Multi-sensor fusion on hypergraph for fault diagnosis

***Abstract*— Multi-sensor information fusion techniques based on deep learning are crucial for machinery fault diagnosis. However, there are two major issues in previous research. First, the relationship between multi-sensor samples is disregarded, which is important to enhance the diagnostic performance. Second, the structure of the fusion algorithm becomes extremely complex with prolonged training when dealing with machinery equipped with a large number of sensors. To address the above two issues, our study proposes a new multi-sensor fusion mechanism that fuses multi-sensor information on hypergraphs, by building a single-sensor fusion hypergraph and a multi-sensor fusion hypergraph in the sensor space to embed the fault samples as nodes. Additionally, a dual-branch hypergraph neural network is designed to compute the two hypergraphs to obtain the feature representation of the samples and diagnose faults. The algorithm is validated on two datasets for its performance.**

***Index Terms*— Multi-sensor fusion, Fault diagnosis, Hypergraph neural network, Graph neural network, Information fusion.**

I. Introduction

As industrial technology advances, rotating machinery becomes more powerful, efficient, and precise, raising concerns about equipment safety and making fault identification a hot research area. The study of data-driven fault diagnosis, which extracts useful information from machine operating data without requiring knowledge of the structure or dynamic characteristics of the machinery, has attracted much attention due to the rapid development of artificial intelligence. The quantity and quality of the operating data determine the diagnostic accuracy; however, the limited fault samples are available for training due to the high cost of obtaining fault data in reality. It determines deep mining feature information from fault samples as an important way to resolve the fault diagnosis issue.

To mine useful information from fault samples, researchers have attempted various strategies in past studies. The features are extracted from multiple perspectives such as time domain, frequency domain, and time-frequency domain to describe faults, and information is transferred from one operating condition or machine to another operating condition or machine for fault diagnosis. Among these strategies, multi-sensor data fusion is a key technique for mining sample features.

We define signal samples captured by a single sensor as single-sensor samples (SSSs) and the composite samples captured by multiple sensors at the same moment as multi-sensor samples (MSSs). Multi-sensor fusion techniques for fault diagnosis have been well-studied, and they are typically classified into three levels, the data level, feature level, and decision level, depending on the characteristics of the fused information.

Data-level fusion approaches involve preprocessing the raw multi-sensor data prior to feature extraction. Guan [1] computed the correlation coefficients between SSSs and treated them as weights to combine SSSs into composite signals for fault classification using 1D-convolutional neural network (CNN). Wang [2] reshaped each SSS into a 2D image that was stacked to create a multi-channel color image as fault representation for further classification with CNN. Tong [3] calculated the kurtosis value of each SSS as the weight to fuse SSSs into a single 1D signal, which was then fed into a residual neural network with an attention module to determine the category of the faults. He [4] turned an MSS into a 2D image with each SSS as a row and then carried out locality-preserving projection to reduce data dimension for fault classification with a dual residual network with various layers. Since the data-level fusion approaches preserve enough information from the raw signals, they have become the fundamental preprocessing techniques for many models.

To remove noises and redundant information from the signals and obtain more effective representations of fault samples, the feature-level fusion approaches extract features from SSSs separately and combine the features in a certain way before decision-making. Tang [5] transformed SSS into a symmetrized dot pattern (SDP) feature and then stacked the SDPs generated from various sensors into a multi-channel SDP, achieving fault diagnosis of drive motors using support vector machine. Since deep neural networks have an end-to-end learning mechanism for simultaneous feature extraction and classification, a multi-branch neural network structure is developed where single sensor features are first extracted through a branch network and then multiple branches are fused by a single network or classifier. Jalayer [6] processed the drive-end and fan-end sensor signals of a motor using the fast Fourier transform (FFT) and continuous wavelet transform (CWT). Then, these FFT and CWT features were further separately extracted using convolutional layers and long short-term memory, along with statistical features as input to multi-layer perceptron (MLP) for fault classification. Cui [7] developed a multi-task CNN architecture with multi-sensor signals as inputs and two tasks as outputs. In addition to the cross-entropy, the intra-class dispersion of the feature values in the middle fusion layer of the network is optimized with metric learning to improve the effectiveness of diagnosis. Guo [8] computed the cyclic spectra of motor current signals and fused the covariance matrices of the corresponding modal vectors in the cyclic spectra of multiple signals, and these signals were input into an extreme learning machine to achieve fault classification.

Similar to a voting system, decision-level fusion algorithms integrate the results of various classifiers to make a final decision. The signals from two sensors were independently categorized with K-nearest neighbor (KNN), and the outcomes are fused using a waterfall fusion model by Safizadeh [9]. Shao [10] classified SSSs with wavelet autoencoders (AEs), and the classification accuracies were utilized as weights to combine each classifier's output to obtain the final result. To achieve the failure detection of marine electric thruster bearings, Zhang [11] performed evidential reasoning to combine the outcomes of multiple 1D CNNs trained with SSSs, where the training accuracies and the losses of the associated branches were treated as weights and reliability. Zhong [12] pretrained CNNs on SSSs captured by two vibration sensors and two pressure sensors, respectively, then fused the multi-sensor results based on Dempster-Shafer evidence theory and assigned pseudo-labels to the unlabeled samples, attaining the monitoring of the hydraulic valve in a semi-supervised learning way. In those decision-based fusion algorithms, each sensor signal is treated independently and the coupling information between different sensors is lost in fact, leading to degraded performance.

The categorization of the three multi-sensor fusion mechanisms is not strict, and some algorithms synthesizing different fusion techniques are available. For instance, Yan [13] chose any two signals from five sensors to create a generalized shaft orbit and then sent a series of orbits into a multi-branch CNN (MB-CNN) to further extract the features and classify the faults, accomplishing multi-sensor information fusion at both the data level and feature level. Niu [14] integrated the characteristic frequencies of the inner ring, outer ring, and ball of bearing into 3-channel color images created from the raw multi-sensor signals at both the data and feature levels, and determined the fault types with the residual CNN.

Although these multi-sensor fusion algorithms have achieved success in machinery fault diagnosis, they also have two drawbacks.

First, in the fault diagnosis task, samples are cut from sequential and periodic sensor signals, revealing the existence of complex correlations between MSSs and SSSs that are helpful in recognizing fault types. However, most multi-sensor fusion algorithms follow the assumption of independent identically distribution (IID) and neglect the correlations between samples. Recent publications [15] [16] describe some graph-based multi-sensor fusion techniques that construct graphs to consider the correlations between SSSs and mainly disregard the correlations between MSSs as a whole. Moreover, a simple graph can only describe the pairwise correlation between samples, and cannot present the high-order correlation between multiple sensor signals with common characteristics. This results in the loss of information and limits the multi-sensor techniques to achieve better performance.

Second, a large number of sensors are equipped in the actual machines or engineering systems; however, most existing algorithms deal with the information fusion of a few sensors. As the number of sensors grows, the structure of the models will become complicated, especially for deep learning algorithms based on feature fusion or decision fusion, leading to overfitting and increasing the computational complexity.

To address the above two challenges, our study proposes a novel multi-sensor fusion mechanism, i.e., fusion on the hypergraph. Distinguishing from the traditional three fusion mechanisms, it embeds SSSs and MSSs into the hypergraphs as nodes, builds correlations between them individually, and fuses the multi-sensor information during the construction of the hypergraphs. A multi-sensor fusion hypergraph neural network (MsfHGNN) with a dual-branch structure is designed to learn the fault feature representation and diagnose faults by computing the hypergraphs with multi-sensor information.

The main contributions of this paper are threefold.

1) A novel multi-sensor fusion mechanism based on hypergraphs is proposed. Specific hypergraphs based on SSSs and MSSs are generated, respectively, not only achieving effective fusion of multi-sensor information but also acquiring the correlation between fault samples.
2) A hypergraph learning framework MsfHGNN is designed. The architecture of MsfHGNN is fixed as a dual-branch structure that does not change with the variation of the number of sensors, and the training process is also efficient.
3) On two rotating machinery datasets for fault diagnosis, the proposed model achieves superior performance as compared with previous multi-sensor fusion methods with multi-branch structures. Additionally, it is also resistant to noise interference.

The rest of the paper is organized in the following manner. Section II introduces the basic concepts and computational principles of hypergraphs, while Section III provides the novel multi-sensor fusion framework and the algorithm details. Section IV presents the experimental validation, and the entire paper is concluded in Section V.

II. Theoretical Background

A. Hypergraph

A simple graph is a kind of data structure describing the pairwise relationship between samples. As deep learning has been applied to graph learning in recent years, graph neural networks (GNNs) such as ChebyNet [17], graph convolutional networks (GCN) [18], etc., break away from the assumption of IID and efficiently accomplish graph learning tasks, such as node classification, graph classification, and link prediction. GNNs have led to breakthroughs in social network analysis, protein function prediction, and intelligent transportation systems. However, simple graphs are only employed to describe the pairwise correlations between the nodes and their neighbors, and high-order relationships between multiple nodes are not taken into account. This issue can be solved using hypergraph $G = (V, E, \mathbf{W})$, consisting of a set of nodes V, hyperedges E, and the weights of hyperedges $\mathbf{W}$. In contrast to simple graphs, hypergraphs allow hyperedges to connect multiple nodes, characterizing the high-order relationships between nodes and reflecting their shared properties.

To simplify the problem, we define $\mathbf{W} = \mathbf{I}$ in this work, i.e., each hyperedge has an equal weight. The adjacent matrix is adopted to depict the relationship between the nodes in a simple graph, and the relationship of the nodes in a hypergraph is represented by the incident matrix $\mathbf{H} \in R^{|V|\times|E|}$. The rows and columns of $\mathbf{H}$ correspond to the nodes and hyperedges, respectively, with a value of 1/0 indicating whether or not the node lies on the hyperedges. For $v \in V$, $e \in E$,

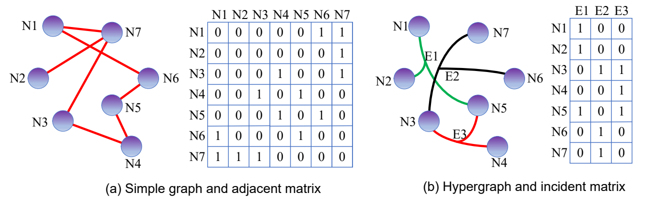
$$\mathbf{H}(v,e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{if } v \notin e \end{cases} \tag{1}$$

The node degree matrix and hyperedge degree matrix are denoted by the diagonal matrices $\mathbf{D}_V \in R^{|V|\times|V|}$ and $\mathbf{D}_E \in R^{|E|\times|E|}$, respectively; their diagonal elements are specified as follows:

$$d_e = \sum_{v\in V} \mathbf{H}(v,e) \tag{2a}$$

$$d_v = \sum_{e\in E} \mathbf{H}(v,e) \tag{2b}$$

Fig. 1 lays out the differences between a simple graph and a hypergraph. A hypergraph is composed of multiple hyperedges. Each hyperedge can serve as both the connection between the nodes similar to a simple graph and a group of nodes with common characteristics along with the attribute of the hypergraph; thus a hypergraph can be described by a series of attributes [19]. A new attribute shared by a set of nodes can be achieved by adding a hyperedge to the hypergraph, i.e., adding a column to the incident matrix of hypergraph. It implies that extra hyperedges can be added into the hypergraph to produce more high-order correlations or fuse information from diverse sources.



	N1	N2	N3	N4	N5	N6	N7
N1	0	0	0	0	0	1	1
N2	0	0	0	0	0	0	1
N3	0	0	0	1	0	0	1
N4	0	0	1	0	1	0	0
N5	0	0	0	1	0	1	0
N6	1	0	0	0	1	0	0
N7	1	1	1	0	0	0	0

(a) Simple graph and adjacent matrix

	E1	E2	E3
N1	1	0	0
N2	1	0	0
N3	0	1	1
N4	0	0	1
N5	1	0	1
N6	0	1	0
N7	0	1	0

(b) Hypergraph and incident matrix

Fig. 1: Comparison of simple graph and hypergraph.

B. Hypergraph Learning

Based on graph-cut theory, Zhou [20] proposed a hypergraph learning framework with both empirical loss and the smoothness of the label distribution as the optimization targets. Feng [21] designed hypergraph neural network (HGNN) and implemented hypergraph convolution based on the hypergraph Laplace transform, generalizing hypergraph learning into a deep learning framework. Bai [22] focused on the weights of the nodes on the hyperedges and presented an attention technique for hypergraph convolution. Yadati [23] transformed the hypergraph into a simple graph in a certain way and performed GCN to learn the node representations. Gao [24] expanded the spectral analysis of hypergraph convolution into the spatial domain and assigned trainable group weights to the hyperedges, which achieved the multi-modal fusion.

Benefiting from their capabilities, hypergraphs can be utilized in 3D object retrieval [25], behavior recognition [26], and histopathological image analysis [27]. Due to the periodicity and the continuity of signals, complex relationships between fault samples potentially exist; hence, hypergraphs have been applied to fault diagnosis of rotating machinery. Yan [28] built hypergraphs by decomposing the signals into sub-signals under various resolutions and then concatenated the hypergraphs to approximate the data structure hidden in the fault samples. Shi [29] designed a hypergraph convolutional layer in AE, causing the hidden representation of the AE to capture high-order corrections between the unlabeled fault samples.

C. Hypergraph Neural Network

The hypergraph learning problem can be summarized [20] as

$$\arg\min_{f} \{R_{emp}(f) + \Omega(f)\} \tag{3}$$

where $f(\cdot)$ is the classifier, $R_{emp}(f)$ denotes the empirical loss, and $\Omega(f) = f^T \mathbf{\Delta} f$ is a regularization item. $\mathbf{\Delta} = \mathbf{\Phi}^T \mathbf{\Lambda} \mathbf{\Phi}$ is the Laplacian matrix where $\mathbf{\Phi} = [\phi_1, \phi_2, ..., \phi_N]$ are eigenvectors and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_N)$ is a diagonal matrix consisting of eigenvalues. Hence, the spectral convolution of the hypergraph can be written as:

$$h * x = \mathbf{\Phi}((\mathbf{\Phi}^T h) \odot (\mathbf{\Phi}^T x)) = \mathbf{\Phi} h(\mathbf{\Lambda}) \mathbf{\Phi}^T x \tag{4}$$

where $\odot$ represents the element-wise product.

Since $\mathbf{\Phi}$ requires eigendecomposition of the matrix, the computation of hypergraph convolution needs much more memory and is more time-consuming. Leveraging the idea of simplifying the spectral convolution of simple graph in ChebyNet and GCN, the hypergraph convolution is directly simplified [21] as:

$$h * x = \theta \mathbf{D}_V^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_E^{-1} \mathbf{H}^T \mathbf{D}_V^{-1/2} x \tag{5}$$

where $\mathbf{D}_E^{-1}$ and $\mathbf{D}_V^{-1/2}$ are viewed as normalization items, and θ is the convolution parameter.

HGNN consists of a stack of hypergraph convolutional layers (as in Fig. 2), each of which can be defined as:

$$\mathbf{X}^{l+1} = \sigma(\mathbf{D}_V^{-1/2} \mathbf{H} \mathbf{W} \mathbf{D}_E^{-1} \mathbf{H}^T \mathbf{D}_V^{-1/2} \mathbf{X}^l \mathbf{\Theta}^l) \tag{6}$$

where the input feature is $\mathbf{X}^l \in R^{N\times C_1}$ and the output is

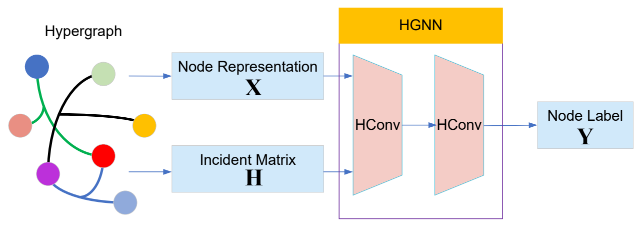


Fig. 2: HGNN

$\mathbf{X}^{l+1} \in R^{N\times C_2}$, with only one learnable parameter $\mathbf{\Theta}^l \in R^{C_1\times C_2}$.

Similar to GNNs, as the number of hypergraph convolutional layers increases, HGNN suffers from over-smoothing, leading consistent features of all nodes. Consequently, the number of layers of HGNN used in this paper is set to 2.

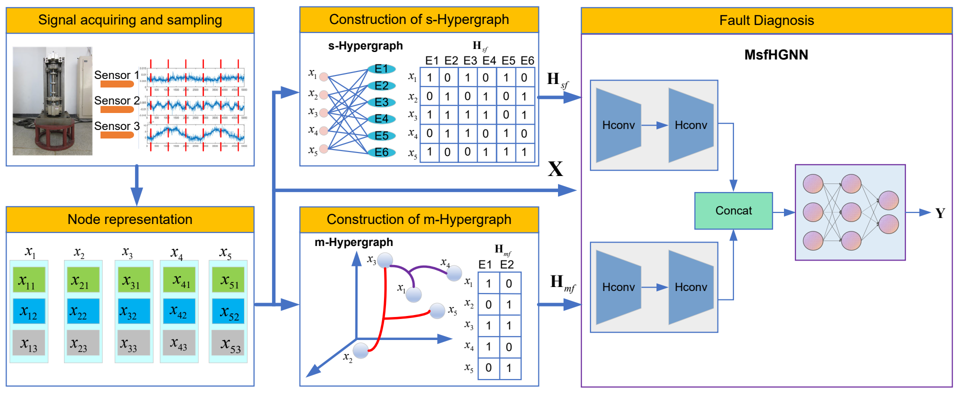


Fig. 3: The pipeline of the proposed algorithm.

III. The Proposed Algorithm

The framework of our proposed algorithm is depicted in Fig. 3 and mainly consists of four parts: multi-sensor raw signal acquisition and sampling samples for MSSs and SSSs, node representation, hypergraphs construction for multi-sensor fusion, and hypergraph computation for fault diagnosis. In this section, the basic hypergraph generation strategy is introduced first, followed by the construction methods for two hypergraphs from SSSs and MSSs, respectively. Then MsfHGNN for computing the two hypergraphs is detailed, and finally some discussion and analysis are provided.

A. Hypergraph generation

In contrary to social networks, there is no explicit hypergraph structure among fault samples. Therefore, the correlations between samples needs to be established manually, i.e. representing nodes and creating hyperedges.

Due to the limited characterization capability of the raw signals, the fault samples cut from the signals are transformed into the frequency domain by FFT as node representations, which are constrained to [0,1] by maximum-minimum normalization as:

$$x_{\text{FFT}} = \text{FFT}(x_{\text{raw}}) \tag{7a}$$

$$x = \frac{x_{\text{FFT}} - \min(x_{\text{FFT}})}{\max(x_{\text{FFT}}) - \min(x_{\text{FFT}})} \tag{7b}$$

where x is used as the node representation with strong noise resistance compared to the raw signals.

Assuming there are N samples considered as nodes, we calculate the Euclidean distance between nodes based on the KNN to construct hyperedges, and the incident matrix of the hypergraph $\mathbf{H} \in R^{N\times N}$ is

$$\mathbf{H}_{ij} = \begin{cases} 1 & x_i \in \text{KNN}(x_j) \text{ or } i = j \\ 0 & \text{others} \end{cases} \tag{8}$$

where x_i denotes the node representation of the ith sample, and $\text{KNN}(x_j)$ stands for the K nearest nodes of x_j in the feature space, with the distance of nodes calculated as $d = \|x_i - x_j\|_2$. The generated hypergraph has N hyperedges with the same number of nodes.

B. Single-sensor fusion hypergraph

The multi-sensor fault dataset $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_S\}$ contains S SSS datasets, where $\mathbf{X}_i$ is captured by the ith sensor, $i = 1, 2, ..., S$. For each $\mathbf{X}_i$, we apply the hypergraph generation method described in Section III-A to construct a single-sensor hypergraph with the incident matrix $\mathbf{H}_i$. To attain multi-sensor fusion, the hypergraphs are concatenated to create a single-sensor fusion hypergraph, defined as s-Hypergraph, whose incident matrix is

$$\mathbf{H}_{sf} = \text{concat}(\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_S) \tag{9}$$

where $\mathbf{H}_{sf} \in R^{N\times NS}$ and "concat" denotes matrix concatenation operation.

Fig. 4 depicts the construction of the s-Hypergraph. The construction of s-Hypergraph can be summarized in two steps. First, single- sensor hypergraphs are created from SSSs respectively, and the relationship between the SSSs captured by the same sensor is then acquired. Second, multiple single-sensor hypergraphs are concatenated to achieve information interaction and fusion between SSSs from various sensors.

It is worth noting that the concatenation operation of multiple hypergraphs for multi-sensor fusion is only suitable to hypergraphs; moreover, it cannot be performed in simple graphs and usually requires a multi-branch network to fuse multiple graph information. In essence, the hypergraph can aggregate more information by adding additional hyperedges. Each SSS in the samples can be regarded as a type of feature of samples and the corresponding single-sensor hypergraph describes high-order correlations between samples in the feature space, hence the concatenation of hypergraphs is equivalent to adding groups of hyperedges generated in various feature spaces. The s-Hypergraph takes advantage of this characteristic of the hypergraph and stacks a series of hyperedges representing different sensor information, and the node representations can be updated with multi-sensor information through s-Hypergraph learning.

C. Multi-sensor fusion hypergraph

As a whole, MSS $x_i = \{x_{i1}, x_{i2}, ..., x_{iS}\}$ consists of a series of SSSs captured simultaneously and contains multi-sensor coupling information that is helpful to enhance performance but not considered in most previous studies. To mine

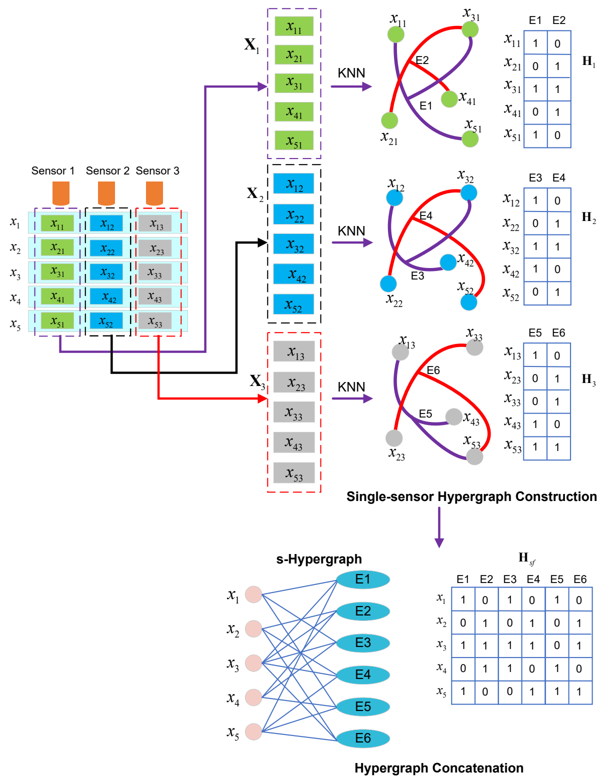


Fig. 4: The generation of s-Hypergraph.

the information from MSSs, we establish a sensor space R^S with each dimension representing one sensor. MSS can be viewed as a point $x_i = (x_{i1}, x_{i2}, ..., x_{iS})$ while the coordinate value $x_{ij} \in R^d$ is the feature vector of the jth SSS in MSS.

Define the distance between MSSs in R^S as

$$d(x_i, x_j) = \sum_{s=1}^{S} \|x_{is} - x_{js}\|_2 \tag{10}$$

which is used as the distance metric in KNN. A multi-sensor fusion hypergraph with incident matrix $\mathbf{H}_{mf} \in R^{N\times N}$, defined as m-Hypergraph, is constructed based on the method in Section III-A. Fig. 5 presents the generation of m-Hypergraph. The m-Hypergraph is generated by projecting MSS into the sensor space and redefining the distance metric in R^S. This allows it to incorporate not only the correlations between MSSs but also the coupling relationships between multiple sensors. It is worth noting that the dimension of $\mathbf{H}_{mf}$ does not depend on the number of sensors but only on the quantity of samples.

Moreover, the nodes in s-Hypergraph/m-Hypergraph are consistent, and the difference is the hyperedges. Each hyperedge in s-Hypergraph is built from the SSSs of a single sensor, reflecting the correlation between SSSs. There are a total of NS hyperedges proportional to the number of sensors, and the dimension of the incident matrix is $N \times NS$. For m-Hypergraph, each hyperedge is constructed based on the distance between MSSs, representing the correlation between MSSs as a whole, with a total of N hyperedges that are independent of the number of sensors.

D. MsfHGNN

After obtaining s-Hypergraph and m-Hypergraph, the hypergraph computation framework MsfHGNN is designed to compute the two hypergraphs individually and fuse their results for fault diagnosis as shown in Fig. 6. First, let $\mathbf{X} = \text{concat}(\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_S)$, i.e., $x_i = \text{concat}(x_{i1}, x_{i2}, ..., x_{iS})$, which means the node representation of s-Hypergraph and m-Hypergraph is the concatenation of SSS features. Then, s-Hypergraph neural network (s-HGNN) and m-Hypergraph neural network (m-HGNN), as two branches of MsfHGNN, are both set up with two hypergraph convolutional layers:

$$\mathbf{Z}_{sf} = \text{HConv}(\text{HConv}(\mathbf{X}, \mathbf{H}_{sf}), \mathbf{H}_{sf}) \tag{11a}$$

$$\mathbf{Z}_{mf} = \text{HConv}(\text{HConv}(\mathbf{X}, \mathbf{H}_{mf}), \mathbf{H}_{mf}) \tag{11b}$$

where HConv represents hypergraph convolution. The computed results of s-Hypergraph and m-Hypergraph $\mathbf{Z}_{sf}$, $\mathbf{Z}_{mf}$ are joined and input into an MLP:

$$\mathbf{Y} = \text{MLP}(\text{concat}(\mathbf{Z}_{sf}, \mathbf{Z}_{mf})) \tag{12}$$

where $\mathbf{Y}$ denotes the prediction of the fault sample labels. The cross-entropy is employed as the loss function.

E. Summary

The entire algorithm can be summarized as follows:

1) An MSS dataset is created by splitting up the operating data of a machine installed with S sensors into N samples, where each MSS comprises S SSSs. The dataset is divided into training and testing portions.
2) Extract FFT features from all SSSs. Build node feature representations $\mathbf{X}$ by combining all SSS features.
3) Construct s-Hypergraph and m-Hypergraph from SSSs and MSSs separately, and obtain the corresponding incident matrices $\mathbf{H}_{sf}$ and $\mathbf{H}_{mf}$, acquiring two types of high-order corrections between SSSs captured by a single sensor and between MSSs as a whole in the sensor space.
4) Train MsfHGNN model by feeding $\mathbf{X}$, $\mathbf{H}_{sf}$ and $\mathbf{H}_{mf}$ into MsfHGNN, and only the labels of the training samples are known. During the training process, multi-sensor fusion is implemented in three ways. First, the initial node representation is made by combining FFT features of SSSs from different sensors. Second, s-Hypergraph is generated by concatenating single-sensor hypergraphs. In s-Hypergraph learning, the node information is aggregated to the hyperedges constructed by different sensors and then aggregated to the center node; multi-sensor information is fused into the updated node representation. Third, the coupling information of multiple sensors in the sensor space is revealed by the m-Hypergraph with MSSs as the nodes.
5) The labels of the test samples in $\mathbf{X}$ are predicted by inputting $\mathbf{X}$, $\mathbf{H}_{sf}$ and $\mathbf{H}_{mf}$ into the trained MsfHGNN model.

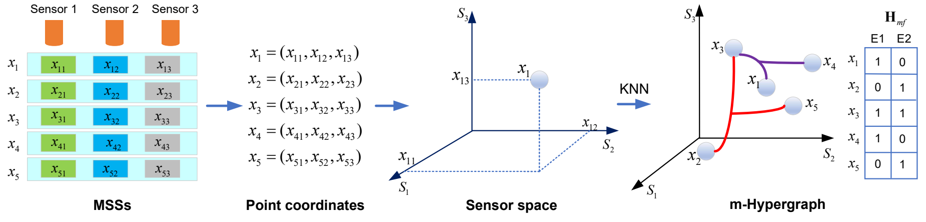


Fig. 5: The generation of m-Hypergraph.

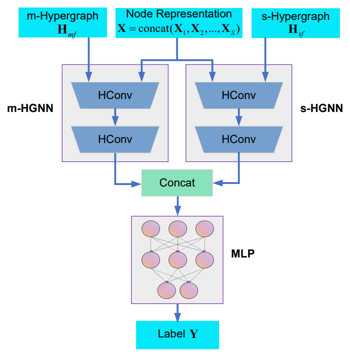


Fig. 6: MsfHGNN

IV. Experiments

A. Datasets

The Paderborn dataset and AMB-S5 dataset are adopted to verify the effectiveness of the proposed algorithm.

The Paderborn dataset is a benchmark widely used in rolling bearing fault diagnosis, designed by Paderborn University team [30], which is collected from a horizontal test rig (shown in Fig. 7) equipped with vibration, current, and torque sensors. The signals captured by the two current sensors are chosen for our study. The dataset contains three types of faults: normal, inner bearing ring, and outer bearing ring, collected from 15 bearings, corresponding to five bearings per type of fault (see TABLE I). Each bearing is subjected to 20 tests at a speed of 1500 rpm, a load torque of 0.1 N.m, and a radial force of 1000 N. There are 14996 samples in total, with a 64 kHz sampling rate and a length of 5120 points. The training and testing data come from various tests, accounting for 80% of the training data.

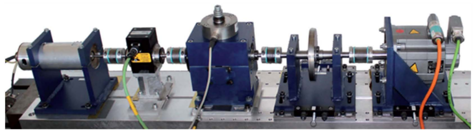


Fig. 7: The platform of Paderborn dataset [30].

TABLE I: Paderborn dataset setting

Fault Type	Sample number	Bearings for test
normal	5000	K001,K002,K003,K004,K005
inner ring fault	4999	KI04,KI14,KI16,KI18,KI21
outer ring fault	4997	KA04,KA15,KA16,KA22,KA30

The AMB-S5 dataset is collected by our team through a series of manually designed experiments, obtained from a vertical rig (shown in Fig. 8) supported by active magnetic bearings (AMB) with five displacement sensors. In addition to being used as feedback for rotor suspension control, the signals acquired by the displacement sensors are employed as vibration quantity for monitoring. The dataset primarily focuses on four types of mechatronics system-level faults, including normal, unbalance, misalignment, and rub-impact. Each type of fault is tested twice and a total of 4356 samples with a length of 5000 points and a sampling rate of 25 kHz are collected (see TABLE II). Moreover, the training and testing data are selected at random from different tests.

There is no overlap between samples in the two datasets, and the data for testing and training come from different tests, which increases the within-class scatter and raises the challenge of the datasets. It is worth noting that all samples in the two datasets have corresponding tagged labels, and the labels of testing samples are masked during the training process. The accuracy, the ratio of correctly identified samples among testing samples, is utilized to evaluate how effective the algorithm is. All experiments are conducted 10 times with randomly chosen training and testing data and the average accuracy is acquired.

Our experiments are carried out on a computer with a Central Processing Unit of Intel(R) Xeon(R) E5-2697 v4@2.30GHz and a Graphics Processing Unit (GPU) of N-VIDIA GeForce GTX 3090Ti. The GPU is used to accelerate the execution of algorithms. The softwares used include Python 3.9.7, Sk-learn 1.1.1, Pytorch 1.10.1, and DeepHypergraph Toolbox [21].

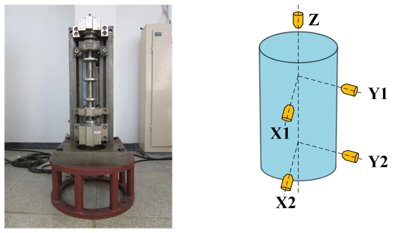


Fig. 8: Test rig for AMB-S5 dataset and the layout of sensors.

TABLE II: AMB-S5 dataset setting

Fault Type	Test1 sample number	Test2 sample number
normal	856	372
rotor unbalance	968	740
misalignment	476	500
rub-impact	152	292

B. Parameters setting

MsfHGNN is compared with five other multi-sensor fusion algorithms, each of which selects FFT as features. SSS features are directly concatenated as the node representation of HGNN+ [24] with a two-layer structure. The number of branches is consistent with the number of sensors in multi-branch ChebyNet (MB-ChebyNet) [16], multi-branch GCN (MB-GCN) [15], MB-CNN [13], and multi-branch HGNN (MB-HGNN) [31], all of which have a multi-branch structure shown in Fig. 9. Each branch cascades two graph convolutional/convolutional/hypergraph convolutional layers as the branch backbone, with MLP to fuse multiple branches. In MB-CNN, three fully connected layers are included in the MLP, while all other algorithms have two fully connected layers. Predefine $K = 10$ to generate graphs or hypergraphs in HGNN+, MB-ChebyNet, MB-GCN, MB-HGNN, and MsfHGNN, and then set the output dimension of graph/hypergraph convolutional layers to 512 and 32. The epoch number is 2000 and 1000 in Paderborn and AMB-S5, respectively.

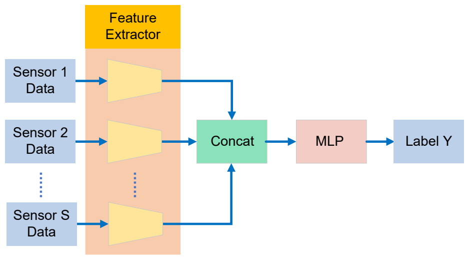


Fig. 9: Multi-branch multi-sensor fusion algorithm framework.

C. Comparison

TABLE III presents the performance of the six multi-sensor fusion algorithms on the two datasets. It can be seen that MsfHGNN has the highest accuracy among the six multi-sensor fusion algorithms. In comparison to MB-CNN, MsfHGNN acquires additional information about the correlation between samples due to breaking the IID assumption. As opposed to MB-ChebyNet and MB-GCN, MsfHGNN captures the high-order correlation between samples beyond the pairwise relationship. Specifically, HGNN+ and MB-HGNN are feature-level fusion algorithms and capture only the relationship between SSSs, while MsfHGNN takes the relationship between MSSs/SSSs into account simultaneously and achieves multi-sensor fusion on hypergraphs.

In terms of training time, MsfHGNN only exceeds MB-CNN on Paderborn, but on AMB-S5, it trains faster than MB-ChebyNet, MB-CNN, and MB-HGNN. This result roughly shows that as the number of sensors increases, the advantage of the proposed algorithm in training time grows, and a more thorough analysis is provided in Section IV-E.

D. Ablation experiment

MsfHGNN primarily consists of two branches, s-HGNN, and m-HGNN, computing s-Hyergraph and m-Hypergraph that fuse multi-sensor information from different aspects. Therefore, we conduct ablation experiments testing both s-HGNN and m-HGNN, to confirm the impact of each on the overall algorithm. In the last hypergraph convolutional layer of s-HGNN and m-HGNN, the output dimension is adjusted to

TABLE III: Comparison of six multi-sensor fusion algorithms

Algorithms	Paderborn		AMB-S5	
	Accuracy/%	Training time/s	Accuracy/%	Training time/s
HGNN+	92.09±2.01	117.65	99.52±0.26	**37.20**
MB-ChebyNet	94.44±1.61	298.20	99.05±0.48	223.52
MB-GCN	92.96±1.76	**52.49**	98.45±0.71	61.26
MB-CNN	93.17±4.23	13307.26	96.60±0.82	3326.03
MB-HGNN	96.59±2.00	350.78	99.47±0.20	175.28
MsfHGNN	**97.07±2.28**	465.68	**99.53±0.24**	138.04

match the number of fault types, and the activation function is changed to the Softmax function.

According to Fig. 10, on Paderborn, m-HGNN achieves approximate performance with MsfHGNN; however, s-HGNN has poorer performance, and the opposite conclusion is drawn on AMB-S5. It is demonstrated that under the circumstances of different datasets with a various numbers of sensors, the multi-sensor information acquired by s-Hypergraph and m-Hypergraph separately plays different roles in diagnosis.

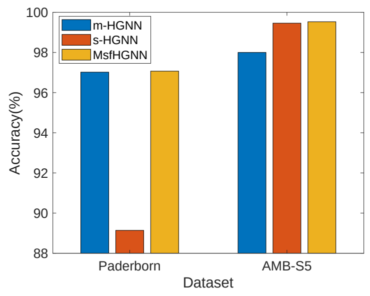


Fig. 10: Ablation experiment on two datasets.

Two Venn diagrams are shown in Fig. 11 to illustrate the quantities of samples successfully identified by the corresponding algorithms. The circle intersection denotes that two or three corresponding algorithms properly categorize the samples. Clearly, s-HGNN and m-HGNN each have advantages and drawbacks of their own. On Paderborn, 188 samples are correctly categorized by m-HGNN and incorrectly classified by s-HGNN. 14 samples, however, that m-HGNN is unable to accurately classify, are successfully classified by s-HGNN. MsfHGNN effectively identifies 189 out of the 202 samples by integrating the characteristics of s-HGNN and m-HGNN.

On AMB-S5 with varied numbers of sensors, we further analyze s-HGNN, m-HGNN, and MsfHGNN. In this experiment, 2/3/4/5-sensor data combinations are randomly selected from AMB-S5 for training, corresponding to 10/10/5/1

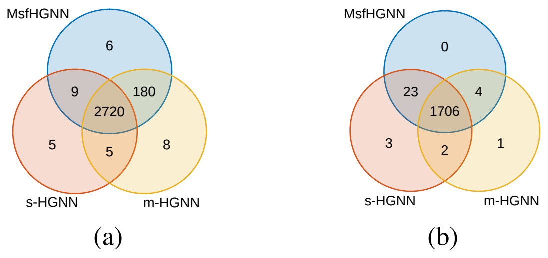


Fig. 11: Venn diagrams (a) Paderborn (b)AMB-S5.

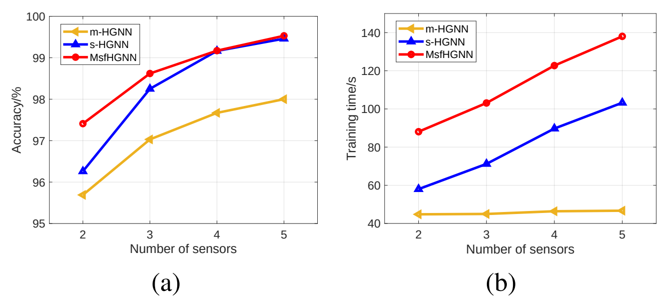


Fig. 12: Ablation experiment under various number of sensors.

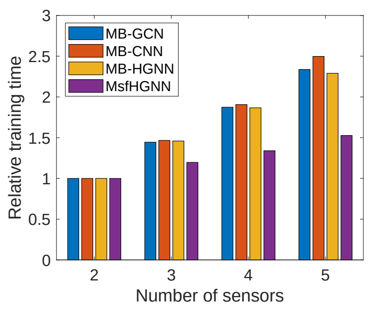


Fig. 13: Relative training time under various sensors.

combinations. The average of accuracies corresponding to the combinations with the same number of sensors is utilized as the final result.

In most cases, the more hyperedges there are, the more information in the hypergraph there is. More information will be brought to the s-Hypergraph as more sensors are added since the number of hyperedges NS is linearly associated with the number of sensors S. Since the sensor space R^S used to build the m-Hypergraph has a higher dimensionality and contains more multi-sensor coupling information as S expands, m-Hypergraph always maintains N hyperedges but can gather richer information. As a result, as shown in Fig. 12(a), the accuracy of both s-HGNN and m-HGNN significantly increases with S.

According to [22], the computational complexity of the hypergraph convolution is $O(2N^2E+2E^2N)$ which depends on the number of nodes N and the number of hyperedges E. In m-Hypergraph, N and E are constant as S rises, the dimension of $\mathbf{H}_{mf}$ remains $N \times N$; hence the computational cost for m-HGNN remains almost constant. In s-Hypergraph, the hyperedge number is NS and the training time of s-HGNN should grow quadratically with S but it does not because all operations involve matrix multiplications, which can be executed in parallel on the computer's GPU. The experiments support the findings of the study in Fig. 12(b).

Since m-Hypergraph and s-Hypergraph extract information from multiple sensors in different perspectives, both s-HGNN and m-HGNN have advantages in terms of recognition accuracy or training time and can complement each other. As seen in the aforementioned experiments, MsfHGNN successfully unifies the two within a single computational framework and further enhances the recognition performance.

E. The effect of sensor number on training time growth

Similar to the experiment setting in Section IV.D, we further analyze the relationship between the increase of sensors and the growth of the algorithms training time on AMB-S5. The algorithms with multi-branch structure and MsfHGNN are selected for comparison. To explicitly compare the growth rate of training time as S increases, define the relative training time $\bar{t}_a$ as:

$$\bar{t}_a = \frac{t_a}{t_2} \tag{13}$$

where t_a is the training time of the algorithm when $S = a$. In Fig. 13, the experimental results show that the multi-branch structure algorithms' training time quickly climbs as S increases meanwhile the complexity of the network structure increases. MsfHGNN's training time growth is the slowest. Three main factors account for this: first, MsfHGNN's network structure always maintains two branches and does not grow with S, keeping the structure simple; second, the computational cost of m-HGNN stays constant, flattening the overall growth trend of MsfHGNN's training time; and third, because matrix multiplication is the primary operation and can be more readily accelerated in parallel on the GPU, its training time growth is noticeably slower than that of other multi-branch algorithms. Therefore, MsfHGNN is more suitable for diagnosing machinery equipped with a large number of sensors.

F. Influence of Hyperparameters

In the MsfHGNN framework, there are two key hyperparameters: the K in KNN employed in the generation of hypergraphs, and the number of layers of HGNN. We test on the Paderborn dataset and discuss the selection of these two parameters.

1) Evaluation of K in KNN: The performance of MsfHGNN is evaluated as K varies, and the results are presented in Fig. 14(a). The results show that the performance of the algorithm improves as K rises initially, but when K exceeds a certain value, the algorithm's performance no longer improves and instead fluctuates within a narrow range. Thus, it indicates that the nodes on the same hyperedge have common attributes; when K is small, the nodes with common features but relatively far away from the center node may be lost, resulting in an information loss. When K is too large, all nodes that share common attributes with the central node are laid on the hyperedge without sacrificing information; however, the nodes without common attributes shared with the center node can potentially be introduced. Therefore, a compromised K value is adopted to achieve a balance, and the above analysis is presented in Fig. 14(b).

2) Evaluation of the Number of Layers: Generally, graph/hypergraph neural networks are designed with a few layers to avoid over-smoothing. A generalized residual learning strategy presented in Ref. [32] is introduced for comparison. Due to the distinct structure from the classical residual network, it was designed to handle the over-smoothing issue of GNN. To evaluate the influence of the number of hypergraph convolutional layers comprehensively,

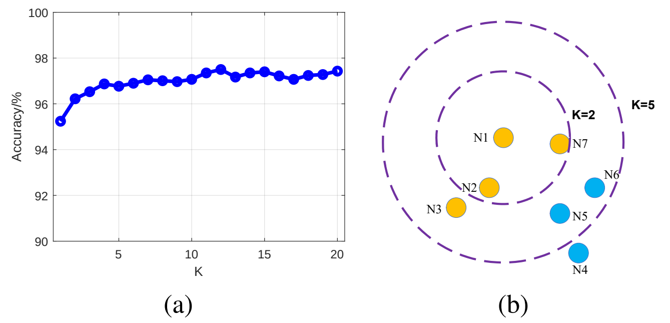


Fig. 14: Hyperparameter K influence on MsfHGNN (a) Experiment on Paderborn (b) Intuitive explanation

we redesign the HGNN with residual learning in MsfHGNN as shown in Fig. 15, and compare MsfHGNN and MsfHGNN with residual learning at various depths.

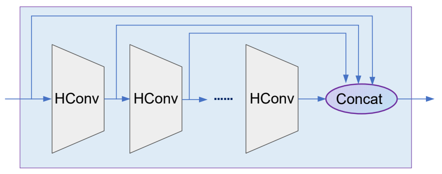


Fig. 15: The algorithm structure of HGNN with residual learning.

The experimental results are shown in Fig. 16. The accuracy of the algorithms only slightly improves with more layers, and does not depend on whether residual learning is used, and MsfHGNN with residual learning performs a little worse than MsfHGNN without residual learning. Despite a 1% gain in accuracy, both algorithms exhibit a considerable increase in training time as depth increases, with 6-layer HGNN taking twice the time as long as two-layer HGNN. From the perspective of balancing accuracy and training time, it is a reasonable choice to set the number of hypergraph convolutional layer to 2.

G. Influence of Noise Interference

The performance of the multi-sensor method suffers when the sensors are interfered with noise. In a multi-branch network, the branches with interfered signals as input are the only branches affected by noise, isolating the influence of noise on the other branches. Although some influences are presented during the fusion step, this network structure improves the algorithm's overall robustness. Therefore, the multi-sensor algorithms with multi-branch structure have strong anti-noise ability. However, for MsfHGNN, the dual-branch utilizes all sensor signals. In s-Hypergraph, a portion of the hyperedges are affected during creation if one sensor is noise-affected, but for m-Hypergraph, all hyperedges are affected. We evaluate the performance of several multi-branch algorithms and MsfHGNN to assess their anti-noise ability on Paderborn where one of the two sensors is set to interfere with noise.

Let $x = x_s + x_n$ as the signal interfered by noise in our study, where x_s and x_n are the raw signal and additional Gaussian noise respectively. Consider the signal-to-noise ratio (SNR) as a gauge of how much a signal is impacted by noise:

$$\mathrm{SNR} = 10\log(\frac{S}{N}) \tag{14}$$

where S and N are the average power of x_s and x_n.

According to the experimental findings in Fig. 17, MsfHGNN has anti-noise capability comparable to those of other multi-branch structure algorithms and can retain more than 80% accuracy until $\mathrm{SNR} = 5\mathrm{dB}$. There are primarily two reasons for this. First, since all hyperedges in m-Hypergraph are influenced by noise, the accuracy of m-HGNN declines sharply as SNR decreases. In s-Hypergraph, half of the hyperedges are created from SSSs taken by the sensor that is not impacted by noise, ensuring that the accuracy of s-HGNN is maintained at nearly 80%. Therefore, when MsfHGNN aggregates the two branches, it inherits the characteristics of s-HGNN and maintains sufficient performance under noise interference. Second, MB-HGNN outperforms other multi-branch structure algorithms, demonstrating that hypergraphs are more effective at expressing data correlations than simple graphs. Therefore, although only keeps a portion of hyperedges without noise interference, MsfHGNN can achieve performance comparable to other graph-based multi-branch algorithms.

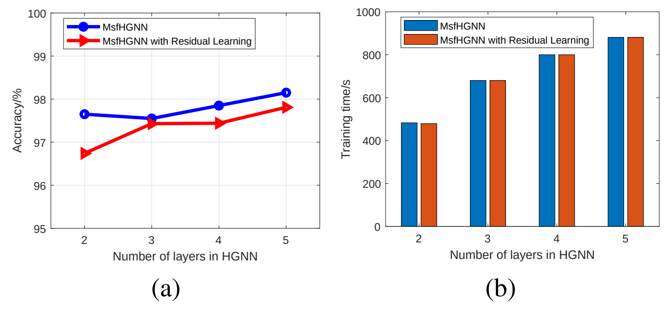


Fig. 16: Evaluation of the number of hypergraph convolutional layers on MsfHGNN.

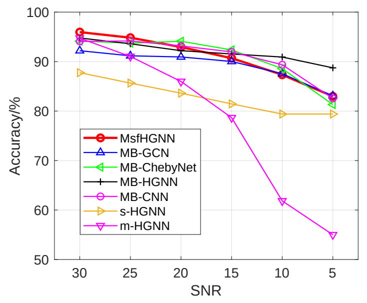


Fig. 17: Evaluation of anti-noise ability of algorithms on Paderborn.

H. Experiment Conclusions

Through a series of experiments, the following conclusions can be drawn.

1) Compared to other fusion algorithms, MsfHGNN acquires the high-order relationship between SSSs/MSSs and hence achieves the best accuracy in our experiments.

2) Due to the straightforward structure and parallel computation on the computer's GPU, the training time of MsfHGNN grows considerably much slower than that of other multi-branch structure algorithms as the number of sensors increases.
3) The two branches of MsfHGNN, s-HGNN and m-HGNN, capture diverse types of information, hence their performance differs on various datasets. MsfHGNN joins the two branches and strikes a balance in terms of accuracy, training time, and model complexity.
4) The K of KNN and the number of layers of HGNN affect the performance of MsfHGNN to a certain extent. The selection of the two hyperparameters requires a compromise between information loss and information redundancy, or between accuracy and training time.
5) When some sensors are interfered by noise, MsfHGNN can still guarantee a certain recognition accuracy because s-Hypergraph still retains a portion of the hyperedges that are not affected by noise.

V. Conclusion

In this paper, we propose MsfHGNN with a dual-branch network structure to achieve the fusion of multi-sensor data on two hypergraphs. Due to breaking the IID constraint, the suggested algorithm not only fuses the sensor information on the hypergraphs but also captures the correlation between SSSs or MSSs, such that the MsfHGNN algorithm obtains more sample information and achieves excellent results on both experimental datasets.

In future research, we will investigate applying this novel multi-sensor fusion framework to monitor machinery with heterogeneous sensors.

References

[1] Y. Guan, Z. Meng, D. Sun, J. Liu, and F. Fan, "2mnet: Multi-sensor and multi-scale model toward accurate fault diagnosis of rolling bearing," *Reliab. Eng. Syst. Safe.*, vol. 216, p. 108017, 2021.

[2] H. Wang, S. Li, L. Song, and L. Cui, "A novel convolutional neural network based fault recognition method via image fusion of multi-vibration-signals," *Comput. Ind.*, vol. 105, pp. 182–190, 2019.

[3] J. Tong, C. Liu, J. Zheng, and H. Pan, "Multi-sensor information fusion and coordinate attention-based fault diagnosis method and its interpretability research," *Eng. Appl. Artif. Intel.*, vol. 124, p. 106614, 2023.

[4] D. He, Z. Lao, Z. Jin, C. He, S. Shan, and J. Miao, "Train bearing fault diagnosis based on multi-sensor data fusion and dual-scale residual network," *Nonlinear Dynam.*, vol. 111, no. 16, pp. 14 901–14 924, 2023.

[5] Y. Tang, X. Zhang, S. Huang, G. Qin, Y. He, Y. Qu, J. Xie, J. Zhou, and Z. Long, "Multisensor-driven motor fault diagnosis method based on visual features," *IEEE Trans. Ind. Inform.*, vol. 19, no. 4, pp. 5902–5914, 2023.

[6] M. Jalayer, C. Orsenigo, and C. Vercellis, "Fault detection and diagnosis for rotating machinery: A model based on convolutional LSTM, fast fourier and continuous wavelet transforms," *Comput. Ind.*, vol. 125, p. 103378, 2021.

[7] J. Cui, P. Xie, X. Wang, J. Wang, Q. He, and G. Jiang, "M2FN: An end-to-end multi-task and multi-sensor fusion network for intelligent fault diagnosis," *Measurement*, vol. 204, p. 112085, 2022.

[8] J. Guo, Q. He, D. Zhen, F. Gu, and A. D. Ball, "Multi-sensor data fusion for rotating machinery fault detection using improved cyclic spectral covariance matrix and motor current signal analysis," *Reliab. Eng. Syst. Safe.*, vol. 230, p. 108969, 2023.

[9] M. S. Safizadeh and S. K. Latifi, "Using multi-sensor data fusion for vibration fault diagnosis of rolling element bearings by accelerometer and load cell," *Inform. Fusion*, vol. 18, pp. 1–8, 2014.

[10] H. Shao, J. Lin, L. Zhang, D. Galar, and U. Kumar, "A novel approach of multisensory fusion to collaborative fault diagnosis in maintenance," *Inform. Fusion*, vol. 74, pp. 65–76, 2021.

[11] X. Zhang, C. Sheng, W. Ouyang, and L. Zheng, "Fault diagnosis of marine electric thruster bearing based on fusing multi-sensor deep learning models," *Measurement*, vol. 214, p. 112727, 2023.

[12] Q. Zhong, E. Xu, Y. Shi, T. Jia, Y. Ren, H. Yang, and Y. Li, "Fault diagnosis of the hydraulic valve using a novel semi-supervised learning method based on multi-sensor information fusion," *Mech. Syst. Signal Pr.*, vol. 189, p. 110093, 2023.

[13] X. Yan, C.-A. Zhang, and Y. Liu, "Multi-branch convolutional neural network with generalized shaft orbit for fault diagnosis of active magnetic bearing-rotor system," *Measurement*, vol. 171, p. 108778, 2021.

[14] G. Niu, E. Liu, X. Wang, P. Ziehl, and B. Zhang, "Enhanced discriminate feature learning deep residual cnn for multitask bearing fault diagnosis with information fusion," *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 762–770, 2023.

[15] S. Bao, J. Feng, X. Xu, P. Hou, Z. Zhang, J. Meng, and F. Steyskal, "Multi-input parallel graph neural network for semi-supervised rolling bearing fault diagnosis," *Meas. Sci. Technol.*, vol. 34, no. 5, 2023.

[16] C. Yang, J. Liu, K. Zhou, X. Jiang, and X. Zeng, "An improved multi-channel graph convolutional network and its applications for rotating machinery diagnosis," *Measurement*, vol. 190, p. 110720, 2022.

[17] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[19] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou, "Hypergraph learning: Methods and practices," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 5, pp. 2548–2566, 2022.

[20] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007.

[21] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2019.

[22] S. Bai, F. Zhang, and P. H. S. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recogn.*, vol. 110, p. 107637, 2021.

[23] N. Yadati, P. Yadav, A. Louis, M. Nimishakavi, V. Nitin, and P. Talukdar, "Hypergcn: A new method of training graph convolutional networks on hypergraphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[24] Y. Gao, Y. Feng, S. Ji, and R. Ji, "HGNN+: General hypergraph neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3181–3199, 2023.

[25] L. Nong, J. Wang, J. Lin, H. Qiu, L. Zheng, and W. Zhang, "Hypergraph wavelet neural networks for 3d object classification," *Neurocomputing*, vol. 463, pp. 580–595, 2021.

[26] X. Hao, J. Li, Y. Guo, T. Jiang, and M. Yu, "Hypergraph neural network for skeleton-based action recognition," *IEEE Trans. Image Process.*, vol. 30, pp. 2263–2275, 2021.

[27] D. Di, C. Zou, Y. Feng, H. Zhou, R. Ji, Q. Dai, and Y. Gao, "Generating hypergraph-based high-order representations of whole-slide histopathological images for survival prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5800–5815, 2023.

[28] X. Yan, Y. Liu, and C.-A. Zhang, "Multiresolution hypergraph neural network for intelligent fault diagnosis," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022.

[29] M. Shi, C. Ding, R. Wang, Q. Song, C. Shen, W. Huang, and Z. Zhu, "Deep hypergraph autoencoder embedding: An efficient intelligent approach for rotating machinery fault diagnosis," *Knowl.-Based Syst.*, vol. 260, p. 110172, 2023.

[30] C. Lessmeier, J. Kimotho, D. Zimmer, and W. Sextro, "Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification," in *Eur. Conf. Progn. Health Manag. Soc.*, 2016.

[31] J. Zhu, X. Zhao, H. Hu, and Y. Gao, "Emotion recognition from physiological signals using multi-hypergraph neural networks," in *Proc. Int. Conf. Multi. Expo*, 2019.

[32] K. Xu, C. Li, Y. Tian, T. Sonobe, K. I. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. Int. Conf. Mach. Learn.*, 2018.