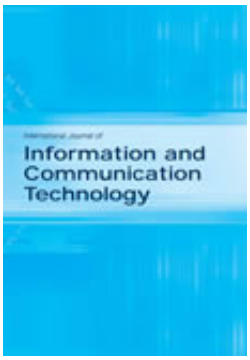


International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

Electric-power telecommunication alarm analysis method based on spatio-temporal graph neural networks

Yuhang Pang, Tongtong Zhang, Qiong Liu, Jiaxing Ren, Meiru Huo, Jianliang Zhang

DOI: 10.1504/IJICT.2025.10070824

Article History:

Received:	06 January 2025
Last revised:	20 February 2025
Accepted:	20 February 2025
Published online:	06 May 2025

Electric-power telecommunication alarm analysis method based on spatio-temporal graph neural networks

Yuhang Pang*, Tongtong Zhang, Qiong Liu and Jiaxing Ren

Information and Communication Department,
China Electric Power Research Institute,
Beijing, 100192, China
Email: 17710660959@163.com
Email: zh_tongt@126.com
Email: 657798691@qq.com
Email: rjx13781378@126.com
*Corresponding author

Meiru Huo and Jianliang Zhang

Department of Technology Development,
State Grid Information and Telecommunication Corporation of SEPC,
Shanxi, Taiyuan, 030021, China
Email: 15333669912@163.com
Email: 15333666875@163.com

Abstract: With the growing complexity of electric-power telecommunication networks, alarm messages have surged, challenging operation and maintenance. Traditional methods overlook spatio-temporal characteristics, hindering root cause identification and fault propagation analysis. This paper proposes a spatio-temporal graph neural network-based alarm analysis method, extracting spatial and temporal features to construct a graph model and node alarm subtrees based on alarm propagation. Subtree features incorporate node relationships and alarm rules for accurate analysis. To reduce computational overhead from model parameters, a swarm intelligence optimisation-based lightweight technique is introduced, enhancing deployment efficiency in resource-constrained environments. Experiments demonstrate superior performance in alarm correlation accuracy, compression, and fault localisation compared to traditional methods. The approach effectively minimises redundant alarms, boosts maintenance efficiency, and supports intelligent network operation.

Keywords: electric-power telecommunication network; spatio-temporal graph neural network; alarm subtree; particle swarm optimisation; PSO.

Reference to this paper should be made as follows: Pang, Y., Zhang, T., Liu, Q., Ren, J., Huo, M. and Zhang, J. (2025) 'Electric-power telecommunication alarm analysis method based on spatio-temporal graph neural networks', *Int. J. Information and Communication Technology*, Vol. 26, No. 10, pp.1–20.

Biographical notes: Yuhang Pang graduated from the Xi'an Jiaotong University in 2021. He has been engaged in scientific research in the field of electric-power communication and its intelligence for four years. He often participates in academic association activities and wins awards. He has published more than ten papers and patents in related fields.

Tongtong Zhang graduated from the University of Chinese Academy of Sciences in 2019. She worked in China Electric Power Research Institute (CEPRI). Her research interests include digitalisation and intelligence of information and communication.

Qiong Liu graduated from the Huazhong University of Science and Technology in 2021. She worked in China Electric Power Research Institute. Her research interests include information and communication engineering.

Jiaxing Ren graduated from the Beijing Technology and Business University in 2018. She worked in China Electric Power Research Institute. Her research interests include computer science and technology.

Meiru Huo graduated from the Shanxi University in 2018. She worked in State Grid Information and Telecommunication Corporation of SEPC. Her research interests include blockchain, quantum, artificial intelligence and other technologies and applications.

Jianliang Zhang graduated from the Jilin University in 2004. He worked in State Grid Information and Telecommunication Corporation of SEPC. His research interests include power information communication.

1 Introduction

In modern power systems, electric-power telecommunication networks (EPTN) play a crucial role in ensuring real-time communication and control signal exchange between various devices. However, as power systems become increasingly complex, the scale and diversity of equipment in EPTN are rapidly growing, leading to a rise in the frequency and complexity of network faults. Fault diagnosis is a core component in ensuring the safe and stable operation of power systems. By timely and accurately identifying potential faults in the network, it can effectively avoid or mitigate the impact of accidents, improving system stability and reliability (Liao et al., 2021). Traditional fault diagnosis methods, typically based on rules and statistical analysis, rely on expert experience and struggle to handle the complex spatio-temporal characteristics of EPTN alarm events. Moreover, traditional methods perform poorly when dealing with large-scale data and nonlinear features, failing to meet the real-time and accuracy requirements for fault diagnosis in smart grids (Jiang et al., 2022).

In recent years, with the rapid advancement of big data and artificial intelligence technologies, intelligent fault diagnosis methods have emerged as a prominent area of research. Specifically, deep learning-based approaches for fault diagnosis, by leveraging pattern recognition and feature extraction from alarm data in EPTN, have facilitated the intelligent prediction and diagnosis of fault events. The alarm information in EPTN reflects abnormal changes and trends in the network's operating status, exhibiting strong temporal and spatial correlations (Rivas and Abrao, 2020). The analysis of alarm

information not only helps identify the current fault state but also predicts potential fault risks. Related studies have shown that alarm analysis based on spatio-temporal characteristics can improve the accuracy and efficiency of fault diagnosis. For example, spatio-temporal graph neural networks (STGNNs), by modelling on graph structures, can simultaneously capture the temporal evolution and spatial correlations of alarm events. This has become one of the cutting-edge research directions (Yang et al., 2024; Jamshidiha et al., 2024). However, the complexity of STGNNs is closely related to the network scale, and as the network size increases, the model becomes more bloated and complex, leading to a sharp increase in the required computational resources.

Intelligent alarm analysis is a significant breakthrough in fault diagnosis for EPTN. Compared to traditional alarm filtering and threshold detection methods, deep learning-based alarm analysis methods can dynamically and automatically identify multiple alarm patterns, enabling real-time processing of large-scale alarm data. During the alarm analysis process, utilising graph neural networks (GNNs) to handle the graph structural characteristics of EPTN allows for the exploration of complex dependencies between nodes in the network (Hussain et al., 2019). By integrating long short-term memory (LSTM) networks or temporal convolutional networks (TCN), the temporal dynamics of alarm event sequences can be efficiently captured and analysed (Jiang and Bai, 2023). Fault diagnosis methods based on STGNNs not only improve the recognition and classification ability of alarm events but also enhance the accuracy of root cause analysis, providing new insights for the intelligent management and optimisation of EPTN (Wang et al., 2019).

Based on the above, this paper proposes an intelligent fault diagnosis method based on STGNNs for alarm data analysis in electric-power telecommunication network fault diagnosis. First, statistical methods are used to extract the spatial and temporal features of the electric-power telecommunication network, and a spatio-temporal graph model is constructed using graph convolutional neural networks. Based on this model, alarm rules are integrated with alarm features and mapped into a fault tree. By utilising the spatial and temporal dependencies of alarm events, precise fault event localisation and prediction are achieved. Finally, to address the issue of excessive computational resource consumption during model training and optimisation, a model lightweight technique is proposed, which improves the efficiency of model deployment.

2 Spatio-temporal graph neural network model for EPTN

Since traditional CNNs cannot handle data from non-Euclidean spaces, researchers have defined a neural network structure capable of processing graph-domain data: GNNs. The concept of GNNs was introduced in the 2005 literature (Gori et al., 2005), and in 2013, the concept of convolution was introduced into GNNs. A graph convolutional network model was proposed based on frequency-domain convolution, applying learnable convolution operations to graph-domain data (Bruna et al., 2013).

2.1 Spatial feature extraction based on graph convolution

The topology of the electric-power telecommunication network can be represented as a graph, where the normalised Laplacian matrix serves as a tool in the frequency domain to characterise the structural properties of the power system's topology:

$$L_{sys} = I - D^{-\frac{1}{2}} Y D^{-\frac{1}{2}} \tag{1}$$

where Y is used as the weighted adjacency matrix A of the electric-power telecommunication system, where weight can be selected as line admittance; I represents the identity matrix; and the meaning of D is the weighted degree matrix of the graph, with its elements $D_{ii} = \sum_j A_{ij}$, where A_{ij} represents the elements of the matrix A.

Perform spectral decomposition of the Laplacian matrix to further extract features:

$$L_{sys} = U \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} U^T \tag{2}$$

$U = [u_1, \ldots, u_n]$ is a column vector and an orthogonal matrix of unit eigenvectors, satisfying $UU^T = I$; n is the total number of eigenvalues; λ_i represents the eigenvalues of the matrix.

Drawing a parallel with the conventional Fourier transform, the graph Fourier transform can be represented in matrix form as follows:

$$\hat{X}_g = U^T X_g \tag{3}$$

where $\hat{X}_g$ represents the eigenvectors of the graph, and $U^T X_g$ denotes the graph Fourier transform matrix.

By drawing an analogy with traditional convolution operations, the graph convolution formula can be formulated as follows:

$$\left(X_g * g\right)_G = U\left(\left(U^T g\right) \odot \left(U^T X_g\right)\right) \tag{4}$$

where $\odot$ denotes the Hadamard product; * represents graph convolution; g is the convolution kernel and $(X_g * g)_G$ indicates the application of graph convolution on graph G. By defining $g_\theta = U^T g$ as a learnable convolution kernel, the graph convolution can be expressed as:

$$\left(X_g * g\right)_G = U g_\theta U^T X_g \tag{5}$$

The computation in the above formula is intensive, leading to slower processing speeds. Chebyshev polynomials can be used to approximate the convolution kernel (Defferrard et al., 2016) to reduce computational complexity:

$$g_\theta = g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k\left(\tilde{\Lambda}\right) \tag{6}$$

where Λ represents the diagonal matrix composed of the eigenvalues of the matrix L_{sys}, $\tilde{\Lambda} = 2\Lambda / \lambda_{\max} - I$ denotes the scaled eigenvector matrix, T_k is the k^{th} order Chebyshev polynomial, θ_k corresponds to the Chebyshev coefficients, and K is the number of terms in the Chebyshev polynomial.

Thus, the convolution formula can be expressed as follows:

$$g_\theta * x = \sum_{k=0}^{K-1} \theta_k T_k\left(\tilde{L}\right) x \tag{7}$$

where $\tilde{L} = 2L_{sys} / -I = U\tilde{\Lambda}U^T$.

2.2 Time feature extraction based on gated convolution

Recurrent neural network (RNN)-based models have been extensively utilised in time series analysis tasks. However, RNNs and their derivatives, such as LSTM networks and BiLSTM networks, still suffer from issues like serial iteration time consumption and complex gating mechanisms. Inspired by the work in reference (Zhang et al., 2018), this paper utilises gated CNNs along the time axis to capture temporal dynamic features more quickly, and employs residual connections (He et al., 2016) to mitigate the overfitting problem.

In addition, we employed strategies such as gradient clipping and adaptive learning rate to further enhance the stability of the training process. During training, dynamic batch normalisation was used to ensure that each layer of the network could effectively learn, preventing issues such as gradient explosion or vanishing gradients.

The time convolution layer is structured to include a one-dimensional causal convolution, and the gated linear unit (GLU) is set to linear activation, allowing for effective feature extraction from time series data of varying lengths. In this layer, the input at each node consists of a sequence of length M with multiple input channels C_i, denoted as $X \in R^{M \times C_i}$. Therefore, the time convolution layer can be defined as:

$$\Gamma * X = P \odot \sigma(Q) \in R^{(M-K_t+1)\times C_o} \tag{8}$$

where P and Q are the inputs to the gates in the GLU; K_t is the size of the time convolution kernel; C_o represents the number of output channels; and $\sigma(\cdots)$ denotes the activation function, which, in this study, is the sigmoid activation function.

To address the diversity of alarm responses across time scales in power communication networks, we introduced a multi-scale convolutional structure in the temporal convolution module. This structure aims to simultaneously capture the features of both short-term and long-term alarm sequences, in order to handle alarm responses ranging from millisecond-level reactions to cumulative effects that last for hours or even days.

The core idea of the multi-scale convolutional structure is to use convolutional kernels of different sizes to separately capture the temporal features of short-term (millisecond to second scale) and long-term (minute to hour scale) alarm sequences. Let the time-series data $x(t)$ represent the alarm data at time t. In traditional convolution operations, the convolution is performed between the kernel w_i and the input data:

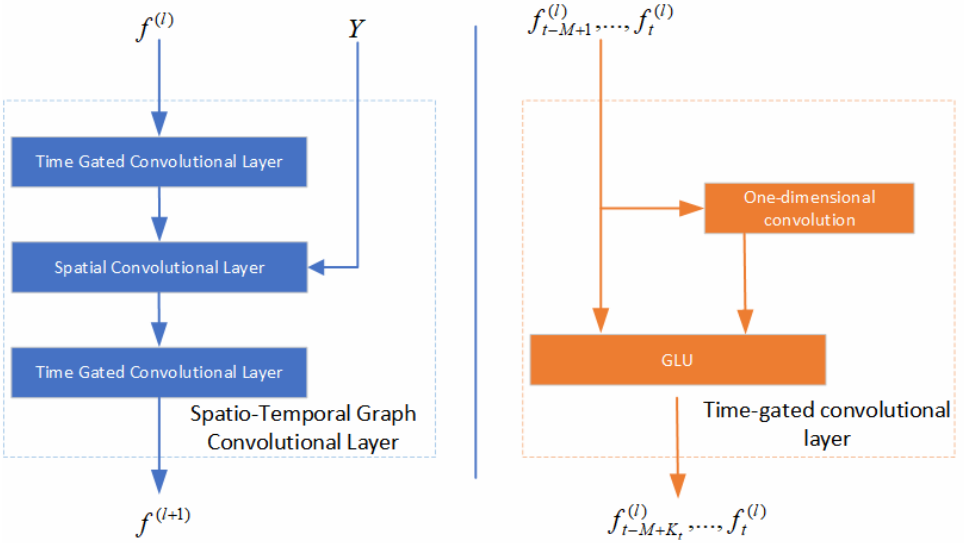
$$y(t) = \sum_{i=1}^{k} w_i \times x(t-i)$$

where k represents the size of the convolutional kernel, and w_i denotes the weights of the kernel. By using kernels of different sizes, we can extract features at different scales in each layer. For instance, small-sized kernels can capture short-term changes, such as millisecond-level alarm responses, while larger kernels can effectively capture long-term variations, such as alarm accumulation effects over hours or days. The feature maps output by each convolutional kernel are processed by an activation function, followed by concatenation or weighted summation, to form a multi-scale fused feature representation.

2.3 Spatio-temporal feature extraction using spatio-temporal graph convolutional layers

With the purpose to simultaneously capture spatial and temporal features, combining the spatial map convolution layer with the time gated convolution layer, the space-time map convolution block (ST Conv) is designed. The spatio-temporal graph convolutional layer is structured by placing a spatial layer between two temporal layers, enabling effective spatial information propagation across consecutive temporal convolutions (Yu et al., 2017). The architecture of the spatio-temporal graph convolutional layer is depicted in Figure 1.

Figure 1 Spatio-temporal convolutional layer architecture diagram (see online version for colours)



In the spatio-temporal graph convolutional layer, the temporal layers receive the alarm information flow f as input, while the spatial layer is fed with the admittance matrix Y. The alarm information flow f first undergoes temporal convolution to capture temporal features. Subsequently, the temporal features are combined with the admittance matrix Y and passed through the spatial convolution layer to extract spatial features. After activation, the information flows through another temporal convolution layer, ultimately producing the extracted spatio-temporal features.

Based on the graph convolution and temporal convolution layers, the information flow $f(l) \in R^{M\times n\times C^l}$ entering the spatio-temporal graph convolution block l (where C^l is the number of channels in the spatio-temporal graph convolution block) can be computed as the output information flow $f(l+1)$ as follows:

$$f^{(l+1)} = \Gamma_1^{(l)} * \sigma\left(\Theta_G^{(l)} * \left(\Gamma_0^{(l)} * f^{(l)}\right)\right) \tag{9}$$

In the equation: $\Gamma_0^{(l)}, \Gamma_1^{(l)}$ represents the temporal convolution kernels from the upper and lower layers within the spatio-temporal graph convolution block l. These features may

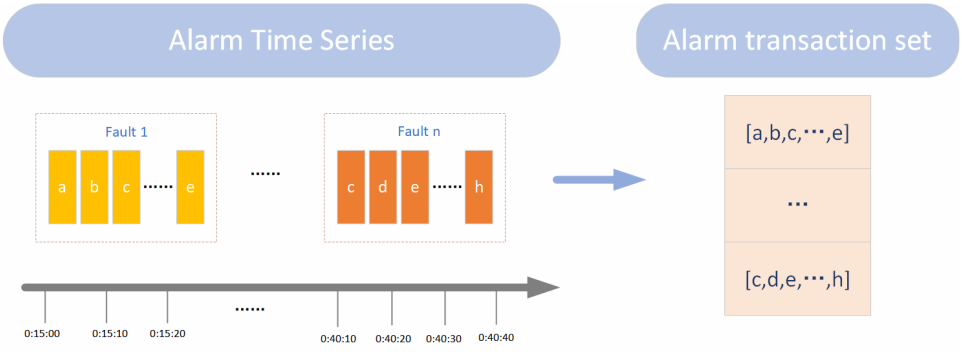
need further processing based on the specific application in order to produce the final results.

3 Alarm correlation analysis based on PrefixSpan

3.1 DBSCAN clustering for extracting alarm transaction sets

When a network fault occurs, it triggers a series of alarms. These alarms exist as individual entities. In order to perform association rule mining to uncover the intrinsic relationships between alarms, it is necessary to aggregate all the alarm time series triggered by a particular fault into a single transaction set, ensuring their correlation as much as possible, as shown in the figure. The first step in correlation analysis is to transform the alarm data into an alarm transaction database.

Figure 2 Alarm transaction sequence and alarm transaction set (see online version for colours)



The DBSCAN clustering algorithm is primarily used for alarm transaction set extraction because the clusters formed by this algorithm represent the largest set of density-connected points (Ma et al., 2022). When clustering alarm event times, it maximises the temporal correlation by grouping events with the highest temporal correlation into the same cluster, thereby enhancing the temporal relevance of the alarm events. When using this algorithm, the final clustering result is mainly obtained by adjusting the parameters eps and MinPts. The combination of different values for these two parameters will have an impact on the clustering outcome, making it somewhat complex to determine the optimal clustering parameters. The basic clustering algorithm involves manually tuning the parameters eps and MinPts one by one, which makes the algorithm cumbersome and limits its efficiency. The application of a visual parameter tuning algorithm in this chapter helps avoid the repetitive manual adjustment of these two parameters. The key idea of this algorithm is to propose an optimal intermediate score and automatically cycle through a range of values for both parameters (Nguyen et al., 2023). The algorithm then evaluates the clustering performance and compares it with the intermediate score to ultimately determine the optimal parameter combination that produces the best clustering result (Wang et al., 2019). The central idea of the visual parameter tuning algorithm is to jointly adjust the parameters eps and MinPts to determine the optimal combination of parameters that produces the best clustering result, as shown in the following algorithm:

Table 1 Visual parameter tuning algorithm

Algorithm:	
Input: eps = [eps_s, eps_t]: the range of eps	
	MinPts = [mp_s, mp_t]: the range of MinPts
	Δ_i: The adjustment step size for eps and MinPts
	best_core: Silhouette coefficient
1:	for eps eps_i and MinPts mp_j of all values in (eps, MinPts: Δ_i) do
2:	timedis = d(t_i, t_j)
3:	score = score(eps_i, mp_j)
4:	if score > best_score then
5:	best_core = core
6:	end if
7:	$eps_i = eps_i + \Delta_1$, $mp_j = mp_j + \Delta_2$
8:	end for

In DBSCAN, the distance metric needs to be considered, and here the Manhattan distance is used, as shown in the following formula:

$$d\left(a_i, a_j\right) = \sum_{i=1}^{n} \left|t_i - t_j\right| \tag{10}$$

where t_i and t_j represent the times at which alarms a_i and a_j occur, respectively.

The most common evaluation metric in clustering algorithms is the silhouette coefficient. The closer the result is to 1, the stronger the correlation of the clusters obtained from the alarm samples. For a sample, the silhouette coefficient is defined as follows:

$$SC(i) = \frac{SCb(i) - SCa(i)}{\max(SCa(i), SCb(i))} \tag{11}$$

In the data pre-processing stage, handling missing values and noisy features is crucial for improving the stability and performance of the model. If missing values and noisy data are not properly addressed, they can significantly impact subsequent analysis results, leading to a decline in model performance or instability. To this end, we propose effective methods for handling missing data and noisy features in this study, ensuring data integrity and quality, and thereby enhancing the model's accuracy and reliability.

1 Mean imputation

To ensure data integrity and minimise the impact of missing data on the analysis process, we employ the mean imputation method to fill in missing values. Let the dataset be denoted as $D = \{(x_1, y_1), (x_2, y_2), \ldots\ldots, (x_n, y_n)\}$, where x_i represents the feature values and y_i represents the label values. If a feature value is missing ($x_j = NaN$), the mean value of that feature $\overline{x_1}$ is used for imputation:

$$\overline{x_J} = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \tag{12}$$

2 Feature selection algorithm

Noisy data is often a key factor in the instability of clustering algorithm performance, particularly in high-dimensional data, where noisy features can dilute effective signals, affecting the model's convergence and accuracy. To mitigate the impact of noisy features, we use the Chi-square test as a feature selection method to filter the original feature set. To address the issue of redundant features, we performed a feature correlation analysis and applied principal component analysis (PCA) for dimensionality reduction. We removed highly correlated redundant features and then performed clustering on the reduced data, comparing the clustering results before and after dimensionality reduction. The results show that after removing redundant features, the clustering performance improved, and computational efficiency was also enhanced.

Thus, this model can perform clustering analysis on the information flows extracted in the previous chapter. By inputting the actual dataset into the model, it is evident that, despite data pre-processing, the alarm data volume remains substantial. With such a large volume of data, the convergence time of the clustering algorithm is lengthy and time-consuming. DBSCAN (Nguyen et al., 2023) is a density-based clustering algorithm suitable for handling noisy data and irregularly shaped clusters, requiring multiple calculations of distances between points to determine the neighbourhood of a point. If the data volume is large or the dimensionality is high, directly calculating pairwise distances for all points significantly reduces algorithm efficiency. Therefore, we improved the DBSCAN clustering algorithm by incorporating a KD-Tree (Zhao et al., 2023) to reduce runtime. KD-Tree is an efficient spatial partitioning data structure that significantly optimises neighbourhood query speed.

The algorithm is shown in Table 2.

Table 2 KD_Tree construction algorithm

KD_Tree Algorithm:	
Input: data: Alarm start time	
	f: Segmentation features
	x: Partition point
1:	generate root node *t*
2:	for $f \leftarrow f(Var_{max})$ do /* Choose the feature with the highest variance */
3:	$x \leftarrow f(Med)$ /* Take the feature's median as the segmentation point */
4:	if $x[cd] <$ t.data[*cd*] then
5:	*t.left* = insert(*x*, *t.left*, (*cd* + 1)) /* Assign samples with feature values less than the median to the left subtree */
6:	else
7:	*t.right* = insert(*x*, *t.right*, (*cd* + 1))
8:	end if
9:	*KD_Tree* ← t
10:	end for

3.2 Generate alarm rules with the PrefixSpan algorithm

In the previous section, the DBSCAN clustering algorithm divides each cluster, based on alarm occurrence times, as a transaction. All the clusters form a transaction set database. After generating the transaction set database, the PrefixSpan algorithm can be used to mine the temporal and spatial correlations between alarm events, discovering the sequential relationships and the frequency of strong association rules between different alarm events (Liao et al., 2020). This provides a foundation for locating the root alarm position based on network element information.

When performing PrefixSpan sequence pattern mining, the alarm sequence set from each cluster obtained by the DBSCAN clustering algorithm serves as the input for the PrefixSpan algorithm. By adjusting the support and confidence levels multiple times, the ratio of the number of frequent alarm sequences to the total number of alarms determines the final support and confidence values. Frequent sequence set mining is then carried out. The strong association rules and their occurrence frequencies mined by the PrefixSpan algorithm are organised into rules and converted into the format ($[A \Rightarrow B]$, m), where $[A \Rightarrow B]$ represents the mined strong association rule, and m is the frequency with which the rule occurs.

Statistical analysis of the mined rules and their frequencies reveals that the number of mined alarm strong association rules is still vast (Wu et al., 2023). Since the goal is to explore the correlations between multiple alarm events, 1-alarm strong association rules are excluded from the study. Although the removal of 1-alarm frequent sequences alleviates the pressure on subsequent research of alarm strong association rules, the number of alarm strong association rules remains large. Upon observing the frequencies of these rules, it was found that some strong association rules occur in alarm events several hundred times, while others appear only a few dozen times or even less. To further solidify the strong correlations between alarm events, a threshold is set based on rule frequencies. The frequencies of all strong association rules are statistically analysed, and the average frequency is calculated. Strong association rules with frequencies greater than the average are selected for further study. This reduction in the number of strong association rules accelerates the process of root alarm localisation.

3.3 Root alarm localisation based on fault tree

The fault tree systematically presents the causal relationships and logical dependencies between components in a complex system. By decomposing possible faults into multiple layers and connecting them with logic gates (such as AND, OR, etc.) (Janu et al., 2022), the fault tree can clearly describe the propagation paths and dependencies between alarms. When certain leaf nodes trigger alarms, the fault tree can quickly trace the potential root alarm location based on its structure, avoiding interference from irrelevant alarms. At the same time, the fault tree leverages expert knowledge and historical data for modelling, allowing it to cover common fault scenarios and dynamically adapt to system changes, thus efficiently and accurately locating the root cause of problems in complex environments.

To address the issue of overlapping root cause analysis caused by multiple intersecting propagation paths, we introduced an alarm weight allocation method based on the influence of propagation paths in the model. In this approach, the weight of each propagation path is adjusted according to its contribution to the final alarm result, taking

into account factors such as the path's propagation time, frequency, and overall impact on the system. We also designed merging rules to ensure that when different propagation paths intersect, the information can be integrated in a reasonable manner, thereby avoiding conflicts in root cause analysis.

First, an influence metric is calculated for each alarm propagation path. This metric combines the path's propagation time, (i.e., the time delay for an alarm to travel from the source node to the current node), the frequency of alarm events, (i.e., the occurrence frequency of alarms along the path), and the path's impact on the overall system (i.e., the importance of the path within the system). These factors can be quantified through a weighted summation, resulting in the weight of each path. Let the influence of path $P(i)$ be denoted as $WP(i)$, then the following formula applies:

$$W(P_i) = \alpha \times T(P_i) + \beta \times F(P_i) + \gamma \times I(P_i)$$

where $T(P_i)$ is the propagation time of the path, $F(P_i)$ is the frequency of alarm events along the path, and $I(P_i)$ is the metric representing the path's overall impact on the system. Through this weight allocation method, alarm propagation paths with greater influence will be assigned higher weights, while paths with less influence will receive lower weights.

Assuming that multiple propagation paths P_1, P_2, …, P_n intersect at the same node, we can perform a weighted fusion of the alarm information from each path based on their respective weights, resulting in the merged alarm severity S:

$$S = \frac{\sum_{i=1}^{n} W(P_i) \cdot S_i}{\sum_{i=1}^{n} W(P_i)}$$

With this weighted merging strategy, we ensure that the information from multiple propagation paths is effectively integrated, while minimising the root cause analysis conflicts caused by path intersections.

First, extract three features from the alarm data – alarm impact on business, alarm impact on equipment, and alarm severity level – to represent the alarm influence in the fault tree. The strong association rules mined by the sequence pattern mining algorithm in the previous section can form branch paths, thereby generating the fault tree. The alarm severity level is determined based on the alarm impact and priority. For example, when calculating the two rules [11, 12, 14] and [11, 13], where 11, 12, 13, and 14 represent the alarm IDs, the feature information contained in the alarm data is as shown in the table.

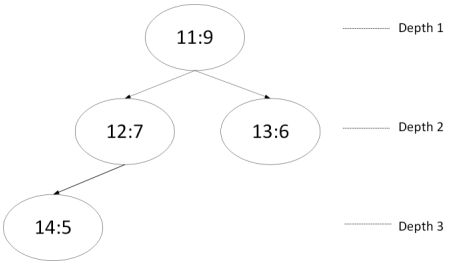
Table 3 Feature information table

Alarm ID	*Alarm location information*	*Impact of alarm on business*	*Impact of alarm on equipment*	*Alarm severity level*
11	36-211-JCDASJF-HW-PTN1	3	2	4
12	36-2-JYLWHJJF-HW-PTN-1	2	2	3
13	36-34-SFXJJF-HW-PTN1	2	1	3
14	22-89-DPJF-HW-PTN1	1	2	2

When generating the fault tree, the nodes are set as A: B, where A represents the alarm node and B represents the primary impact of the alarm. The branches of the fault tree

correspond to the strong association rules. For example, the branch path of Rule 1 [11, 12, 14] is (11 → 12 → 14), and the branch path of Rule 2 [11, 13] is (11 → 13). The branch paths are mapped to the alarm node impacts and are used to construct the tree structure. The fault tree constructed by these two rules is shown in the figure.

Figure 3 Fault tree diagram



By calculating the alarm impact based on alarm features, the primary impact of alarms 11, 12, 13, and 14 is the sum of the three alarm features, which are 9, 7, 6, and 5, respectively. Since alarm 11 triggers the occurrence of alarms 12, 13, and 14, the impact of alarm 11 is recorded as 27. As alarm 11 causes alarms 12, 13, and 14 to occur, the final impact of alarm 11 is the sum of the impacts of its child nodes, that is

$$W = \sum_{i=1}^{n} w_i \tag{13}$$

It can be concluded that the final impact of alarms 11, 12, 13, and 14 are 27, 12, 6, and 5, respectively. In identifying the alarm severity level, the priority of the depth of the fault tree is higher than that of the alarm impact. First, all alarms at depth 1 of the fault tree are sorted in descending order of their final impact. Then, all alarms at depth 2 of the fault tree are sorted in descending order of their final impact, and so on, until completion. The four alarms are ranked by alarm severity as follows: 11, 12, 13, and 14. Based on the tree structure, alarms that are closer to the root node of the fault tree, i.e., those with a smaller fault tree depth and higher impact are closer to the source of the network anomaly. Network administrators process alarms based on priority and use a scoring mechanism to identify the root alarm node. After calculating the scores of all alarms, the most influential alarm is detected as the highest-scoring alarm.

4 Research on model lightweighting based on swarm intelligence

In the scenario of fault diagnosis in EPTN, models based on spatiotemporal GNNs typically have high complexity. To improve model deployment efficiency and adapt to resource-constrained environments (such as embedded devices or edge computing nodes), this paper combines techniques like particle swarm optimisation (PSO) to achieve model lightweighting. The optimisation directions are mainly as follows: selecting important graph nodes or edges, (e.g., redundant alarm association edges) for pruning, and dynamically optimising the quantisation bit-widths of different layers.

4.1 PSO algorithm

PSO (Qian et al., 2023) is an evolutionary-based intelligence algorithm that operates on a population of particles, each of which explores the solution space to identify the global optimum. Throughout the search process, every particle updates its velocity based on both its own previous experience and the information shared by other particles, leading to the collective convergence of all particles towards the optimal solution.

Let the solution space be represented as a D-dimensional space. Denote $X_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]$ as the position vector of the i^{th} particle, $V_i = [v_{i1}, v_{i2}, \ldots, v_{iD}]$ as its velocity vector, $P_i = [p_{i1}, p_{i2}, \ldots, p_{iD}]$ as the best historical position encountered by the iii^{th} particle, and P_{gd} as the global best position discovered by the entire swarm. The position and velocity of each particle in the swarm are then updated according to the following equations:

$$x_{id}(t+1) = x_{id} + v_{id}(t+1) \tag{14}$$

$$v_{id}(t+1) = w \times v_{id}(t) + C_1 \times rand() \times \left(P_{id} - x_{id}\right) + C_1 \times rand() \times \left(P_{gd} - x_{id}\right) \tag{15}$$

In the equation, w denotes the inertia weight, while C_1 and C_2 are acceleration coefficients, and *rand*() represents a random value between 0 and 1. As indicated by equation (14), the particle's position at the next time step is influenced by its current position and velocity. The future velocity is primarily governed by three components: the current velocity, the difference between the current position and the particle's personal best position, and the difference between the current position and the global best position of the swarm.

4.2 Redundancy pruning for model optimisation

The optimisation objective of this model is to maximise the accuracy of the alarm fault diagnosis model while minimising the model's computational complexity and parameter count (Koca and Avci, 2024). In the algorithm, the position of the particle represents whether a node or edge in the graph is retained. A particle value of 0 indicates that the iii^{th} node or edge is deleted, while a value of 1 indicates that it is retained.

The fitness function measures the quality of the pruning scheme, and can be designed by combining model accuracy and pruning rate as follows:

$$f(X) = \alpha \cdot Accuracy(X) - \beta \cdot Pruning\_Rate(X) \tag{16}$$

where *Accuracy*(X) represents the accuracy of the pruned model on the validation set, *Pruning_Rate*(X) represents the reduction ratio of parameters after pruning, and α and β are the weights for accuracy and pruning rate, respectively, used to balance the relationship between the two.

To assess the real-time inference performance of the proposed model in practical applications, we conducted a detailed analysis of the latency and spatio-temporal complexity for each inference path. By comparing the performance of the model before and after pruning, we found that an appropriate pruning strategy can significantly reduce inference time. Let the latency of an inference path in the network be τ_i, and the latency of the path after pruning be τ_i'. We measure the impact of the pruning strategy by calculating the following formula:

$$\Delta\tau_i = \tau_i - \tau_i'$$

In addition to inference latency, we also performed a quantitative analysis of the model's spatio-temporal complexity. By calculating the changes in memory usage, we can effectively assess the impact of pruning on memory consumption, while comparing the model's inference time helps us evaluate the impact of pruning on time complexity.

4.3 Layer-wise quantisation bit-width optimisation model

The purpose of applying the PSO algorithm to deep learning models is primarily to maximise the performance of the quantised model and minimise hardware overhead (Janu et al., 2022). Therefore, the position of each particle represents the quantisation bit-width configuration for each layer, and the fitness function can be designed as follows:

$$f(X) = \alpha \cdot Accuracy(X) - \beta \cdot Hardware\_Cost(X) \tag{17}$$

where $Accuracy(X)$ represents the accuracy of the pruned model on the validation set, $Hardware\_Cost(X)$ represents the hardware overhead, such as model size or inference time, and α and β are the weight factors.

5 Simulation experiments and analysis

5.1 Spatiotemporal graph neural network model case study for EPTN

To validate the effectiveness of the graph neural network model proposed in this paper, a case study is conducted on a 10-machine, 39-bus system, which includes ten generators, 39 buses, and 46 lines.

The model is trained using the RMSprop optimiser (Chen et al., 2022), with an initial learning rate set to 0.001. In GNNs, RMSprop generally performs better in handling gradient updates, and an initial learning rate of 0.001 is a common choice that often yields good results across multiple experiments. The dropout probability is set to 0.1, where a lower dropout rate (0.1) is typically used to avoid overfitting while retaining most of the information. In complex network structures like GNNs, moderate dropout can improve the model's generalisation ability.

The batch size is set to 32, and the number of training epochs is set to 30. The learning rate decays at a rate of 0.7 every 5 epochs, which helps improve training stability. Especially after quick convergence in the initial stages, gradually reducing the learning rate helps avoid skipping local optima and leads to a better final model.

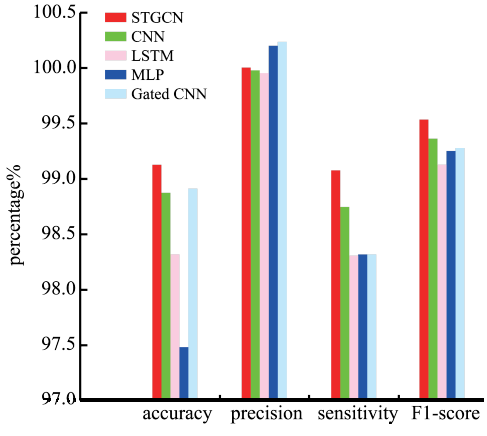
The model's performance is assessed using BCELoss to compute the loss function, while accuracy, precision, recall, and the F1 score are employed as key evaluation metrics to measure the algorithm's effectiveness.

To further evaluate the performance of the proposed STGCN model, a comparison will be made between the second-order pooling-based STGCN model and models based on conventional pooling that only consider temporal features, including CNN, LSTM, multi-layer perceptron (MLP), and gated CNN (Koca and Avci, 2024; Qian et al., 2023; Li et al., 2021). The choice of CNN, LSTM, MLP, and Gated CNN for spatiotemporal modelling is based on their respective strengths in handling spatial features, temporal features, and spatiotemporal correlations. CNN is proficient at extracting local features from spatial data; LSTM is capable of capturing long-term dependencies in time series,

while gated-CNN enhances feature extraction capabilities and improves spatiotemporal modelling through gating mechanisms.

Multiple training sets are constructed to train the models and test their performance. The average results of the five models across various metrics are shown in Figure 4.

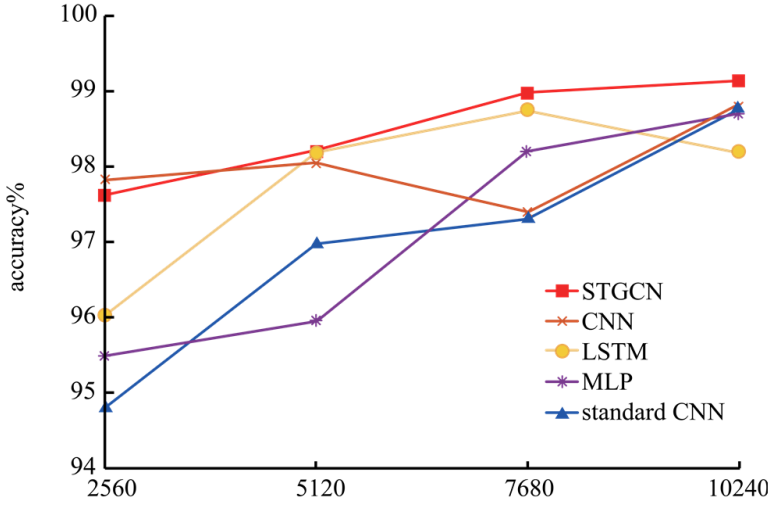
Figure 4 Comparison of model performance (see online version for colours)



As shown in Figure 4, the STGCN model proposed in this paper outperforms the CNN, LSTM, MLP, and gated CNN models in terms of accuracy, precision, recall, and F1 score.

Next, we compare the impact of sample size on model performance. The number of training samples is set to 2,560, 5,120, 7,680, and 10,240, with 33,600 testing samples. The STGCN, CNN, LSTM, MLP, and gated CNN models are trained, and the accuracy evaluation results on the test set are shown in the figure.

Figure 5 Training sample number and accuracy (see online version for colours)



As shown in Figure 5, except when the training set has 2,560 samples, where the accuracy of the STGCN model is slightly lower than that of the CNN model, the accuracy of the STGCN model is higher than that of the other models for all other training sample sizes. Moreover, for the same test accuracy, the STGCN model requires the fewest samples among the five models.

Table 4 Alarm information table

Region	*Alarm title*	*Network element name*	*Impact of alarm on equipment*
XD	SETUP_FAILURE	A2_XHGPGS	Potential local failure
XHL	BD_STATUS	20_104_SLXXJF_HW_PTN1	No Impact
TY	OTN_UAS	36_211_JCDASJF	Potential full equipment set failure
WBL	MUT_LOS	36-2-JYLWHJJF-HW-PTN-1	Potential local failure
GJ	COMMUN_FAIL	36-34-SFXJJF-HW-PTN1	No impact
XHL	BUS_ERR	22-89-DPJF-HW-PTN1	Equipment performance decline
Alarm level	Impact of alarm on business	Alarm occurrence time	Alarm clearance time
Level 3 alarm	Decline in business quality	2018/6/1 13:00:01	2018/6/1 17:23:41
Level 2 alarm	Partial disruption of business operations	2018/6/1 15:33:30	2018/6/1 15:33:35
Level 1 alarm	No impact	2018/6/1 17:04:17	2018/6/1 17:06:48
Level 4 alarm	Potential impact on business	2018/6/1 17:58:26	2018/6/1 18:00:02
Level 3 alarm	Potential complete business blockage	2018/6/1 18:59:59	2018/6/1 19:06:37
Level 4 alarm	No impact	2018/6/1 20:04:18	2018/6/1 20:07:19

5.2 Alarm correlation

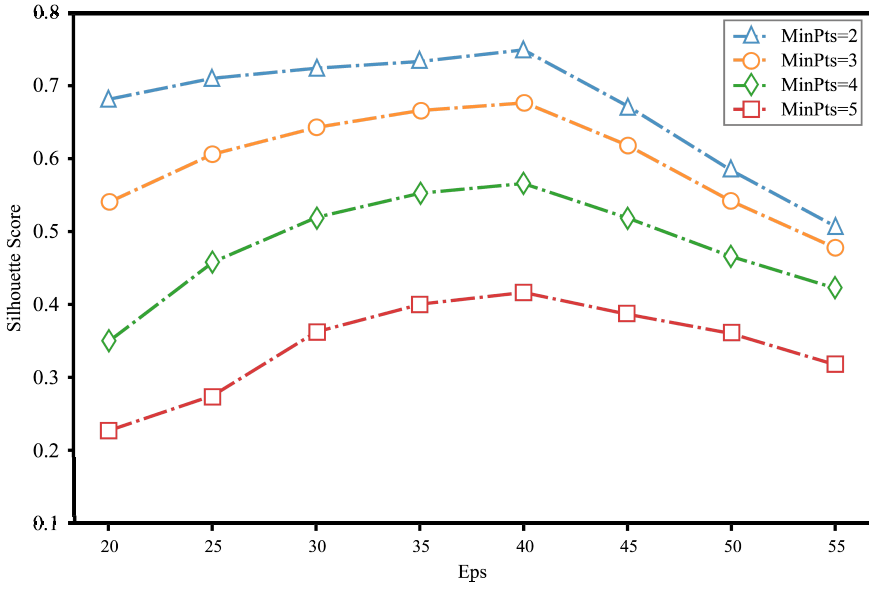
This experiment primarily uses alarm data recorded by the NMS for a week in July 2018. The table lists the detailed feature information of some alarm data from the pre-processed alarm dataset used in this experiment. In the fault source localisation of alarm data, during the pre-processing feature extraction phase, this experiment extracted eight attributes to represent an alarm instance: county, alarm title, network element name, impact on the device, alarm level, impact on the business, alarm occurrence time, and alarm clearance time.

Since the relationship between alarm severity and business interruption may not be a simple linear one, we introduced a more flexible weighting strategy and threshold setting in the model, taking into account the varying impact of different types of alarms on business operations. We use a data-driven threshold selection method (K-means clustering) to more accurately define the severity of different alarms. In terms of the weighting strategy, we employ different weight factors to adjust the final severity score based on the impact of alarms on different business units and devices. These weight

factors consider not only the frequency and duration of alarms but also integrate their potential impact on business operations and equipment.

First, compare the changes in silhouette coefficient when adjusting eps and MinPts:

Figure 6 EPS and silhouette score (see online version for colours)



From Figure 6, it can be observed that when eps is fixed, the score is higher with MinPts = 2. Therefore, MinPts is fixed first. As eps increases, the clustering score gradually increases until eps = 40, after which it starts to decrease. By adjusting the parameters (eps, MinPts) and observing the changes in the silhouette coefficient, the final clustering parameters are determined to be (40, 2).

Next, compare the clustering runtime of DBSCAN with and without KD-Tree optimisation across ten regions:

It can be observed that as the alarm data volume increases, the runtime of the optimised DBSCAN is significantly shorter than that of the original DBSCAN. During the alarm transaction set generation phase using the DBSCAN clustering algorithm, the creation of the KD-Tree saves runtime.

In the PrefixSpan experiment, different support thresholds were set to obtain various frequent sequences, and confidence thresholds were further applied to derive association rules. However, the number of rules mined remained large. To reduce the number of rules and lighten the burden for further localisation, only high-frequency alarm rules are considered. A parameter experiment was conducted on the minimum confidence value, varying it between 0.1 and 0.6, while the minimum support was set to vary between 0.040 and 0.060.

From Figure 8, it can be observed that as the support increases, the number of rules decreases gradually in both methods. This is because a higher support threshold limits low-frequency patterns, resulting in fewer sequence patterns that meet the criteria. The number of rules mined by the standard method decreases more slowly, while the number of rules after threshold filtering decreases more rapidly. By setting a high confidence

threshold, only patterns with high confidence are retained, significantly reducing the number of rules and alleviating the burden of subsequent analysis caused by the large number of rules. Additionally, it can be seen that when the support is low, the number of mined results is large, and even after confidence filtering, the number of rules remains high. However, when the support is high, the number of rules decreases significantly, regardless of whether confidence filtering is applied, but some potentially important patterns may be lost.

Figure 7 Regions and time consumption (see online version for colours)

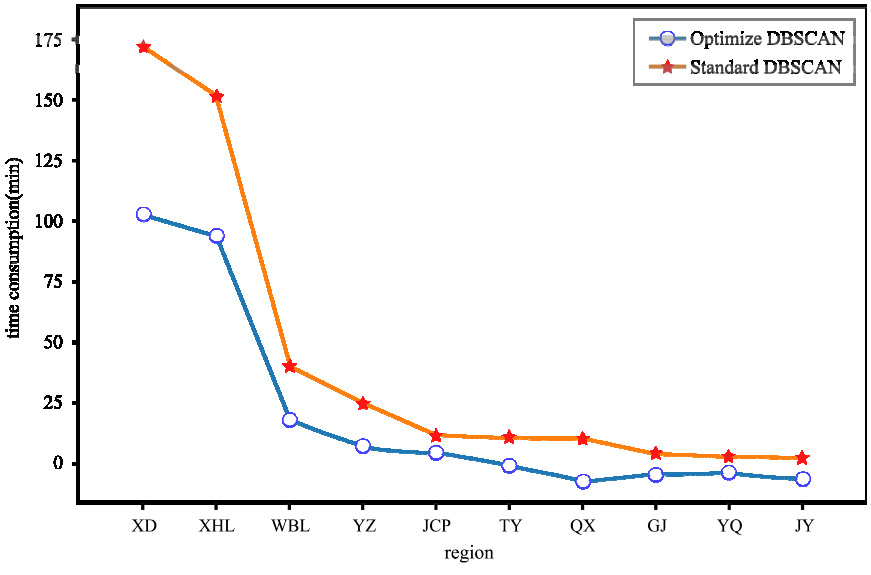
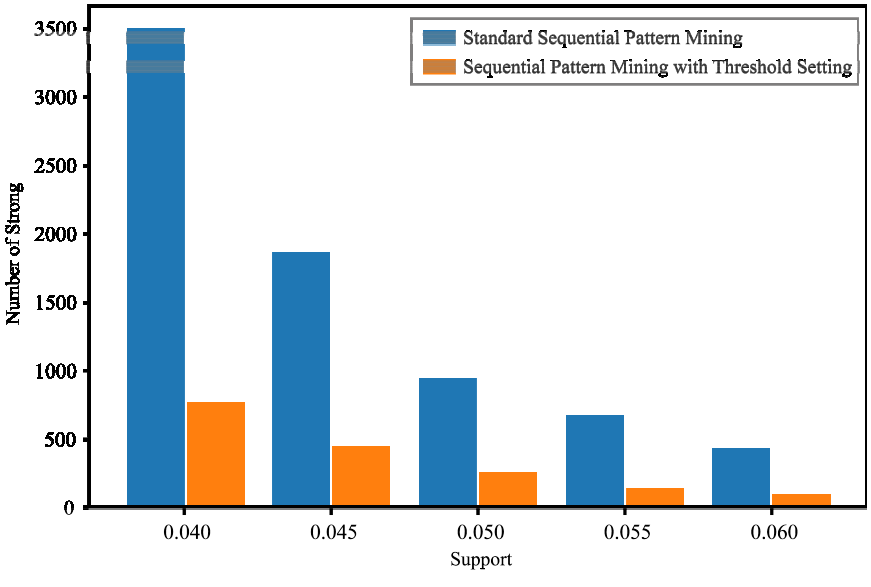


Figure 8 Support and number of strong association rules (see online version for colours)



6 Conclusions and outlook

With the development of communication networks of power grid, the network structure is gradually presenting complex and ever-changing new characteristics, and the electric-power telecommunication system faces more fault risks. Efficient and precise fault location in EPTN is essential to ensure the reliable and stable operation of power systems. In recent years, the advancement of deep learning has significantly contributed to progress in pattern recognition and data mining, offering valuable applications in fault location methodologies.

This paper proposes an intelligent fault diagnosis method based on spatiotemporal graph neural networks (STGCN) for alarm data analysis in electric-power telecommunication network fault diagnosis. By utilising the spatial and temporal dependencies of alarm events, this method achieves accurate fault event localisation and prediction.

Currently, the proposed STGCN model mainly targets offline and online analysis scenarios based on simulation analysis. The method has high accuracy and effectively shortens simulation time, improving analysis efficiency. In future research, we will focus on the challenges of network topology maintenance and limited high-resolution measurement data in some online analysis scenarios. Further adaptability design and modifications of the STGCN model will be conducted to enhance its applicability in various online analysis scenarios.

Declarations

This work was funded by the Science and Technology Project of State Grid Corporation of China: 'Modelling and Alarm Analysis Method based on Spatio-Temporal Graph for Electric-Power Optical Telecommunication Networks' (5700-202355735A-3-3-JC).

The author declares that it does not have any known interests or personal relationships that could potentially influence the reported work in this paper.

References

Bruna, J., Zaremba, W., Szlam, A. and LeCun, Y. (2013) *Spectral Networks and Locally Connected Networks on Graphs*, arxiv preprint arxiv:1312.6203.

Chen, X., Liu, C.Y., Proietti, R., Li, Z. and Yoo, S.B. (2022) 'Automating optical network fault management with machine learning', *IEEE Communications Magazine*, Vol. 60, No. 12, pp.88–94.

Defferrard, M., Bresson, X. and Vandergheynst, P. (2016) 'Convolutional neural networks on graphs with fast localized spectral filtering', *Advances in Neural Information Processing Systems*, p.29.

Gori, M., Monfardini, G. and Scarselli, F. (2005) 'A new model for learning in graph domains', in *Proceedings 2005 IEEE International Joint Conference on Neural Networks*, IEEE, Vol. 2, pp.729–734.

He, K, Zhang, X., Ren, S. and Sun, J. (2016) 'Deep residual learning for image recognition', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.770–778.

Hussain, M., Dhimish, M., Holmes, V. and Mather, P. (2019) 'Deployment of AI-based RBF network for photovoltaics fault detection procedure', *AIMS Electronics and Electrical Engineering*, Vol. 4, No. 1, pp.1–18.

Jamshidiha, S., Pourahmadi, V., Mohammadi, A. and Bennis, M. (2024) 'Power allocation using spatio-temporal graph neural networks and reinforcement learning', *Wireless Networks*, Vol. 31, pp.1–14.

Janu, D., Kumar, S. and Singh, K. (2022) 'A graph convolution network based adaptive cooperative spectrum sensing in cognitive radio network', *IEEE Transactions on Vehicular Technology*, Vol. 72, No. 2, pp.2269–2279.

Jiang, W. and Bai, Y. (2023) 'APGNN: alarm propagation graph neural network for fault detection and alarm root cause analysis', *Computer Networks*, Vol. 220, p.109485, https://doi.org/10.1016/j.comnet.2022.109485.

Jiang, Y., Dai, P., Fang, P., Zhong, R.Y. and Cao, X. (2022) 'Electrical-STGCN: an electrical spatio-temporal graph convolutional network for intelligent predictive maintenance', *IEEE Transactions on Industrial Informatics*, Vol. 18, No. 12, pp.8509–8518.

Koca, M. and Avci, I. (2024) 'A novel hybrid model detection of security vulnerabilities in industrial control systems and IoT using GCN+ LSTM', *IEEE Access*, Vol. 12, pp.143343–143351, https://doi.org/10.1109/ACCESS.2024.3466391.

Li, Z., Zhao, Y., Li, Y., Rahman, S., Wang, F., Xin, X. and Zhang, J. (2021) 'Fault localization based on knowledge graph in software-defined optical networks', *Journal of Lightwave Technology*, Vol. 39, No. 13, pp.4236–4246.

Liao, W., Bak-Jensen, B., Pillai, J.R., Wang, Y. and Wang, Y. (2021) 'A review of graph neural networks and their applications in power systems', *Journal of Modern Power Systems and Clean Energy*, Vol. 10, No. 2, pp.345–360.

Liao, W., Yang, D., Wang, Y. and Ren, X. (2020) 'Fault diagnosis of power transformers using graph convolutional network', *CSEE Journal of Power and Energy Systems*, Vol. 7, No. 2, pp.241–249.

Ma, Z., Wei, F., Song, X. and Ma, Z. (2022) 'Design and implementation of smart power firefighting IoT based on massive heterogeneous terminals', *International Journal of Information and Communication Technology*, Vol. 21, No. 3, pp.304–316.

Nguyen, B.L., Vu, T.V., Nguyen, T.T., Panwar, M. and Hovsapian, R. (2023) 'Spatial-temporal recurrent graph neural networks for fault diagnostics in power distribution systems', *IEEE Access*, Vol. 11, pp.46039–46050, https://doi.org/10.1109/ACCESS.2023.3273292.

Qian, L., Zhou, Z., Cai, X. and Zhu, P. (2023) 'Fault diagnosis combining power grid and communication system based on graph neural network', in *2023 IEEE 23rd International Conference on Communication Technology (ICCT)*, IEEE, pp.651–658.

Rivas, A.E.L. and Abrao, T. (2020) 'Faults in smart grid systems: monitoring, detection and classification', *Electric Power Systems Research*, Vol. 189, p.106602, https://doi.org/10.1016/j.epsr.2020.106602.

Wang, D., Lou, L., Zhang, M., Boucouvalas, A.C., Zhang, C. and Huang, X. (2019) 'Dealing with alarms in optical networks using an intelligent system', *IEEE Access*, Vol. 7, pp.97760–97770, https://doi.org/10.1109/ACCESS.2019.2929872.

Wu, T., Scaglione, A. and Arnold, D. (2023) 'Complex-value spatio-temporal graph convolutional neural networks and its applications to electric power systems AI', *IEEE Transactions on Smart Grid*, Vol. 15, No. 3, pp.3193–3207.

Yang, Y., Zhu, X., Song, X., Zhang, H. and Zhang, X. (2024) 'Early warning and decision-making model of geological disaster damage of transmission lines based on intelligent online monitoring technology', *International Journal of Information and Communication Technology*, Vol. 24, No. 7, pp.65–89.

Yu, B., Yin, H. and Zhu, Z. (2017) *Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting*, arxiv preprint arxiv:1709.04875.

Zhang, M., Lam, I.C., Kumar, A., Chow, K.K. and Chong, P.H.J. (2018) 'Optical environmental sensing in wireless smart meter network', *AIMS Electronics and Electrical Engineering*, Vol. 2, No. 3, pp.103–116.

Zhao, H., Yang, B., Cui, J., Xing, Q., Shen, J., Zhu, F. and Cao, J. (2023) 'Effective fault scenario identification for communication networks via knowledge-enhanced graph neural networks', *IEEE Transactions on Mobile Computing*, Vol. 23, No. 4, pp.3243–3258.