network operational experts when we encounter complex fault scenarios. These two challenges limit the scale of our collected datasets, but it is actually sufficient for performance evaluation. More details of why the process of labeling datasets is challenging are further described in Appendix D.

4.1.3 Metrics

We use the identification accuracy to judge the performance of different FSI methods from a macroscopical view. Eq. (8) shows the computation process. True Number means how many fault scenario graphs are identified accurately, and the Total Number means the number of all fault scenario graphs.

$$Accuracy = \frac{True\ Number}{Total\ Number} \tag{8}$$

Further, we select several well-performed methods and use Precision, Recall, and F1-Score to compare their abilities to identify different fault scenario types. Thereafter, we can know which fault scenario types our proposed method can enhance. The related definitions are shown by Fig. 4.



$\text{Precision(P)} = \frac{tp}{tp+fp}$
$\text{Recall (R)} = \frac{tp}{tp+fn}$
$\text{F1} - \text{Score} = 2 \cdot \frac{\text{P} \cdot \text{R}}{\text{P} + \text{R}}$
Truth
Identification
l
$\tilde{l}$
tp
fp
fn
tn
l : Target Label
$\tilde{l}$: Non-target label

Fig. 4. The Target label corresponds to the fault scenario type that we measure. The Non-target label corresponds to other fault scenario types.

4.1.4 Baselines

In this paper, we implement the methods included in [6] and introduce them into our comparison experiments as state-of-the-art, as it is nearly the latest work related to our fault scenario identification task. We implement the methods used in [6], including Bayesian Network (BN), Random Forest (RF), Artificial Neural Network (ANN), and Support Vector Machine (SVM), and the ensemble method of Artificial Neural Network and Support Vector Machine (ANN-SVM), as multi-class forms to tackle the FSI task. Since the XGBoost method [26] is a classical machine learning method together with SVM and RF, we also implement it. Further, we extra implement the rule-based method for the comparison with KE-GNN. Besides, we implement two widely used GNN methods to process graph-based data. Note that it is the two GNNs that we use knowledge to enhance. Table 4 shows the baselines. The implementation detail can be found in Appendix A.

4.2 Q1 Answer: Performance analysis and comparison of KE-GNN on FSD1-15

To evaluate the performance of KE-GCN and KE-GAT, we train them by using the fault scenario dataset, FSD1-15. Then, we compare the performance of KE-GCN and KE-GAT with the performance of baseline methods trained with FSD1-15. In this experiment, we select 756 fault scenario instances as the training set and 5579 fault scenario instances as the test set. Every fault scenario type in the training set and test set has about 50 and 371 instances, respectively. Additionally, we first trained our proposed method on the training set, and then we had a business trip to City-1 to validate the real performance of the proposed method, collecting the test set. This is why the scale of the training set is smaller than the test set.

TABLE 4
Baselines: Nine methods are selected as the baselines in the comparison experiment about identification accuracy, including the state-of-the-art.

Method	Input	Reference
BN	Fault Scenario Variables	[6], [27]
RF	Fault Scenario Features	[6], [28]
ANN	Fault Scenario Features	[6], [29]
SVM	Fault Scenario Features	[6], [30], [31]
ANN-SVM	Fault Scenario Features	[6], [32]
XGBoost	Fault Scenario Features	[26]
Rule-based	Fault Scenario Sequence	-
FsiGCN[1]	Fault Scenario Graphs	[19]
FsiGAT[2]	Fault Scenario Graphs	[24]
The Concrete Settings are illustrated in Appendix A		

[1] It is divided into FsiGCN-NO and FsiGCN-O. FsiGCN-NO is implemented by according to [19], with the same structure. FsiGCN-O is an improved FsiGCN-NO.

[2] It is divided into FsiGAT-NO and FsiGAT-O. FsiGAT-NO is implemented by according to [24], with the same structure. FsiGAT-O is an improved FsiGAT-NO.

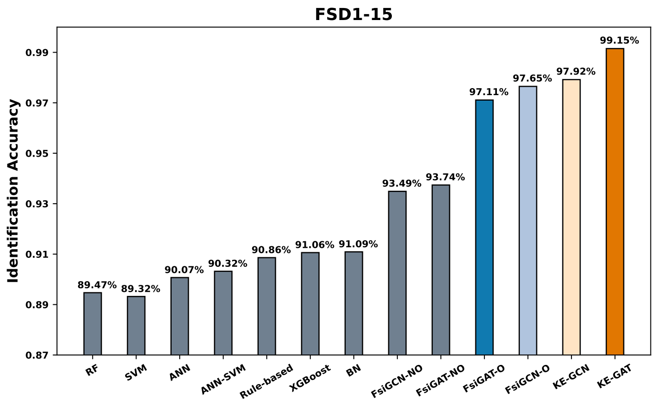


Fig. 5. Accuracy comparison of different representative methods on FSD1-15, including the state-of-the-art.

As illustrated in Fig. 5, KE-GAT outperforms the baseline methods, achieving overall identification accuracy of 8.29% higher than rule-based methods. Though the average accuracy of the other two methods based on GNN can achieve approximately 97.5%, it is still lower than KE-GAT and KE-GCN, showing that knowledge can further enhance not only FsiGAT-O but also FsiGCN-O in different degrees. According to the analysis of Precision, Recall, and F1-score shown in Table 5, Table 6, and Table 7, we further analyze FsiGCN-O, KE-GCN, FsiGAT-O, and KE-GAT.

Obviously, FsiGCN-O completely fails to handle the fault scenarios, *Surrounding Temperature High* and *Port Double No Light*, where belong to the level of *Unique Alarm* and *Fixed Combination*. However, KE-GCN successfully addresses the first one and dramatically improves the effectiveness of *Port Double No Light* with the precision achieving 1.0. The fault scenarios, which belong to *Complex Combination* (*Inside*

TABLE 5
Precision comparison under different fault scenarios for FsiGAT-O, KE-GAT, FsiGCN-O, and KE-GCN. Complex fault scenarios (*Chain Network Element Log Out* and *Inside Network Element Log Out*) are improved noticeably by knowledge. Note that FsiGCN-O completely fails to identify *Surrounding Temperature High* and *Port Double No Light*, while KE-GCN can improve them.

Fault Scenario Type	Method			
	GAT[1]	KE-GAT	GCN[2]	KE-GCN
Network Element Out Of Service	0.995	0.990	0.989	0.991
Card Error	0.993	1.000	0.988	0.999
Power Error	0.520	**0.951 (+82.9%)**	1.000	0.951
Port Self No Light	0.910	**0.966 (+6.2%)**	0.905	0.905
Port Light Error	0.982	0.996	0.992	0.994
Fan Error	0.818	**1.000 (+22.3%)**	1.000	1.000
Light Module Error	1.000	0.500	1.000	1.000
Surrounding Temperature High	1.000	1.000	0.000	**1.000 (+∞)**
Equipment Temperature High	1.000	0.993	1.000	1.000
Inside Network Element Log Out	0.407	**0.667 (+63.9%)**	0.353	**0.394 (+11.6%)**
Port Double No Light	1.000	0.933	0.000	**1.000 (+∞)**
Port Mask Exceed Limitation	0.996	1.000	0.998	1.000
Link Error	0.994	0.991	0.997	0.997
Memory Exceed Limitation	1.000	1.000	1.000	1.000
Chain Network Element Log Out	0.184	**0.909 (+394.0%)**	0.047	**0.190 (+304.3%)**

[1] FsiGAT-O
[2] FsiGCN-O

TABLE 6
Recall comparison under different fault scenarios for FsiGAT-O, KE-GAT, FsiGCN-O, and KE-GCN. Utilizing knowledge, KE-GAT and KE-GCN can improve the recall of four fault scenarios (*Surrounding Temperature High*, *Port Double No Light*, *Inside Network Element Log Out*, and *Chain Network Element Log Out*) together. The four scenarios include two complex scenarios.

Fault Scenario Type	Method			
	GAT[1]	KE-GAT	GCN[2]	KE-GCN
Network Element Out Of Service	0.961	0.995	0.982	0.965
Card Error	0.997	0.994	0.999	0.998
Power Error	1.000	1.000	1.000	1.000
Port Self No Light	0.857	**0.969 (+13.1%)**	0.844	0.844
Port Light Error	0.999	1.000	0.999	0.999
Fan Error	0.900	**1.000 (+11.1%)**	1.000	1.000
Light Module Error	1.000	1.000	1.000	1.000
Surrounding Temperature High	0.769	**1.000 (+30.0%)**	0.000	**1.000 (+∞)**
Equipment Temperature High	0.993	0.998	1.000	1.000
Inside Network Element Log Out	0.647	**0.706 (+9.1%)**	0.706	**0.765 (+8.4%)**
Port Double No Light	0.391	**0.609 (+55.8%)**	0.000	**0.652 (+∞)**
Port Mask Exceed Limitation	0.988	0.990	0.986	0.986
Link Error	0.908	**0.988 (+8.8%)**	0.985	0.982
Memory Exceed Limitation	1.000	1.000	1.000	1.000
Chain Network Element Log Out	0.636	**0.909 (+42.9%)**	0.182	**1.000 (+449.5%)**

[1] FsiGAT-O
[2] FsiGCN-O

Network Element Log Out and *Chain Network Element Log Out*), can be identified better by KE-GCN. These elevations show the ability of knowledge enhancing FsiGCN-O.

In terms of FsiGAT-O and KE-GAT, the difference is noticeable, showing the function of knowledge in FsiGAT-O. From Table 5, we can see nearly all the precision values of KE-GAT are higher than FsiGAT-O, except for the value of *Port Double No Light*. From Table 6 and Table 7, nearly all the other recall and F1-score values of KE-GAT are higher than FsiGAT-O, including the recall and F1-score values of *Port Double No Light*. Since the number of *Light Module Error* instances is small in the test set, a small number of errors can lead to low recall and F1-score values of KE-GAT. In general, from Table 7, we find that knowledge mainly helps FsiGAT-O improve seven fault scenarios (*Power Error*, *Port Self No Light*, *Fan Error*, *Surrounding Temperature High*, *Inside Network Element Log Out*, *Port Double No Light* and *Chain Network Element Log Out*). Note that the fifth and seventh fault scenario types belong to *Complex Combination* level, which is vitally important for network operators. This elevation indicates that knowledge can help improve the identification performance of different fault scenarios, which belong to different levels, including *Complex Combination*.

The above analysis shows the ability of knowledge enhancing FsiGCN-O and FsiGAT-O, especially the improvements of FsiGAT-O. A clearer performance analysis displayed by the confusion matrix can be referred to Appendix E. Next, for vividly understanding the mechanism of the knowledge of enhancing the original GNNs, we take the fault scenario graph shown by Fig. 6 as an example. Without considering the extra information, the graph structure can represent *Chain Network Element Log Out* and *Port Self No Light*. In this case, extra information is the key factor for GNN identifying the two fault scenarios. When we use FsiGAT-O to identify the fault scenario graph, *Chain Network Element Out Of Service*, the graph is identified as a *Port Self No Light*. The reason is that the information in two fault scenario graphs is too similar for FsiGAT-O to distinguish the little difference (i.e., different extra information in just one node of a fault scenario graph). However, KE-GAT can give an accurate identification result. The reason is that KE-GAT captures specific knowledge, which can help distinguish the difference of extra information. Formula (9) and formula (10) show the knowledge of the two fault scenarios.

$$\begin{aligned} &Report\ (E\ LOS) \\ &\Longrightarrow Happen\ (Port\ Self\ No\ Light) \end{aligned} \tag{9}$$

$$\begin{aligned} &Report\ (E\ LOS) \wedge \\ &In\ Chain\ (E\ LOS) \\ &\Longrightarrow Happen\ (Chain\ Network\ Element\ Log\ Out) \end{aligned} \tag{10}$$

TABLE 7
F1-score comparison under different fault scenarios for FsiGAT-O, KE-GAT, FsiGCN-O, and KE-GCN. F1-score can reflect the overall ability to identify different fault scenario types. The bold words in the table show the fault scenarios improved by knowledge.

Fault Scenario Type	Method			
	GAT[1]	KE-GAT	GCN[2]	KE-GCN
Network Element Out Of Service	0.977	0.992	0.986	0.978
Card Error	0.995	0.997	0.993	0.998
Power Error	0.684	**0.975 (+42.5%)**	1.000	0.975
Port Self No Light	0.883	**0.968 (+9.6%)**	0.873	0.873
Port Light Error	0.990	0.998	0.996	0.996
Fan Error	0.857	**1.000 (+16.7%)**	1.000	1.000
Light Module Error	1.000	0.667	1.000	1.000
Surrounding Temperature High	0.870	**1.000 (+17.2%)**	0.000	**1.000 (+∞)**
Equipment Temperature High	0.996	0.995	1.000	1.000
Inside Network Element Log Out	0.500	**0.686 (+37.2%)**	0.471	**0.520 (10.4%)**
Port Double No Light	0.563	**0.737 (+30.9%)**	0.000	**0.789 (+∞)**
Port Mask Exceed Limitation	0.992	0.995	0.992	0.993
Link Error	0.949	**0.990 (+4.3%)**	0.991	0.990
Memory Exceed Limitation	1.000	1.000	1.000	1.000
Chain Network Element Log Out	0.286	**0.909 (+217.8%)**	0.074	**0.319 (+331.1%)**

[1] FsiGAT-O
[2] FsiGCN-O

TABLE 8
Precision Comparison for KE-GAT assigned three different λ values. The bold fault scenario types are improved by KE-GAT with different λ in various degrees, but KE-GAT with $\lambda = 0.1$ performs the best.

Fault Scenario Type	Method			
	GAT[1]	KE-GAT		
		$\lambda = 0.01$	$\lambda = 1$	$\lambda = 0.1$
Network Element Out Of Service	0.995	0.995	0.988	0.990
Card Error	0.993	0.980	1.000	1.000
Power Error	0.520	0.929 (+78.7%)	0.951 (+82.9%)	**0.951 (+82.9%)**
Port Self No Light	0.910	0.910	0.832	**0.966 (+6.2%)**
Port Light Error	0.982	0.980	0.997	0.996
Fan Error	0.818	0.900 (+10.0%)	1.000 (+22.3%)	**1.000 (+22.3%)**
Light Module Error	1.000	1.000	0.333	0.500
Surrounding Temperature High	1.000	1.000	1.000	1.000
Equipment Temperature High	1.000	0.993	0.993	0.993
Inside Network Element Log Out	0.407	0.379	0.333	**0.667 (+63.9%)**
Port Double No Light	1.000	0.714	0.909	0.933
Port Mask Exceed Limitation	0.996	1.000	1.000	1.000
Link Error	0.994	0.997	0.993	0.991
Memory Exceed Limitation	1.000	1.000	1.000	1.000
Chain Network Element Log Out	0.184	0.600 (+226.1%)	0.750 (+307.6%)	**0.909 (+394.0%)**

[1] FsiGAT-O

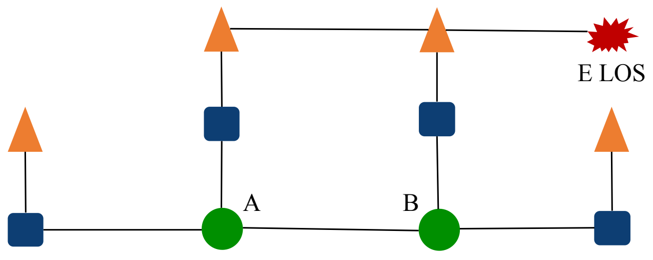


Fig. 6. The fault scenario can represent not only a *Port Self No Light* fault scenario but also a *Chain Network Element Log Out* when the fault scenario graph has no extra information. Note that this figure is just a simple case abstracted from real-world fault scenario graphs. In real-world fault scenarios, the topology structure can be more complex.

4.3 Q2 Answer: The impact of the λ value in KE-GNN on FSD1-15

According to Eq. (7), knowledge can hardly enhance the performance of FsiGAT-O when λ is assigned a small value. Moreover, if the λ is assigned a big value, the FsiGAT-O cannot capture graph structure information well. Therefore, it is necessary to evaluate the impact of λ values, and we need to leverage the λ values to achieve a fine performance. To support our assumptions, we design the following experiment. The segmentation of FSD1-15 is the same as the experimental setting of Q1 Answer.

In this experiment, three different λ values (0.01, 0.1, and 1) are selected. The accuracies of FsiGAT-O, KE-GAT ($\lambda = 1$), KE-GAT ($\lambda = 0.01$), and KE-GAT ($\lambda = 0.1$), respectively are 97.11%, 97.67%, 97.99%, and 99.15%. Note that all three KE-GAT implementations perform better than FsiGAT-O, especially when the $\lambda = 0.1$. Next, we focus on the comparisons among the three implementations. Table 8, Table 9, and Table 10 show the concrete performance of identifying different fault scenario types. Combining the three tables to analyze together, we find that when the λ is assigned 0.01, the improvement of FsiGAT-O is not enough. When $\lambda = 0.1$, KE-GAT achieves the best performance, compared with the other λ settings. When the λ is raised from 0.1 to 1, the overall identification ability of different fault scenario types drops in varying degrees. More intuitive performance analysis by using a bar chart and several confusion matrixes can be found in Appendix E.

From the results of this experiment, we show the impact of different λ values, indicating that not any values can be used to train KE-GAT and $\lambda = 0.1$ can be a good selection.

4.4 Q3 Answer: The identification ability of KE-GNN on a small sample dataset (FSD1-13)

Since it is hard to obtain labeled fault scenario data sometimes, it is necessary for us to evaluate our method on a small training set. We design a small sample experiment on FSD1-13, where the training set has 97 fault scenario instances and the test set has 1885 fault scenario instances. Every fault scenario type in the training set has about 7 and 145 instances, respectively. The instances in training sets are selected manually by experts, which cover the core fault scenarios. Besides, for further validating the effectiveness of KE-GAT and KE-GCN trained with a small training set, we train them with sufficient training data (normal training setting).

TABLE 9
Recall Comparison for KE-GAT assigned three different λ values. Note that when KE-GAT adopts $\lambda = 0.1$, the number of improved fault scenario types doubles. Also, in terms of the overlapped improved types, KE-GAT with $\lambda = 0.1$ performs better.

Fault Scenario Type	Method			
	GAT[1]	KE-GAT		
		$\lambda = 0.01$	$\lambda = 1$	$\lambda = 0.1$
Network Element Out Of Service	0.961	0.990	0.961	0.995
Card Error	0.997	0.994	0.994	0.994
Power Error	1.000	1.000	1.000	1.000
Port Self No Light	0.857	0.959 (+11.9%)	0.963 (+12.4%)	**0.969 (+13.1%)**
Port Light Error	0.999	0.999	0.999	1.000
Fan Error	0.900	0.900	0.900	**1.000 (+11.1%)**
Light Module Error	1.000	1.000	1.000	1.000
Surrounding Temperature High	0.769	0.923 (+20.0%)	1.000 (+30.0%)	**1.000 (+30.0%)**
Equipment Temperature High	0.993	0.998	0.998	0.998
Inside Network Element Log Out	0.647	0.647	0.647	**0.706 (+9.1%)**
Port Double No Light	0.391	0.435 (+11.3%)	0.435 (+11.3%)	**0.609 (+55.8%)**
Port Mask Exceed Limitation	0.988	0.984	0.988	0.990
Link Error	0.908	0.875	0.908	**0.988 (+8.8%)**
Memory Exceed Limitation	1.000	1.000	1.000	1.000
Chain Network Element Log Out	0.636	0.273	0.273	**0.909 (+42.9%)**

[1] FsiGAT-O

TABLE 10
F1-score Comparison for KE-GAT assigned three different λ values. Corresponding to the analysis of precision and recall, KE-GAT with $\lambda = 0.1$ elevates the most quantity of fault scenario types.

Fault Scenario Type	Method			
	GAT[1]	KE-GAT		
		$\lambda = 0.01$	$\lambda = 1$	$\lambda = 0.1$
Network Element Out Of Service	0.977	0.989	0.977	0.992
Card Error	0.995	0.997	0.987	0.997
Power Error	0.684	0.975 (+42.5%)	0.963 (+40.8%)	**0.975 (42.5%)**
Port Self No Light	0.883	0.891	0.936 (+6.0%)	**0.968 (+9.6%)**
Port Light Error	0.990	0.998	0.990	0.998
Fan Error	0.857	0.947 (+10.5%)	0.900 (+5.0%)	**1.000 (+16.7%)**
Light Module Error	1.000	0.500	1.000	0.667
Surrounding Temperature High	0.870	0.960 (+10.3%)	1.000 (+17.2%)	**1.000 (+17.2%)**
Equipment Temperature High	0.996	0.995	0.995	0.995
Inside Network Element Log Out	0.500	0.440	0.478	**0.686 (+37.2%)**
Port Double No Light	0.563	0.588	0.541	**0.737 (+30.9%)**
Port Mask Exceed Limitation	0.992	0.992	0.994	0.995
Link Error	0.949	0.931	0.950	**0.990 (+4.3%)**
Memory Exceed Limitation	1.000	1.000	1.000	1.000
Chain Network Element Log Out	0.286	0.400 (+39.9%)	0.375 (+31.1%)	**0.909 (+217.8%)**

[1] FsiGAT-O

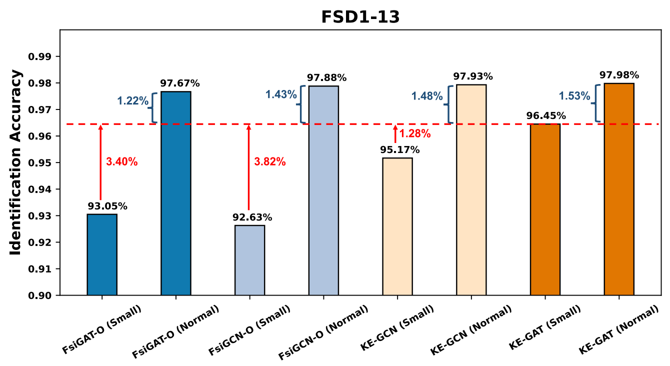


Fig. 7. Identification accuracy comparison on FSD1-13. The red dotted line corresponds to KE-GAT (Small). It shows that the performance of KE-GAT trained with a small sample dataset outperforms FsiGAT-O (Small), FsiGCN-O (Small), and KE-GCN (Small). At the same time, the performance of KE-GAT (Small) is the closest to the methods trained with sufficient sample data. The difference between KE-GAT (Small) and FsiGAT-O (Normal) is only 1.22%.

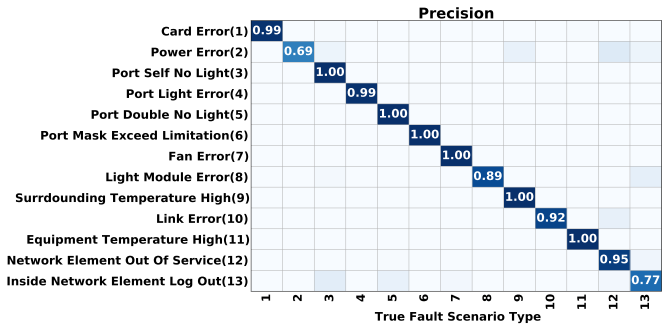


Fig. 8. Confusion Matrix of KE-GAT trained with FSD1-13.

Fig. 7 gives the experimental comparison results. KE-GAT (Small) achieves the best accuracy of 96.45% compared with the methods trained with FSD1-13. It can improve the accuracy of FsiGAT-O (Small) by up 3.4%. FsiGCN-O (Small) can also be enhanced by up 2.5% through knowledge, but the performance of KE-GCN (Small) is worse than KE-GAT (Small). Further, we find that the difference between KE-GAT (Small) and other methods trained with sufficient data is from 1.22%∼1.53%. This means the proposed method trained with the small sample dataset has the ability to improve the performance of FsiGAT-O (Small) and FsiGCN-O (Small). Further, the performance of KE-GAT (Small) is the closest to the other methods trained with sufficient data.

According to the confusion matrix of KE-GAT (Small) shown by Fig. 8, every fault scenario type is tackled relatively satisfactorily. This indicates that KE-GAT under the small sample dataset can also perform well.

4.5 Q4 Answer: The identification ability of trained KE-GNN on a new carrier site (FSD2-4)

For evaluating the identification ability of trained KE-GNN on a new carrier site, we apply the method trained by FSD1-13 directly on FSD2-4, whose network topology in City-2 is different from the topology in City-1. Besides, some extra information, which can help identify complex fault scenario types, varies from City-1 to City-2, such as different gateway configurations. The difference sometimes leads to

TABLE 11
Identification accuracy comparison on FSD2-4: Identification number of the four classes and Total Accuracy.

Sample Scale	Small				Normal			
Method	GAT[1]	KE-GAT	GCN[2]	KE-GCN	GAT[1]	**KE-GAT**	GCN[2]	KE-GCN
Network Element Out Of Service	2/2	2/2	0/2	0/2	2/2	2/2	2/2	2/2
Inside Network Element Log Out	2/10	7/10	6/10	7/10	6/10	10/10	5/10	5/10
Port Self No Light	35/35	35/35	35/35	35/35	35/35	35/35	35/35	35/35
Port Double No Light	17/17	17/17	17/17	17/17	17/17	17/17	17/17	17/17
Accuracy	87.50%	95.31% **(+7.81%)**	90.63%	92.18% **(+1.55%)**	93.75%	**100.00% (+6.25%)**	92.18%	92.18%

[1] FsiGAT-O
[2] FsiGCN-O

the identification failure of rule-based methods in industry, but our proposed method has the ability to make accurate identifications. The reason may be that the proposed method can learn some general local topology information or key features of alarm relations that can also be used in different carrier sites.

The concrete performance is shown by Table 11. The small sample scale corresponds to the methods trained with FSD1-13, such as KE-GAT (Small). The normal sample scale corresponds to the methods trained with sufficient data in the previous section, such as KE-GAT (Normal). From the table, we can find that the proposed method can be directly applied in City-2 without much performance decay, but the performance of FsiGAT-O and FsiGCN-O is unsatisfactory. Note that KE-GAT (Normal) achieves an overall accuracy of 100%. According to the table, almost all the methods suffer from lower performance on the *Inside Network Element Log Out* scenario compared with other scenarios, while all the methods perform well on the *Port Double No Light* scenario. The instances of *Port Double No Light* scenario can be identified well by both common GNN and KE-GNN because they usually contain a small number of alarms (two *E LOS* alarms) in FSD2-4. Differently, the instances of the *Inside Network Element Log Out* scenario contain much more alarms compared with other scenarios in FSD2-4, increasing the complexity of fault scenario identification. Thus, the instances of *Inside Network Element Log Out* scenario are hard to tackle by common GNN. By contrast, our proposed method performs well because of introducing knowledge, such as 100% accuracy of KE-GAT (Normal). Additionally, the reason why the KE-GCN trained with the complete data do not be enhanced is that from Fig. 7, we can find that though knowledge can enhance GCN, the rising degree is not significant.

The experimental results show that our proposed method has a certain generalization ability.

5 Related Work

5.1 Network Fault Management

The flourishing of new advanced communication networks drives the rapid development of network fault management (NFM). In existing NFM, fault detection (FD) and fault localization (FL) are two of the components. FD determines whether the current network works in normal conditions or if a fault has occurred. Generally taking the FD output as the input, FL aims to deduce the exact root cause from a set of observed failure indications (e.g., alarms), finding the target minimum operational unit (e.g., the precise location of a broken fiber in the network) [1], [33].

In the field of communication networks, there are many effective methods applied in the FD task, such as multi-layer perceptron, random forest, and support vector machine (SVM) [6], [7], [8], [9], [34], [35], [36], [37]. Further, in the FL task, Bayes-based methods are popular methods to tackle the FL task [38], [39], [40], [41], [42], [43], [44], [45]. Some methods used in the FD task are also included in the FL task, such as SVM [46], [47]. Next, we give some detailed descriptions of recently proposed methods applied in the FL task. Machine learning-based technique methods (SVM, Random Forest, and Multi-layer perceptron) [47] are utilized to localize the link faults in topology. A Bayesian network is constructed based on sequentiality between alarms, where the Bayesian inference is executed for localizing the probable root cause alarms. Because of the network complexity and heterogeneity, the common problems of current methods in the FL task of advanced communication networks can be: (1) low localization efficiency; (2) identifying probable root cause alarms instead of minimum operational units.

To sum up, existing methods applied in the FD task can perform well in communication networks. Though there is extensive research on the FL task, existing methods have not tackled the FL task satisfactorily in present-day communication networks [3], [4], [5]. In this paper, compared with FD and FL tasks, the FSI task also belongs to NFM, but it is an emerging task [3], [4], [5].

5.2 Graph Neural Networks

GNN has been widely applied for graph representation learning in recent years because of its well-performed characteristics. Existing GNNs are mainly based on the Fourier transform theory of graphs developed by [48] and the message passing mechanism [24]. GCN and GAT are two of the popular representatives [19], [24], [49], [50], [51].

In communication networks, every device links to several other devices, and then large volumes of devices constitute a large-scale network topology. When devices encounter faults and report alarms, the alarms can be regarded as the dynamic nodes linked to the topology, constructing different graphs. These graphs are named the fault scenario graphs defined in Section 2. Therefore, the fault scenario graph can be naturally handled by GNN. Finally, according to the summary of GNN, we mainly pay attention to the application of two popular GNN methods (GCN and GAT) to the FSI task in communication networks.

5.3 Neural-symbolic Systems

The symbolic system and the neural system can make good use of knowledge and data, respectively. However, both knowledge and data are essential for people to make decisions.

Now, AI researchers are growing their interest in combing symbolic and neural systems to integrate the advantages of both systems [52], [53], [54], [55], [56], [57]. The conceptual comparison of neural-symbolic systems with neural and symbolic systems can be stated from three dimensions. **Efficiency:** Neural-symbolic systems can be fast, thus promoting reasoning over large-scale data. **Generalization:** Neural-symbolic systems do not completely depend on massive labeled data, and therefore they have sufficient generalization capabilities. A neural-symbolic system can apply background or expert knowledge to make up for the lack of training data, enabling the model to reach the satisfying performance with decent generalization ability. **Interpretability:** The neural-symbolic system can embody a transparent reasoning process and hence is interpretable [58]. This is particularly useful in some applications such as medical image analysis when people need not only a decision but also the reason and the decision-making process. The neural-symbolic system has become an important approach for explainable artificial intelligence but has not yet been applied to the NFM of communication networks.

6 Conclusion

This paper presents an effective FSI method, called KE-GNN, for communication networks. In order to combine the advantages of the rule-based methods and deep learning methods for identifying fault scenarios, this paper adopts the Propositional Logic to express complex knowledge firstly. Then a Teacher-Student Scheme is proposed for using the knowledge to guide the training of deep learning methods. The experiment results show that knowledge can really enhance the GNN methods, especially the KE-GAT, which outperforms other representative baseline methods and achieves an overall identification accuracy of 99.15% on FSD1-15. When the scale of the selected training set (i.e., FSD1-13) is relatively small, KE-GAT still achieves a competitive accuracy of 96.45%. Moreover, when we directly apply a trained KE-GNN method to a new carrier site (i.e., FSD2-4), the performance of KE-GNN method is still effective compared with the original GNN, showing the better generalization ability. Therefore, KE-GAT is an effective method for the identification of the fault scenario and satisfies the requirements of the advanced communication networks, showing the advantage of KE-GNN.

7 Future work

In this paper, the proposed method can effectively identify fault scenarios, which can be further improved from the technique and the usage scenario aspects.

For the technique aspects, in the future, we would like to be devoted to three research directions about our current design. **(1)** Consider analyzing and improving the inference speed of KE-GNN. **(2)** Consider more strict training settings for small datasets. Few-shot and zero-shot technologies aim to tackle the small sample size problem, and introducing them into KE-GNN is a promising research direction. **(3)** Consider different encoding methods for domain rules to realize a good teacher model could be a potential direction. Other encoding methods, such as sequence data-based methods (e.g., RNN, LSTM) and graph data-based methods, could be exploited to further improve the GNN performance.

For the usage scenario aspect, we can also consider evaluating the proposed method on the other scenarios in communication networks [59], [60], [61], [62]. The optical transport network, another advanced communication network, is a natural choice for us to apply the KE-GNN [62].

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant Nos. 2021ZD0112501 and 2021ZD0112502; the National Natural Science Foundation of China under Grant Nos. U22A2098, 62172185 and 61876069; and Jilin Province Natural Science Foundation under Grant No. 20200201036JC. Also, this work has benefited from the financial support of Lingnan University (LU), Lam Woo Research Fund at LU, and the Hong Kong Institute of Business Studies at LU. This work was also supported by the Research Institute for Artificial Intelligence of Things, The Hong Kong Polytechnic University, and TRS RGC Theme-based Research Scheme (T41-603/20-R).

References

[1] Y. Yu, X. Li, X. Leng, L. Song, K. Bu, Y. Chen, J. Yang, L. Zhang, K. Cheng, and X. Xiao, "Fault management in software-defined networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 349–392, 2019.

[2] J. Moysen and L. Giupponi, "From 4g to 5g: Self-organized network management meets machine learning," *Computer Communications*, vol. 129, pp. 248–268, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366418300380

[3] H. Technologies, "ADN Solution White Paper(Autonomous Driving Network)," *White Paper*, 2020.

[4] Z. Corporation, "Autonomous Evolving Network White Paper," *White Paper*, 2020.

[5] H. Technologies, "Enable Autonomous Driving Network—Huawei Network AI Engine(NAIE) White Paper," *White Paper*, 2019.

[6] A. J. García, M. Toril, P. Oliver, S. Luna-Ramírez, and M. Ortiz, "Automatic alarm prioritization by data mining for fault management in cellular networks," *Expert Systems with Applications*, vol. 158, p. 113526, 2020.

[7] A. T. Bouloutas, S. Calo, and A. Finkel, "Alarm correlation and fault identification in communication networks," *IEEE Transactions on communications*, vol. 42, no. 234, pp. 523–533, 1994.

[8] I. Katzela and M. Schwartz, "Schemes for fault identification in communication networks," *IEEE/ACM Transactions on networking*, vol. 3, no. 6, pp. 753–764, 1995.

[9] S. Kiranyaz, A. Gastli, L. Ben-Brahim, N. Al-Emadi, and M. Gabbouj, "Real-time fault detection and identification for mmc using 1-d convolutional neural networks," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 11, pp. 8760–8771, 2018.

[10] H. Fourati, R. Maaloul, L. Chaari, and M. Jmaiel, "Comprehensive survey on self-organizing cellular network approaches applied to 5g networks," *Computer Networks*, vol. 199, p. 108435, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621003960

[11] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[12] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed, and E. Koh, "Attention models in graphs: A survey," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 6, pp. 1–25, 2019.

[13] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[14] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[15] K. C. Klement, "Propositional logic," in *Internet Encyclopedia of Philosophy*, 2004.

[16] V. Detlovs and K. Podnieks, "Introduction to mathematical logic," *University of Latvia*, 2011.

[17] Y. Xie, Z. Xu, K. S. Meel, M. S. Kankanhalli, and H. Soh, "Embedding symbolic knowledge into deep networks," in *NeurIPS*, 2019.

[18] M. Console, P. Guagliardo, and L. Libkin, "Propositional and predicate logics of incomplete information," *Artificial Intelligence*, vol. 302, p. 103603, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370221001545

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[20] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[21] J. Cui, B. Kingsbury, B. Ramabhadran, G. Saon, T. Sercu, K. Audhkhasi, A. Sethy, M. Nussbaum-Thom, and A. Rosenberg, "Knowledge distillation across ensembles of multilingual models for low-resource languages," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4825–4829.

[22] Y. Yang, J. Qiu, M. Song, D. Tao, and X. Wang, "Distilling knowledge from graph convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[24] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[25] L. Yang, B. Ng, W. K. Seah, L. Groves, and D. Singh, "A survey on network forwarding in software-defined networking," *Journal of Network and Computer Applications*, vol. 176, p. 102947, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520304021

[26] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794. [Online]. Available: https://doi.org/10.1145/2939672.2939785

[27] J. Pearl, "Bayesian networks : A model of self-activated memory for evidential reasoning," *Proc. Cognitive Science Society, 1985*, 1985.

[28] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[29] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[30] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, May 2011. [Online]. Available: https://doi.org/10.1145/1961189.1961199

[31] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999, pp. 61–74.

[32] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[33] M. Nouioua, P. Fournier-Viger, G. He, F. Nouioua, and Z. Min, "A survey of machine learning for network fault management," *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics: Theories and Applications*, p. 1, 2021.

[34] L. Lewis, "A case-based reasoning approach to the management of faults in communication networks," in *IEEE INFOCOM '93 The Conference on Computer Communications, Proceedings*, 1993, pp. 1422–1429 vol.3.

[35] H. Zhang, H. Chen, G. Jiang, X. Meng, and K. Yoshihira, "Fast statistical relationship discovery in massive monitoring data," in *IEEE INFOCOM Workshops 2008*, 2008, pp. 1–6.

[36] S. Sozuer, C. Etemoglu, and E. Zeydan, "A new approach for clustering alarm sequences in mobile operators," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 1055–1060.

[37] M. W. Asres, M. A. Mengistu, P. Castrogiovanni, L. Bottaccioli, E. Macii, E. Patti, and A. Acquaviva, "Supporting telecommunication alarm management system with trouble ticket prediction," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1459–1469, 2021.

[38] J. Wang, C. He, Y. Liu, G. Tian, I. Peng, J. Xing, X. Ruan, H. Xie, and F. L. Wang, "Efficient alarm behavior analytics for telecom networks," *Information Sciences*, vol. 402, pp. 1–14, 2017.

[39] P. P. C. Lee, V. Misra, and D. Rubenstein, "Toward optimal network fault correction via end-to-end inference," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, 2007, pp. 1343–1351.

[40] M. Steinder and A. Sethi, "Increasing robustness of fault localization through analysis of lost, spurious, and positive symptoms," in *IEEE INFOCOM 2002 - Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, 2002, pp. 322–331 vol.1.

[41] G. Jakobson and M. Weissman, "Alarm correlation," *IEEE Network*, vol. 7, no. 6, pp. 52–59, 1993.

[42] D. Veitch, B. Augustin, R. Teixeira, and T. Friedman, "Failure control in multipath route tracing," in *IEEE INFOCOM 2009*, 2009, pp. 1395–1403.

[43] M. Moulay, R. G. Leiva, P. J. R. Maroni, J. Lazaro, V. Mancuso, and A. F. Anta, "A novel methodology for the automated detection and classification of networking anomalies," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 780–786.

[44] K. Zhang, M. Kalander, M. Zhou, X. Zhang, and J. Ye, "An influence-based approach for root cause alarm discovery in telecom networks," in *Service-Oriented Computing - ICSOC 2020 Workshops*, 2021, pp. 124–136.

[45] J. Ji, F. Zhu, J. Cui, H. Zhao, and B. Yang, "A dual-system method for intelligent fault localization in communication networks," in *IEEE International Conference on Communications (ICC)*, 2022.

[46] M. Liu, X. Qi, L. Liu, and H. Pan, "Roots-tracing of communication network alarm: A real-time processing framework," *Computer Networks*, no. 4, p. 108037, 2021.

[47] S. M. Srinivasan, T. Truong-Huu, and M. Gurusamy, "Machine learning-based link fault identification and localization in complex networks," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6556–6566, 2019.

[48] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[49] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[50] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, pp. 3844–3852, 2016.

[51] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2018.

[52] T. R. Besold, A. d. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kühnberger, L. C. Lamb, D. Lowd, P. M. V. Lima *et al.*, "Neural-symbolic learning and reasoning: A survey and interpretation," *arXiv preprint arXiv:1711.03902*, 2017.

[53] A. S. d. Garcez, K. Broda, D. M. Gabbay *et al.*, *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media, 2002.

[54] N. Kaur, G. Kunapuli, T. Khot, K. Kersting, W. Cohen, and S. Natarajan, "Relational restricted boltzmann machines: A probabilistic logic learning approach," in *International Conference on Inductive Logic Programming*. Springer, 2017, pp. 94–111.

[55] G. Sourek, V. Aschenbrenner, F. Zelezny, S. Schockaert, and O. Kuzelka, "Lifted relational neural networks: Efficient learning of latent relational structures," *Journal of Artificial Intelligence Research*, vol. 62, pp. 69–100, 2018.

[56] A. S. A. Garcez and G. Zaverucha, "The connectionist inductive learning and logic programming system," *Applied Intelligence*, vol. 11, no. 1, pp. 59–77, 1999.

[57] P. Dragone, S. Teso, and A. Passerini, "Neuro-symbolic constraint programming for structured prediction," *arXiv preprint arXiv:2103.17232*, 2021.

[58] D. Yu, B. Yang, D. Liu, and H. Wang, "A survey on neural-symbolic systems," *arXiv preprint arXiv:2111.08164*, 2021.

[59] R. Jin, B. Wang, W. Wei, X. Zhang, X. Chen, Y. Bar-Shalom, and P. Willett, "Detecting node failures in mobile wireless networks: A probabilistic approach," *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1647–1660, 2016.

[60] A. Gómez-Andrades, R. Barco, P. Muñoz, and I. Serrano, "Data analytics for diagnosing the rf condition in self-organizing networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 6, pp. 1587–1600, 2017.

[61] M. Maity, B. Raman, M. Vutukuru, A. Chaurasia, and R. Srivastava, "Witals: Ap-centric health diagnosis of wifi networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 4, pp. 975–989, 2018.

[62] S. Gonzalez-Diaz, A. Garcia-Saavedra, A. de la Oliva, X. Costa-Perez, R. Gazda, A. Mourad, T. Deiss, J. Mangues-Bafalluy, P. Iovanna, S. Stracca, and P. Leithead, "Integrating fronthaul and backhaul networks: Transport challenges and feasibility results," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 533–549, 2021.

[63] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[64] A. Darwiche, "On the tractable counting of theory models and its application to truth maintenance and belief revision," *Journal of Applied Non-Classical Logics*, vol. 11, no. 1-2, pp. 11–34, 2001.

[65] A. Darwiche and P. Marquis, "A knowledge compilation map," *Journal of Artificial Intelligence Research*, vol. 17, pp. 229–264, 2002.

[66] A. Darwiche, "Decomposable negation normal form," *Journal of the ACM (JACM)*, vol. 48, no. 4, pp. 608–647, 2001.

[67] ——, "New advances in compiling cnf to decomposable negation normal form," in *Proceedings of the 16th European Conference on Artificial Intelligence*, ser. ECAI'04. NLD: IOS Press, 2004, p. 318–322.

Haihong Zhao received his B.E. degree in Software Engineering from The College of Software, Jilin University, Changchun, China, in 2019 and his M.sc degree in Computer Science from The School of Artificial Intelligence, Jilin University, Changchun, China, in 2022. He is currently a fall 2023 Ph.D. student with The Hong Kong University of Science and Technology (Guangzhou). His research interests include graph learning and data mining.

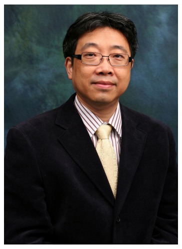
Bo Yang is currently the Director of the Key Laboratory of Symbolic Computation and Knowledge Engineering of the Ministry of Education, Jilin University, Changchun, China. His research interests include data mining, machine learning, knowledge engineering, and complex/social network modeling and analysis. He has published more than 120 articles on international journals, including IEEE TPAMI, IEEE TKDE, IEEE TNNLS, IEEE TCYB, ACM TKDD, ACM TWEB, DKE, JAAMAS, and KBS, and international conferences, including ICLR, NeurIPS, IJCAI, AAAI, WWW, ICDM, and COLING.

Jiaxu Cui received B.E. degree in software engineering from Jilin University, in 2013, and the Ph.D. degree in computer science from Jilin University, in 2021. He is currently a postdoc with the Key Laboratory of Symbolic Computation and Knowledge Engineer of Ministry of Education, Jilin University, Changchun, China. His research interests include Bayesian Optimization, Network Representation Learning, and Causal Inference. He has published several papers in high-impact journals and top conferences, including IEEE TNNLS, ESWA, AAAI, etc.

Qianli Xing is currently a postdoctoral research fellow in the College of Computer Science and Technology, Jilin University, Changchun, China. He received B.E. and M.sc degree in computer science from Jilin University in 2014 and 2017. And he received his Ph.D. degree in computer science from Macquarie University, Australia in 2022. His main research interest is quality control in crowdsourcing, data mining and graph learning. He has published several paper in high-impact journals and top conferences, including IEEE TNNLS, ICMD, ICWS, etc.

Jiaxing Shen is an Assistant Professor with the Department of Computing and Decision Sciences at Lingnan University. He received the B.E. degree in Software Engineering from Jilin University in 2014, and the Ph.D. degree in Computer Science from the Hong Kong Polytechnic University in 2019. He was a visiting scholar at the Media Lab, Massachusetts Institute of Technology in 2017. His research theme is Human Dynamics which aims to understand human behavior and provide actionable insights via cross-disciplinary approaches. Under the theme, his research interests include mobile computing, data mining, and IoT systems. His research has been published in top-tier journals such as IEEE TMC, ACM TOIS, ACM IMWUT, and IEEE TKDE. He was awarded conference best paper twice including one from IEEE INFOCOM 2020.

Fujin Zhu is currently a Senior AI Researcher with the ICT P&S General Development Dept. at Huawei Technologies Co., Ltd., Shenzhen, China. He received his PhD degree in Computer Science from the University of Technology Sydney and PhD degree in Management Science from Beijing Institute of Technology, in 2020. His research interests are causal inference, probabilistic machine learning and deep learning.



Jiannong Cao received the M.Sc. and Ph.D. degrees in computer science from Washington State University, Pullman, WA, USA, in 1986 and 1990, respectively. He is currently the Chair Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His current research interests include parallel and distributed computing, mobile computing, and big data analytics. Dr. Cao has served as a member of the Editorial Boards of several international journals, a Reviewer for international journals/conference proceedings, and also as an Organizing/ Program Committee member for many international conferences.

Appendix A Method Setting

In our paper, GNNs were implemented based on the library *Deep Graph Library* [63]. The concrete method settings are shown by Table 12.

TABLE 12
The Concrete Method Settings in baselines

Method	Setting
Rule-based	The rule-based method is designed by expert knowledge
SVM	SVM cannot be trained by the graph data, so we extract 3+**N** features to generate the feature vector according to the fault scenario sequence, where the **N** means the number of the alarm types. The first three features are the number of the network elements, card and port in the sequence, and the rest features record the the number of different alarms. For example, if current 5G network has three alarm types in all fault scenarios, a feature vector can be *[3,4,5,2,0,1]*. This vector represent there are three network elements, four cards, five ports and three alarms in the current fault scenario sequence
RF	Same with the SVM
XGBoost	Same with the SVM
ANN	A multi-layer perceptron is an implementation way of Artificial Neural Network(ANN). The input is the same with SVM, XGBoost, and RF. The output is the prediction result.
ANN-SVM	ANN-SVM is an ensemble learning method based on a stacking technique. In this work, ANN and SVM are selected as base classifiers for stacking [6]. The input of the base classifier is the same with SVM, XGBoost, RF, and ANN. The output is the prediction result.
BN	A Bayesian network (BN) [27] is a directed acyclic graph, where the nodes represent different variables designed by experts, including *Alarm Type*, *Fault Scenario Type*, *Level*. The edge represents the causalities among nodes, including *Alarm Type* to *Level*, *Alarm Type* to *Alarm Type*, and *Level* to *Faule Scenario Type*. Fig. 9 gives a simple instance designed for *Alarm type*.
FsiGCN	The structure of FsiGCN-NO is same with [19]. The improved FsiGCN-NO has two graph convolutional layers. Every hidden layer includes 256 hidden units. Besides, for identifying the fault scenarios, a multi-layer perceptron is applied to compute the outputs of the FsiGCN-NO/FsiGCN-O to get the predictions.
FsiGAT	The structure of FsiGAT-NO is same with [24]. The improved FsiGAT-NO has two attention layers. Each layer has eight attention heads. Like the implementation of FsiGCN-NO/FsiGCN-O, for identifying the fault scenarios, a multi-layer perceptron is applied to compute the outputs of the FsiGAT-NO/FsiGAT-O to get the predictions.

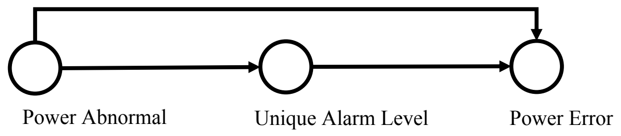


Fig. 9. A simple Bayesian network instance designed for fault scenario *Power Error*, whose level is *Unique Alarm*.

Appendix B Related Knowledge of Fault scenario

Tabel. 13 further gives some other representative alarm types' descriptions. Besides, Tabel. 14 introduces some fault scenario type and their corresponding difficulty levels.

Appendix C Encoder: Training data and Design

Preliminaries: A formula that is a conjunction of clauses (a disjunction of literals) is in Conjunctive Normal Form (CNF). Let S be the set of propositional variables. A sentence in Negation Normal Form (NNF) is defined as a rooted directed acyclic graph (DAG) where each leaf node is labeled with True, False, $s, or \neg s, s \in S$; and each internal node is labeled with $\wedge$ or $\vee$ and can have discretionarily many children. Deterministic Decomposable Negation Normal Form (d-DNNF) [64], [65] further imposes that the representation is: (i) **Deterministic**: An NNF is deterministic if the operands of $\vee$ in all well-formed boolean formula in NNF are mutually inconsistent; (ii) **Decomposable**: An NNF is decomposable if the operands of $\wedge$ in all well-formed boolean formula in the NNF are expressed on a mutually disjoint set of variables. Opposite to CNF and more general forms, d-DNNF has many satisfactory tractability properties (e.g., polytime satisfiability and polytime model counting). Because of having tractability properties, it is appealing for complex AI applications to adopt d-DNNF [66].

The training input of the $Encoder(\cdot)$ is the specific graphs constructed by the formulae extracted from different fault scenario graph, which can be referred to the second output of Algorithm 1. To construct the specific graphs based on these formulae, we first change the formulae in Conjunctive Normal Form (CNF) and then use **c2d** to compile these formulae in Deterministic Decomposable Negation Normal Form (d-DNNF) [64], [65], [67]. For example, based on Formula. (1) in Section 3.1, we construct a concise expression by Formula. (11). m, n represent the relation propositions, $Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ N\ Login)$ and $Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ COM)$, separately. q represents the happen proposition, $Happen\ (Network\ Element\ Out\ Of\ Service)$. Both the equations are in CNF. Then, after executing **c2d**, Formula. cnfexample can be expressed in d-DNNF shown by Formula. (12).

$$(m \wedge n) \implies q \tag{11}$$

$$(\neg_m \wedge n) \vee \neg_n \vee q \tag{12}$$

Then a logical formula can be represented as a directed or undirected graph $G = (V, E)$ with N nodes, $v_i \in V$, and edges $(v_i; v_j) \in E$. Individual nodes are either propositions (leaf nodes) or logical operators ($\wedge$; $\vee$; $\implies$), where propositions are connected to their respective operators. Fig. 10 can help understand the concrete structure. In addition to the mentioned nodes, every graph, like Fig. 10, is further augmented by a global node linked to all other nodes. In $Encoder(\cdot)$, the graphs are regarded as undirected graphs.

TABLE 13
Alarm Type Definition Examples

Alarm Type	Reporting Position	Related Fault Scenario
N N Login	Network Element	1. Network Element Out Of Service 2. Inside Network Element Log Out 3. Equipment Temperature High 4. . . .
E LOS	Port	1. Port Double No Light 2. Port Self No Light 3. Network Element Out Of Service 4. . . .
CLK Abnormal	Network Element	1. Equipment Temperature High 2. Light Module Error 3. . . .
TEMP Abnormal	Card	1. Equipment Temperature High
N COM	Network Element	1. Other 2. Network Element Out Of Service
SUR Temp Abnormal	Network Element	1. Surrounding Temperature High
PW Abnormal	Network Element	1. Power Error
Hard Abnorma	Card	1. Card Error
...	...	...

TABLE 14
Fault Scenario Type Examples and the difficulty level of identifying fault scenario by experts

Fault Scenario	Level	Recognition Complexity
Equipment Temperature High	Unique Alarm	Low
Light Module Error	Unique Alarm	Low
Surrounding Temperature High	Unique Alarm	Low
Power Error	Unique Alarm	Low
Card Error	Unique Alarm	Low
Port Self No Light	Complex Combination	Low
Port Double No Light	Fixed Combination	Middle
Network Element Out Of Service	Complex Combination	High
Inside Network Element Log Out	Complex Combination	High
Chain Network Element Log Out	Complex Combination	High
. . .	. . .	. . .

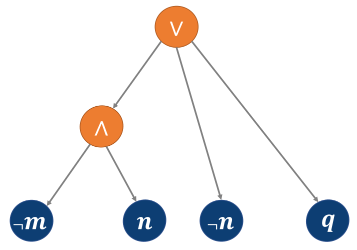


Fig. 10. The d-DNNF graph structure generated based on Formula. (12): **m** represents the relation propositions, $Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ N\ Login)$; **n** represents $Ne\ Next\ To\ Ne\ (N\ N\ Login,\ N\ COM)$, separately; **q** represents the happen proposition, $Happen\ (Network\ Element\ Out\ Of\ Service)$

The layer-wise propagation rule of GCN is,

$$Z^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}Z^{(l)}W^{(l)}) \tag{13}$$

where $Z^{(l+1)}$ represent the learnt latent node embeddings at l^{th} (note that $Z^{(0)} = X$), $\tilde{A} = A + I_N$ represents the adjacency matrix of the undirected graph G with added self-connections through the identity matrix I_N. $\tilde{D}$ is a diagonal degree matrix with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. The weight matrices for layer-specific training are $W^{(l)}$, and $\sigma(\cdot)$ represents the activation function. For better acquiring the semantics associated using the graphs, $Encoder(\cdot)$ furthurly adopts two modifications: heterogeneous node embeddings and semantic regularization [17]. The concrete code can be found in https://github.com/ZiweiXU/LENSR.

Appendix D
Explanation for the difficulty of labeling datasets

The reasons why the process of labeling datasets is challenging are stated as follows:

1) 5G networks are more complex and heterogeneous than the previous networks, so the fault scenario graph cannot be labeled by simple rules. For example, the combinations of alarm types and alarm locations in topology have large amounts of variants. Some similar fault scenario instances belong to different fault scenario types, and some different fault scenario instances belong to same types. This situation requires us to learn and capture considerable expert knowledge to clearly analyze the fault scenario graphs to determine the fault scenario type of different graphs.
2) The relations among alarms can also influence the process of labeling. For example, some alarms in a fault scenario graphs are the key alarms, which imply the cause of the fault and generate the other correlative alarms. Therefore, when we identify the

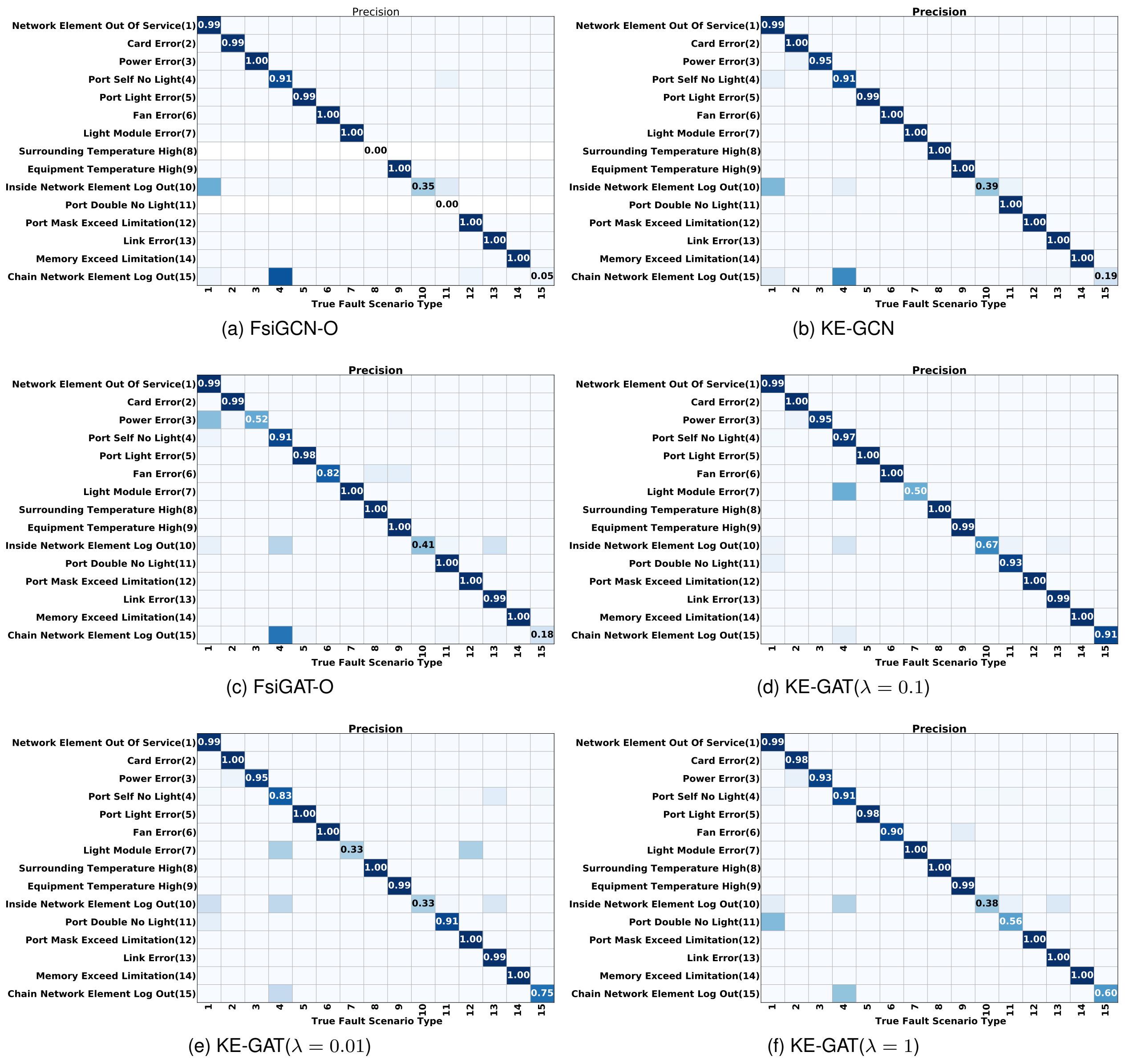


Fig. 11. Confusion matrix comparison for FsiGAT-O, FsiGCN-O, KE-GCN and KE-GAT assigned three different λ values on FSD1-15.

fault scenario type, we have to decide which alarms are the key alarms.

3) Some fault scenario types need extra information, such as *In Chain* information mentioned in Section 2.2.2, to be further identified.

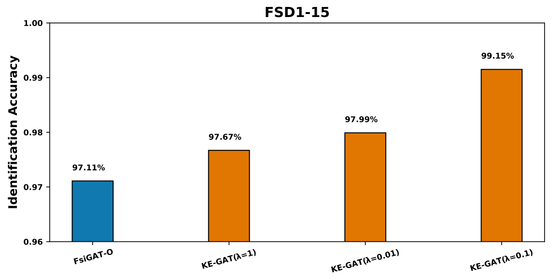


Fig. 12. Accuracy comparison among FsiGAT-O and KE-GAT assigned different λ values on the FSD1-15.

Appendix E
Related Experimental Results

Fig. 12 shows the accuracy of FsiGAT-O and KE-GAT with different λ. Fig. 11 vividly shows the identification ability of FsiGCN-O, FsiGAT-O, KE-GCN, and KE-GAT with different λ. According to the color shapes, we can find the KE-GAT with $\lambda = 0.1$ achieves the best performance.