machines

MDPI

Article

Adaptive Dynamic Threshold Graph Neural Network: A Novel Deep Learning Framework for Cross-Condition Bearing Fault Diagnosis

Linjie Zheng [1], Yonghua Jiang [1,2,*], Hongkui Jiang [2,*], Chao Tang [2], Weidong Jiao [1], Zhuoqi Shi [1] and Attiq Ur Rehman [3]

[1] Key Laboratory of Intelligent Operation and Maintenance Technology and Equipment for Urban Rail Transit of Zhejiang Province, Zhejiang Normal University, Jinhua 321004, China; linjie_z99@zjnu.edu.cn (L.Z.); jiaowd1970@zjnu.cn (W.J.); shizhuoqi@zjnu.edu.cn (Z.S.)
[2] Xingzhi College, Zhejiang Normal University, Lanxi 321100, China; ethan95_tang@163.com
[3] School of Computer Science and Technology, Zhejiang Normal University, Jinhua 321004, China; atnutkani@zjnu.edu.cn
* Correspondence: yonghua_j82@zjnu.cn (Y.J.); jhkass@163.com (H.J.)

Abstract: Recently, bearing fault diagnosis methods based on deep learning have achieved significant success. However, in practical engineering applications, the limited labeled data and various working conditions severely constrain the widespread application of most deep-learning-based fault diagnosis methods. Additionally, many methods focus solely on the amplitude information of samples, neglecting the rich relational information between samples. To address these issues, this paper proposes a novel cross-condition few-shot fault diagnosis method based on an adaptive dynamic threshold graph neural network (ADTGNN). The aim of the proposed method is to rapidly identify fault types after they occur only a few times or even once. The adaptive threshold computation module (ATCM) in ADTGNN dynamically assigns thresholds to each edge based on edge confidence, optimizing the graph structure and effectively alleviating the over-smoothing issue. Furthermore, a dynamic threshold adjustment strategy (DTAS) is introduced to gradually increase the threshold with the training iterations, preventing the model from prematurely discarding crucial edges due to insufficient performance. The proposed model's effectiveness is demonstrated using three bearing datasets. The experimental results indicate that the proposed approach significantly outperforms other comparison methods in cross-condition bearing fault diagnosis.

Keywords: deep learning; bearing fault diagnosis; graph neural network; few-shot learning; cross-condition

Citation: Zheng, L.; Jiang, Y.; Jiang, H.; Tang, C.; Jiao, W.; Shi, Z.; Rehman, A.U. Adaptive Dynamic Threshold Graph Neural Network: A Novel Deep Learning Framework for Cross-Condition Bearing Fault Diagnosis. *Machines* **2024**, *12*, 18. https://doi.org/10.3390/machines12010018

Academic Editors: Amare Desalegn Fentaye and Konstantinos Kyprianidis

Received: 1 December 2023
Revised: 25 December 2023
Accepted: 26 December 2023
Published: 28 December 2023

1. Introduction

As crucial components in rotating machinery, bearings serve the primary function of transferring loads from moving machine parts to stationary ones, thereby facilitating the relative motion of the rotating components [1–5]. The healthy operation of bearings is paramount for ensuring the safe production of rotating machinery. However, in most cases, bearings are subjected to prolonged exposure in high-speed, heavy-load, and corrosive environments, posing a substantial threat to the health of the bearings and leading to potential failures [6]. Once a bearing fault occurs, it not only impacts the operation of the machinery itself but also exerts repercussions on subsequent production, triggering a chain reaction that can result in significant economic losses and even catastrophic accidents causing personnel casualties [7,8].

In recent years, with the rapid development of computer hardware technology, deep learning models have gradually become the mainstream method for bearing fault diagnosis due to their powerful feature extraction capabilities and end-to-end characteristics. Among them, convolutional neural networks [9,10], recurrent neural networks [11,12], and autoencoders [13,14] have received extensive attention from researchers. However, traditional

deep-learning-based methods for bearing fault diagnosis still face certain challenges. Conventional deep learning models typically require a large amount of labeled data for training and assume that the training and testing data are under the same working conditions. In practical industrial production environments, however, the working conditions of rotating machinery (such as speed and load) may change with variations in work tasks, making it difficult to ensure that training and testing data are under identical conditions. Moreover, to prevent personnel injuries and potential damage to the machinery, equipment is promptly shut down once a fault occurs, making it impractical to collect a large amount of labeled data for each working condition [15]. Therefore, achieving few-shot bearing fault diagnosis across different working conditions remains a challenge.

Currently, there are primarily three methods to address the aforementioned issues: data augmentation (DA), transfer learning (TL), and few-shot learning (FSL). DA involves enhancing the training data by applying diverse transformations and expansions to existing samples. Operations such as rotation, flipping, and scaling on images can generate additional training samples, thereby improving the model's generalization capability. Jiang et al. [16] proposed a novel generative adversarial-network-based ensemble data augmentation framework to address the issue of data imbalance in fault diagnosis. Wang et al. [17] introduced a new compressed sensing data augmentation method for bearing fault diagnosis. While this method has achieved some success in mitigating sample scarcity, the introduction of excessive noise through data augmentation may result in disparities between generated samples and real samples. TL leverages knowledge acquired from one task to enhance performance on another related task. This approach facilitates the transfer of knowledge across different domains, allowing for the more efficient utilization of limited samples. Huang et al. [18] proposed a deep adversarial capsule network designed for comprehensive fault diagnosis in industrial equipment. Zeng et al. [19] introduced a transfer-learning-based fault diagnosis approach that integrates nearest neighbor feature constraints, specifically tailored for addressing cross-condition fault diagnosis challenges. However, if the differences between two domains are substantial, a model trained on the source domain may struggle to adapt well to the data distribution of the target domain. FSL is specifically designed to address problems with limited sample quantities. It involves designing more complex model architectures, utilizing advanced optimization algorithms, or incorporating prior knowledge, all aimed at achieving better performance under restricted data conditions. Wang et al. [20] proposed a reinforced relation network for few-shot bearing fault diagnosis. Lei et al. [21] introduced a novel prior knowledge-embedded meta-transfer learning approach to address the challenges of few-shot fault diagnosis under varying conditions.

In recent years, graph neural networks (GNNs) have begun to be employed in the field of few-shot fault diagnosis. Wang et al. [22] propose a dual graph neural network to address fault diagnosis with limited data. Yang et al. [23] propose a graph contrastive learning framework for few-shot machine fault diagnosis. GNNs are a type of deep learning model specifically designed for processing graph data. Graph data consist of nodes and edges, where nodes represent samples and edges signify relationships between nodes. Traditional bearing fault diagnosis methods primarily extract information from the amplitude of bearing vibration signals, overlooking the rich relational information among samples. Transforming vibration signals into graph data allows for modeling these relational details. The application of GNNs enables the deep exploration of relationships between nodes, aggregating this information to learn expressive node representations. Given that few-shot learning involves leveraging relationships between support set samples and query set samples for classification, GNNs exhibit significant potential in addressing few-shot classification challenges.

The main concept of GNNs is to learn expressive node representations through the propagation of information. However, the existing research [24,25] indicates that this propagation method encounters certain issues. Firstly, the majority of GNNs suffer from the problem of over-smoothing. As the network layers and iteration counts increase, the

representations of all nodes tend to converge to a fixed point, rendering them independent of input features. Secondly, this propagation method makes each node highly dependent on its neighborhood, making nodes susceptible to potential data noise. Some approaches mitigate these issues through graph sparsification. Graph sparsification involves removing unnecessary edges in the graph to simplify its structure, enhancing model performance. Rong et al. [26] alleviate over-smoothing by randomly dropping certain edges during training. However, random dropping may discard edges carrying essential information. Xiao et al. [27] proposed an edge dropout threshold, removing edges with lower relevance in the graph to improve its quality. Yet, this threshold remains constant. Considering that the model faces varying difficulties in distinguishing different pairs of nodes, using a fixed threshold to determine whether to discard edges may impede the propagation of crucial information. Additionally, during the early stages of training, the model's performance is suboptimal, and for pairs of nodes that are challenging to differentiate, the model may struggle to provide high similarity scores.

In response to the aforementioned issues, this paper proposes an adaptive dynamic threshold graph neural network (ADTGNN) for achieving cross-condition few-shot bearing fault diagnosis. The proposed adaptive threshold computation module (ATCM) in ADTGNN adaptively assigns thresholds to each edge. The threshold setting is related to the confidence of the edge, reflecting the model's learning state regarding the nodes connected via the edge. This adaptive approach significantly reduces the risk of erroneously discarding edges carrying crucial information, greatly simplifying the graph and alleviating the over-smoothing issue. Additionally, we propose a dynamic threshold adjustment strategy (DTAS), where the constructed thresholds increase with the number of training iterations. This prevents the premature discarding of important edges during the early stages of training, thereby improving training speed and stability. The main contributions of this work are as follows:

1. To alleviate the over-smoothing issue in GNNs, an ATCM is proposed. This module adaptively sets thresholds based on the confidence level of edges, discarding edges with lower relevance and simplifying the structure of graph data.
2. A DTAS is proposed, where a relatively smaller threshold is assigned during the early stages of model training. With the increasing number of iterations, the threshold is gradually raised, preventing the premature discarding of important edges during the early stages of training.
3. An ADTGNN is proposed for cross-condition few-shot bearing fault diagnosis. By employing meta-learning strategies, the model rapidly achieves fault classification in the target working condition without the need for retraining, leveraging prior knowledge acquired from the known working condition.

The rest of the paper is organized as follows. In Section 2 we present the details of the proposed method. In Section 3, we validate the effectiveness of the proposed method on two public bearing datasets. In Section 4, we summarize the work conducted.

2. Proposed Method

2.1. Problem Definition

The proposed method in this paper addresses the issue of cross-condition few-shot bearing fault diagnosis. Let D_{known} denote the dataset composed of known working condition data, and D_{target} denote the dataset composed of target working condition data. D_{known} contains a large number of labeled samples, while D_{target} has only a limited number of labeled samples. The proposed method requires training on D_{known} to acquire prior knowledge and subsequently utilizes the small number of labeled samples from D_{target} to classify unlabeled samples within D_{target}. Specifically, we need to construct numerous few-shot classification tasks on D_{known}. Each few-shot classification task consists of a support set (a set of a few labeled samples) and a query set (a set of unlabeled samples). If a support set has N categories, each with K samples, the few-shot classification task is referred to as an *N-way K-shot* classification task. The model needs to learn how to

both utilize a small number of labeled samples for classifying unlabeled samples in these few-shot classification tasks and apply the learned experience to D_{target}.

Figure 1 illustrates the workflow of the proposed ADTGNN for cross-condition few-shot bearing fault diagnosis. ADTGNN is primarily composed of four modules: the graph initialization module (GIM), the node feature update module (NFUM), the edge feature update module (EFUM), and the ATCM. The details of these four modules will be elaborated in the following subsections.

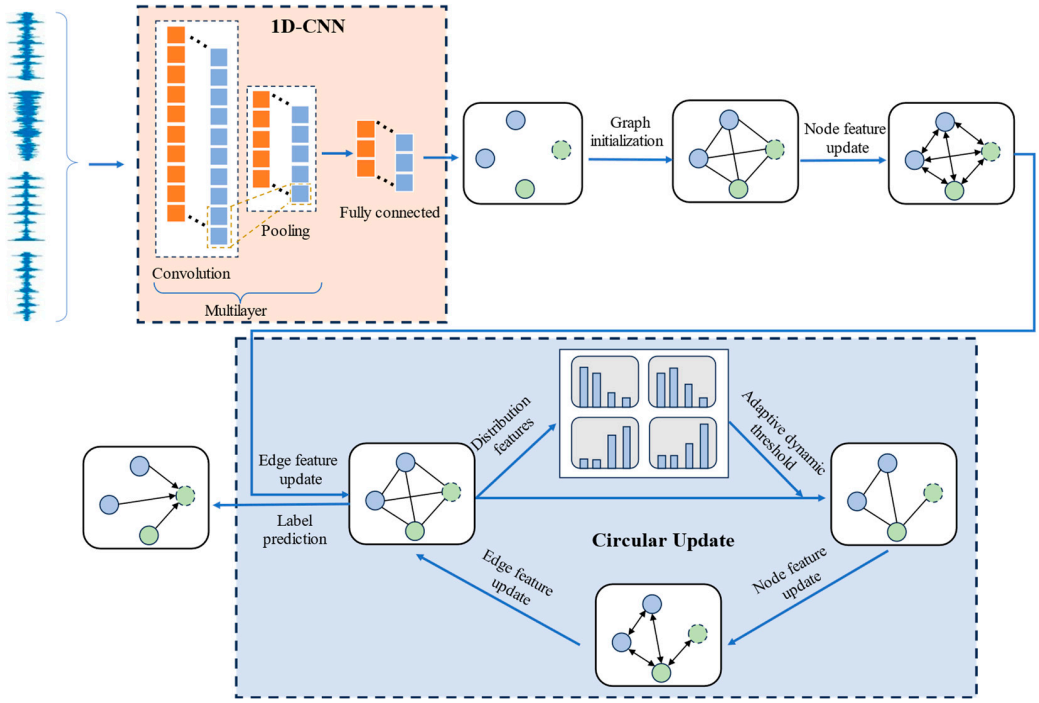

Figure 1. The workflow of the proposed ADTGNN.

2.2. Graph Initialization Module

Let $T = \mathrm{S} \cup \mathrm{Q}$ represent an *N-way K-shot* classification task, where $\mathrm{S} = \{x_i\}_{i=1,2,\ldots,N\times K}$ denotes the support set, $\mathrm{Q} = \{x_i\}_{i=N\times K+1,\ldots,N\times K+C}$ represents the query set, and C is the number of samples in the query set. Here, x_i represents a sample. A convolutional neural network is employed to extract initial node features v_i^0 from x_i, as shown in the following equation:

$$v_i^0 = f_{cnn}(x_i, \theta_{cnn}) \tag{1}$$

where $f_{cnn}(\cdot)$ comprises two convolutional blocks and one fully connected layer. Each convolutional block includes a convolutional layer, BatchNorm layer, LeakyReLU activation function layer, and MaxPool layer. θ_{cnn} represents the parameter set of $f_{cnn}(\cdot)$. To broaden the receptive field, a convolutional layer with a kernel size of 1×32 is applied in the first convolutional block. Let $e_{ij}^0 = \left(e_{ij1}^0, e_{ij2}^0\right)$ denote the initial edge features, where e_{ij1}^0 and e_{ij2}^0 represent similarity and dissimilarity, respectively. The setting process for e_{ij}^0 is expressed as follows:

$$e_{ij}^0 = \begin{cases} (1,0), \text{ if } y_i = y_j \text{ and } i,j \le N \times K, \\ (0,1), \text{ if } y_i \ne y_j \text{ and } i,j \le N \times K, \\ (0.5,0.5), \text{ otherwise.} \end{cases} \tag{2}$$

where y_i and y_j represent the labels of nodes i and j, respectively. Let $V = \left\{v_i^0\right\}_{i=1,2,\ldots,N\times K+C}$ denote the set of initial node features, and $E = \left\{e_{ij}^0\right\}_{i,j=1,2,\ldots,N\times K+C}$ denote the set of initial edge features. Based on V and E, a fully connected graph $G = (V, E)$ can be constructed and input into the following module for node feature updating and edge feature updating.

2.3. Node Feature Update Module

Based on the node features from the previous layer and the edges in the current layer, we can obtain the node features for the current layer, as shown below:

$$v_i^l = f_v\left(v_i^{l-1} || \sum_j \tilde{e}_{ij1}^{l-1} v_j^{l-1} || \sum_j \tilde{e}_{ij2}^{l-1} v_j^{l-1}, \theta_v\right) \tag{3}$$

$$\tilde{e}_{ij1} = \frac{e_{ij1}}{\sum_k e_{ik1}} \tag{4}$$

where $||$ denotes the concatenation operation, and $f_v(\cdot)$ represents the node feature updating network. $f_v(\cdot)$ consists of two convolutional blocks, each comprising a convolutional layer, BatchNorm layer, LeakyReLU activation function layer, and dropout layer, and θ_v is the parameter set of $f_v(\cdot)$. By setting a threshold for e_{ij1}^{l-1} and eliminating some edges, we can reduce the impact of redundant information during node feature updating. Additionally, during the update of v_i^l, both similarity-based aggregation information and dissimilarity-based aggregation information are considered, enhancing the expressive power of node features.

2.4. Edge Feature Update Module

Given the node features v_i^l and v_j^l at the l-th layer of the model, two metric networks, $f_{sim}(\cdot)$ and $f_{dsim}(\cdot)$, are employed to compute the similarity score s_{ij}^l and dissimilarity score d_{ij}^l. The process is outlined as follows:

$$s_{ij}^l = f_{sim}\left(v_i^l, v_j^l, \theta_{sim}\right) \tag{5}$$

$$d_{ij}^l = f_{dsim}\left(v_i^l, v_j^l, \theta_{dsim}\right) \tag{6}$$

where the network architecture of $f_{sim}(\cdot)$ is the same as that of $f_{dsim}(\cdot)$. Both consist of two convolutional blocks and a Sigmoid activation function layer. θ_{sim} and θ_{dsim} are the parameter sets of $f_{sim}(\cdot)$ and $f_{dsim}(\cdot)$, respectively. Based on s_{ij}^l and d_{ij}^l, the current layer's edge features e_{ij}^l can be obtained through the following equations:

$$\overline{e}_{ij1} = \frac{s_{ij}^l e_{ij1}^{l-1}}{\sum_k s_{ik}^l e_{ik1}^{l-1} / \left(\sum_k e_{ik1}^{l-1}\right)} \tag{7}$$

$$\overline{e}_{ij2} = \frac{d_{ij}^l e_{ij2}^{l-1}}{\sum_k d_{ik}^l e_{ik2}^{l-1} / \left(\sum_k e_{ik2}^{l-1}\right)} \tag{8}$$

$$e_{ij}^l = \overline{e}_{ij}^l / \left\|\overline{e}_{ij}^l\right\|_1 \tag{9}$$

2.5. Adaptive Threshold Computation Module

For ideal node features, we desire their similarity to be significantly higher with nodes of the same class than with nodes of different classes. However, in reality, there are always some nodes that are challenging for the model to accurately cluster, especially during the early stages of training. The values of the edges depend on the nodes they connect, implying that for these challenging nodes, the network may struggle to compute edges with larger values. Therefore, when using a fixed threshold to remove unnecessary edges, there is a risk of erroneously discarding these edges, thus reducing the transmission of useful information in the graph. The goal of this paper is to adaptively set corresponding thresholds for each edge, thereby minimizing the risk of discarding useful edges erroneously. We design edge

thresholds based on two principles. Firstly, the threshold setting should be related to the model's confidence in the edges, reflecting the model's learning state. Additionally, the confidence of edges should be related to the confidence of the nodes they connect.

For this purpose, before calculating the confidence of edges, it is necessary to obtain the confidence of the nodes. This paper utilizes the distribution features of nodes to measure their confidence. Given the distribution features of node i as $E_i^l \in R^{1\times D}$, where

$$E_i^l = \left(e_{i11}^l, e_{i21}^l, \ldots, e_{iD1}^l\right) \tag{10}$$

E_i^l is composed of the similarity between node i and the rest of the nodes. If the similarity of node i with the nodes of the same class is significantly higher than its similarity with nodes of different classes, calculating the entropy of E_i^l will result in a smaller value. Conversely, if the similarity of node i with nodes of the same class is not significantly higher than its similarity with the nodes of different classes, calculating the entropy of E_i^l will yield a larger value, as shown in Figure 2.

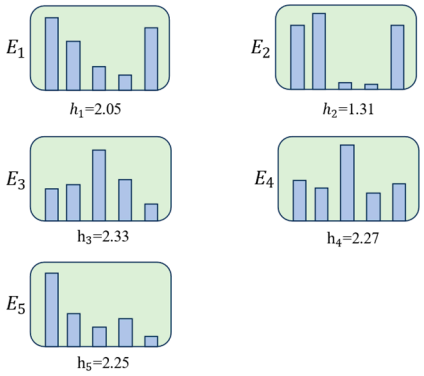

Figure 2. Entropy of different node distribution features.

Based on this principle, the confidence q_i^l of node i can be computed. First, utilize the following equation to calculate the entropy of E_i^l:

$$h_i^l = -\sum_j^C e_{ij1}^l \log\left(e_{ij1}^l\right) \tag{11}$$

where h_i^l represents the uncertainty of node i. Let $H^l = \left(h_1^l, h_2^l, \ldots, h_D^l\right)$, and normalize it to obtain $\tilde{H}^l$, as shown below:

$$\tilde{H}^l = \frac{H^l - Min\left(H^l\right)}{Max\left(H^l\right) - Min\left(H^l\right)} \tag{12}$$

$\tilde{H}_i^l \in (0,1)$ represents the i-th element of $\tilde{H}_i^l$. A higher value of $\tilde{H}_i^l$ indicates that node i is more challenging to distinguish, and its reliability is lower. The confidence q_i^l of node i is then obtained using the following equation:

$$q_i^l = 1 - \tilde{H}_i^l \tag{13}$$

With the confidence of the nodes given, we can now calculate the confidence of the edge C_{ij}^l, which depends on the confidences of nodes i and j:

$$C_{ij}^l = \frac{q_i^l + q_j^l}{2} \tag{14}$$

Based on C_{ij}^{l}, we can calculate the corresponding threshold p_{ij}^{l}, and the process is as follows:

$$p_{ij}^{l} = \tau C_{ij}^{l} \tag{15}$$

where τ represents the global threshold, a constant. Since $C_{ij}^{l} \in (0,1)$, p_{ij}^{l} will not exceed τ. C_{ij}^{l} changes with variations in E_{i}^{l} and E_{j}^{l}, enabling the adaptive adjustment of p_{ij}^{l} based on C_{ij}^{l}. The higher the value of C_{ij}^{l}, the greater the model's confidence in that edge, and p_{ij}^{l} becomes closer to τ.

2.6. Dynamic Adjustment Strategy of the Thresholds

During the early stages of training, the model's performance is relatively poor, making it challenging to accurately determine similarity scores between nodes. In such cases, adopting a high global threshold may lead to the removal of some useful edges, causing instability in the training process. To address this, we establish a dynamic threshold $\hat{p}_{ij}^{l}$, which steadily increases with the number of network iterations t, preventing the premature discard of useful edges during the early training phase, as shown below:

$$\alpha = -\delta^{-(t-1)} + 1 \tag{16}$$

$$\hat{p}_{ij}^{l} = \alpha p_{ij}^{l} \tag{17}$$

where $\delta > 1$ is a constant. Figure 3 illustrates the curve of α for different values of δ. It can be observed that a smaller value of δ leads to a smoother change in α. Due to the adaptive adjustment of p_{ij}^{l} based on E_{i}^{l} and E_{j}^{l}, $\hat{p}_{ij}^{l}$ will actually fluctuate and rise with the increase in t.

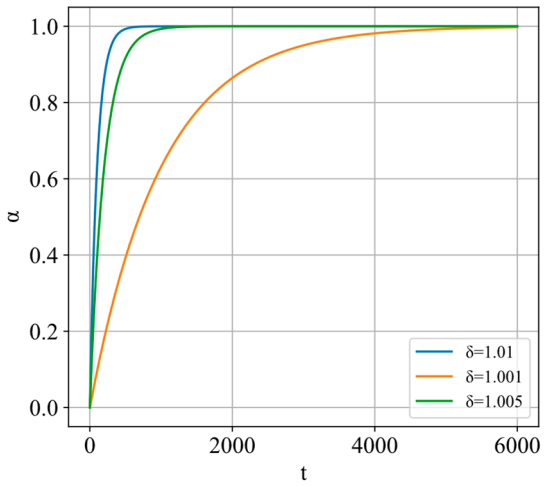

Figure 3. The curve of α for different values of δ.

This paper aims to discard edges below a threshold through adaptive thresholding while leaving edges above the threshold unchanged. So, if the value of e_{ij1}^{l} is below $\hat{p}_{ij}^{l}$, we consider the relationship between these two nodes to be insignificant and set it to 0. Conversely, no action is taken. The equation is expressed as follows:

$$e_{ij1}^{l} = \begin{cases} e_{ij1}^{l}, & e_{ij1}^{l} > \hat{p}_{ij}^{l} \\ 0, & e_{ij1}^{l} \leq \hat{p}_{ij}^{l} \end{cases} \tag{18}$$

2.7. Label Prediction and Loss Function

Through L rounds of iterative updates, the e_{ij}^{L} in the last layer of the network can serve as the final predicted edge label for the model. Let $\hat{y}_{ij} = e_{ij1}^{L}$, where $\hat{y}_{ij} \in [0,1]$, represent the probability that nodes i and j belong to the same category. By utilizing the predicted

edge labels between support set samples and query set samples, we can determine the predicted labels for the query set samples through weighted voting. Firstly, for node i, we need to compute the probability $p_i^{(k)}$ of it belonging to category k, as shown below:

$$p_i^{(k)} = softmax\left(\sum_{\{j:j\neq i \wedge (x_j,y_j)\}} \hat{y}_{ij}\delta\left(y_j = C_k\right)\right) \tag{19}$$

$$\delta\left(y_j = C_k\right) = \begin{cases} 0, \ if \ y_j = C_k \\ 1, \ otherwise \end{cases} \tag{20}$$

where C_k represents category k. Subsequently, we can calculate the predicted label $\hat{y}_i$ for node i using the following equation:

$$\hat{y}_i = argmax\left(\left[p_i^{(k)}\right]_{k=1,2,\ldots,N}\right) \tag{21}$$

where $argmax(\cdot)$ is a function that returns the index of the maximum value. Given the predicted labels and true labels for each layer of query set samples, ADTGNN is trained by minimizing the following loss function:

$$\mathcal{L} = \sum_{l=1}^{L} BCE\left(Y, \hat{Y}^l\right) \tag{22}$$

where Y represents the set of true labels for the query set, $\hat{Y}^l$ represents the set of predicted labels for the l-th layer of the query set samples, and $BCE(\cdot)$ denotes the binary cross-entropy loss function.

2.8. Fault Diagnosis Process

The fault diagnosis process of the proposed method is primarily divided into the following steps, as illustrated in Figure 4:

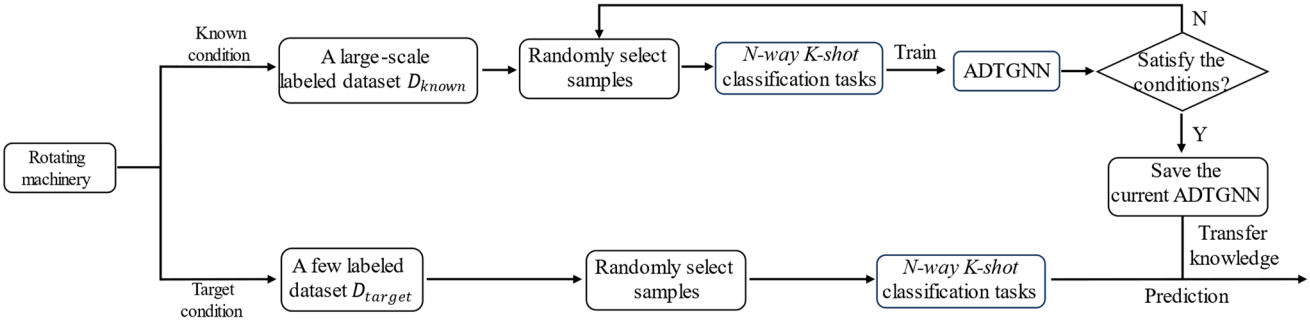

Figure 4. The fault diagnosis process of the proposed method.

Step 1: Initialize the network parameters of ADTGNN (θ_{cnn}, θ_v, θ_{sim}, etc.).

Step 2: Given a dataset D_{known} containing a large number of labeled samples under known working operating conditions, randomly select samples to form multiple few-shot classification tasks.

Step 3: Utilize ADTGNN to predict labels for the query set samples.

Step 4: Calculate the loss based on the predicted labels and true labels of the query set samples, and update the network parameters of ADTGNN using the gradient descent algorithm.

Step 5: If the training iterations reach the preset value, proceed to the next step; otherwise, repeat Steps 2 to 4.

Step 6: Employ the prior knowledge learned on D_{known} to achieve fault diagnosis on the target working condition dataset D_{target}, which contains only a small number of labeled samples.

3. Experimental Results and Analysis

To validate the effectiveness and to exhibit the superiority of the proposed ADTGNN, this paper conducted a series of comparison experiments using the Case Western Reserve University (CWRU) bearing dataset [28], the Paderborn University (PU) bearing dataset [29], and the drivetrain dynamics simulator (DDS) bearing dataset.

3.1. Dataset Introduction

The CWRU dataset is a publicly available dataset widely utilized in bearing fault diagnosis research, collected on the experimental platform illustrated in Figure 5. It comprises vibration signals from motors working at 0 horsepower (hp) (1797 rpm), 1 hp (1772 rpm), 2 hp (1750 rpm), and 3 hp (1730 rpm). For each working condition, the dataset includes data from normal conditions (NCs), as well as various types of fault states. The fault types primarily consist of inner race faults (IFs), outer race faults (OFs), and rolling element faults (RFs). Electrical discharge machining is employed to simulate faults, with each fault induced by three different fault diameters: 0.007, 0.014, and 0.021 inches. This study conducts experiments using vibration signals collected from the drive end, with a sampling frequency of 12 kHz and a signal length of 1024 for each sample. Table 1 provides a description of the working conditions used in this study.

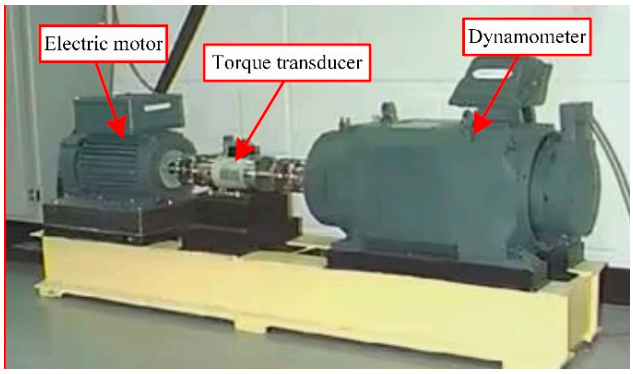

Figure 5. Rolling bearing fault detection platform [28].

Table 1. Details of working conditions in the CWRU dataset.

Dataset Name	Motor Loads (hp)	Speed (rpm)	Fault Type/Label
A	1 hp	1772	NC/0, RF/1–3,
B	2 hp	1750	IF/4–6,
C	3 hp	1730	OF/7–9

The PU dataset is collected on a modular experimental platform, as illustrated in Figure 6. This modular experimental platform consists of an electric motor, torque test shaft, rolling bearing test module, flywheel, and a load motor. By varying the speed of the drive system, the radial force on the bearing, and the load torque of the drive system, the PU dataset encompasses data from four working conditions. Table 2 provides detailed information on the three working conditions used in this study. The detailed parameters of the faulty bearings used in this study are presented in Table 3. The vibration signals used were measured via a piezoelectric accelerometer at the top adapter of the rolling bearing module, with a sampling frequency of 64 kHz and a signal length of 4096 for each sample.

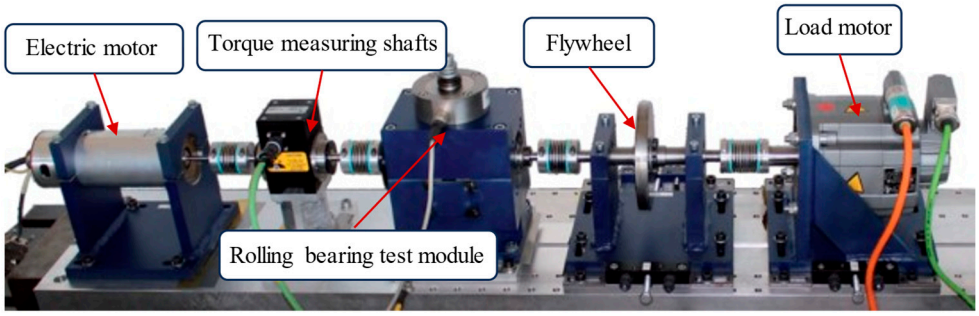

Figure 6. Modular experimental platform [29].

Table 2. Details of working conditions in the PU dataset.

Dataset Name	Load Torque (Nm)	Speed (rpm)	Radial Force (N)
D	0.1	1500	1000
E	0.7	1500	400
F	0.7	1500	1000

Table 3. Detailed information on faulty bearings in the PU dataset used in this study.

Bearing Code	Fault Type/LABEL	Extent of Damage	Damaged Length/Width (mm)
K001	NC/0	/	/
KA03	OF/1	2	3/2
KA06	OF/2	2	3/3
KI03	IF/3	1	<2/ <1
KI08	IF/4	2	3/1

The DDS dataset was collected on the drivetrain dynamics simulator of Spectra Quest, as illustrated in Figure 7. Acceleration sensors were used at the parallel gearbox to gather vibration signals of five different types of bearings at motor frequencies of 15 Hz, 20 Hz, and 25 Hz. These types include NC, IF, OF, RF, and a combination fault (CF). The sampling frequency was set at 20 kHz, and the signal length for each sample was 2048. Table 4 provides detailed information about the DDS dataset.

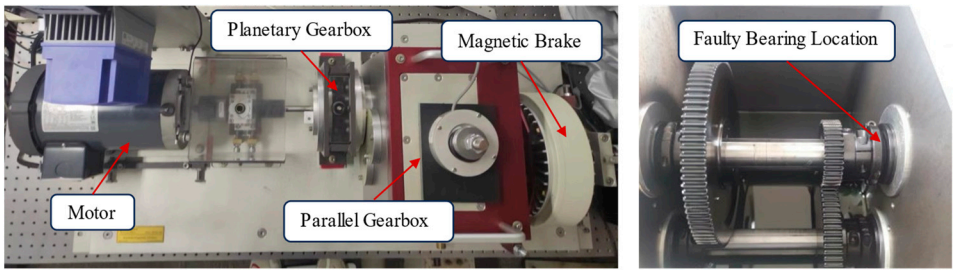

Figure 7. Drivetrain dynamics simulator.

Table 4. Details of working conditions in the DDS dataset.

Dataset Name	Motor Frequencies (Hz)	Fault Type/Label
G	15	NC/0, IF/1, OF/2, RF/3, CF/4
H	20	
I	25	

3.2. Experimental Setup

To validate the effectiveness of the proposed ADTGNN, this study constructed 18 cross-condition few-shot fault diagnosis tasks based on these three datasets. Each task requires the model to be trained on a known working condition dataset with a large number of labeled samples and tested on a target working dataset with only a few labeled samples. Five models were selected for comparison with the proposed ADTGNN: edge-labeling graph neural network (EGNN) [30], prototypical network (PN) [31], relation network (RN) [32], DenseNet [33], and ResNet [34]. EGNN serves as the baseline model for the ADTGNN, and PN and RN have been widely used in recent years to address cross-condition few-shot fault diagnosis problems. DenseNet and ResNet are two popular deep-learning-based classifiers. Among them, PN, RN, EGNN, and ADTGNN use the same structure for feature extractors. The network parameters for the ADTGNN are specified in Table 5. For the two parameters τ and δ in the ADTGNN, this study set them to 0.4 and 1.005, respectively.

Table 5. The network parameters for the ADTGNN.

Module Name	Configuration	
GIM	Conv1d	Channels: 16; kernel size: 1×32
	BN + LR	MaxPool1d (kernel size: 1×3)
	Conv1d	Channels: 32; kernel size: 1×3
	BN + LR	MaxPool1d (kernel size: 1×3)
NFUM	Conv2d	Channels: 128, kernel size: 1×1
	BN + LR	Dropout(p:0.2)
	Conv2d	Channels: 128, kernel size: 1×1
	BN + LR	Dropout(p:0.2)
EFUM	Conv2d	Channels: 128, kernel size: 1×1
	BN + LR	Dropout(p:0.2)
	Conv2d	Channels: 128, kernel size: 1×1
	BN + LR	Dropout(p:0.2)
ATCM + DTAS		

The code used in this study is implemented on the Pytorch platform, employing the Adam optimizer with an initial learning rate of 0.0001. The meta-training and meta-testing rounds are set to 200 and 50, respectively. The computational platform used consists of an Intel (R) Xeon (R) Silver 4210 CPU at 2.20 GHz and an NVIDIA GeForce RTX 2080Ti GPU. To mitigate the impact of randomness, each experiment is repeated five times, and the results are averaged.

3.3. Analysis of Experimental Results on the CWRU Dataset

In order to better showcase the superiority of the proposed model, we introduced Gaussian white noise with a signal-tonoise ratio (SNR) of -6 dB into the original vibration signals from the CWRU dataset. The formula for calculating the SNR is as follows:

$$\mathrm{SNR} = 10lg\frac{P_s}{P_n} \tag{23}$$

where P_s represents the power of the original signal, and P_n represents the power of the signal with added noise. As the intensity of the added noise increases, the SNR decreases. Taking the example of the inner race fault under 1 hp conditions, Figure 8 illustrates the time-domain plots of the original vibration signal and the signal with added Gaussian white noise at -6 dB. It can be observed that after adding the noise, the fault features are obscured, significantly increasing the difficulty of fault diagnosis.

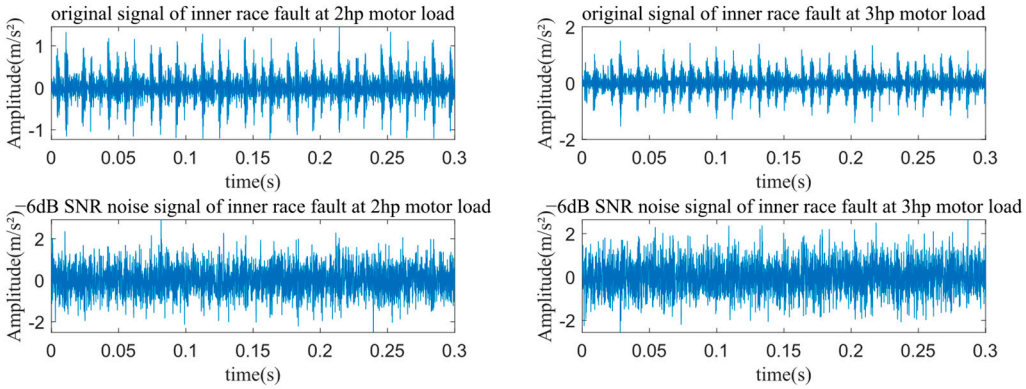

Figure 8. A comparison of the original signal and the signal with added noise under different loads.

Table 6 and Figure 9 display the experimental results of various models on the CWRU dataset. In Table 6, A→B indicates that the model was trained on dataset A with a large number of labeled samples and tested on dataset B with only a few labeled samples. It can be seen that the accuracy of the proposed ADTGNN is consistently higher than that of other comparative models across six cross-condition fault diagnosis tasks. Under the 1-shot condition, ADTGNN achieves an average accuracy of 96.45%, surpassing EGNN by 2.83%. Under the 3-shot condition, ADTGNN demonstrates an average accuracy of 97.93%, exceeding EGNN by 2.26%. These findings highlight the superior performance of ADTGNN. From Figure 9c, it can be observed that the average accuracy of various few-shot learning models under 3-shot conditions is higher than that under 1-shot conditions. This is well understood, as the more samples each class has, the more information the model can learn about that class. Under the 1-shot condition, the average accuracies of DenseNet and ResNet are significantly lower than the other four few-shot learning models. Especially in the C→A task, the accuracy of DenseNet is only 85.12%, and ResNet achieves an accuracy of only 85.41%. The reason for this may be that traditional deep learning training strategies hinder the models from acquiring better general experience, making it challenging to handle fault diagnosis tasks across different operating conditions.

Table 6. Experimental results on the CWRU dataset (add −6 dB Gaussian white noise).

Classification Task	Model	Fault Diagnosis Task						
		A→B	**A→C**	**B→A**	**B→C**	**C→A**	**C→B**	**Average**
/	DenseNet	88.81%	92.62%	88.70%	91.42%	85.12%	86.24%	88.82%
	ResNet	92.45%	91.24%	90.45%	92.61%	85.41%	90.83%	90.50%
10-way 1-shot	PN	91.21%	90.79%	93.91%	90.61%	91.79%	89.60%	91.33%
	RN	91.32%	92.14%	94.46%	91.15%	93.56%	92.59%	92.54%
	EGNN	95.26%	92.17%	96.03%	92.24%	93.98%	92.05%	93.62%
	ADTGNN	96.89%	95.24%	97.13%	95.18%	97.31%	96.93%	96.45%
10-way 3-shot	PN	93.64%	92.80%	95.68%	93.20%	93.79%	90.59%	93.28%
	RN	93.8%	93.41%	97.39%	95.02%	95.41%	94.79%	94.97%
	EGNN	96.73%	94.83%	97.71%	95.78%	95.56%	93.38%	95.67%
	ADTGNN	98.10%	96.76%	98.31%	97.38%	98.87%	98.18%	97.93%

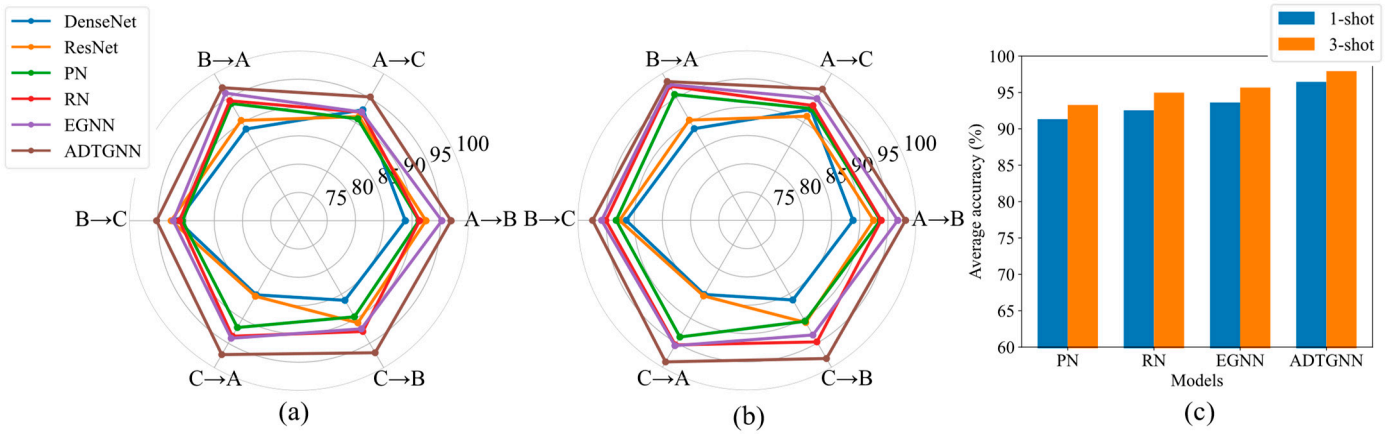

Figure 9. The accuracy of various models on the CWRU dataset. (**a**) The accuracy of each model under the 1-shot condition; (**b**) the accuracy of each model under the 3-shot condition; (**c**) comparison of average accuracy of few-shot learning models.

3.4. Analysis of Experimental Results on the PU Dataset

Table 7 and Figure 10 present the experimental results of various models on the PU dataset. It can be observed that under both 1-shot and 3-shot conditions, ADTGNN outperforms the other comparative models significantly. For instance, under the 1-shot condition, ADTGNN achieves accuracies of 97.01% and 97.51% in the D→E and F→E tasks, respectively, surpassing the second best by 2.47% and 6.08%. It is worth noting that under both 1-shot and 3-shot conditions, the average accuracy of PN and RN is lower than that of DenseNet and ResNeRN. The reason for this phenomenon may be that when there is a significant difference in the data distribution between the source and target domains, the relatively simple feature extractor structure of PN and RN prevents them from learning better general knowledge. From Figure 10b and Figure10a, we observe that the performance of various models in the D→E and F→E tasks is lower compared to the other four tasks. For example, under the 1-shot condition, EGNN achieves 100% accuracy in the D→F task but only 94.54% and 91.43% accuracy in the D→E and F→E tasks, respectively. Referring to Table 2, the probable reason for this is that changes in radial force have a significant impact on the probability distribution of bearing vibration signals. This leads to a substantial difference between known and target working condition data in the D→E and F→E tasks, resulting in poorer performance across all models. However, even under these conditions, the proposed ADTGNN outperforms the baseline EGNN, demonstrating the effectiveness of the improvements introduced in this paper.

Table 7. Experimental results on the PU dataset.

Classification Task	Model	Fault Diagnosis Task						
		D→E	D→F	E→D	E→F	F→D	F→E	Average
/	DenseNet	92.83%	100%	98.41%	99.60%	99.66%	94.49%	97.50%
	ResNet	93.24%	100%	100%	100%	100%	92.8%	97.67%
5-way 1-shot	PN	79.98%	97.95%	93.58%	95.18%	90.75%	77.99%	89.24%
	RN	84.78%	98.71%	97.98%	98.80%	98.30%	79.59%	93.03%
	EGNN	94.54%	100%	100%	100%	100%	91.43%	97.66%
	ADTGNN	97.01%	100%	100%	100%	100%	97.51%	99.09%
5-way 3-shot	PN	94.79%	100%	97.51%	96.73%	99.59%	89.19%	96.30%
	RN	85.18%	100%	99.12%	99.51%	100%	81.97%	94.29%
	EGNN	95.12%	100%	100%	100%	100%	94.54%	98.28%
	ADTGNN	98.42%	100%	100%	100%	100%	98.82%	99.54%

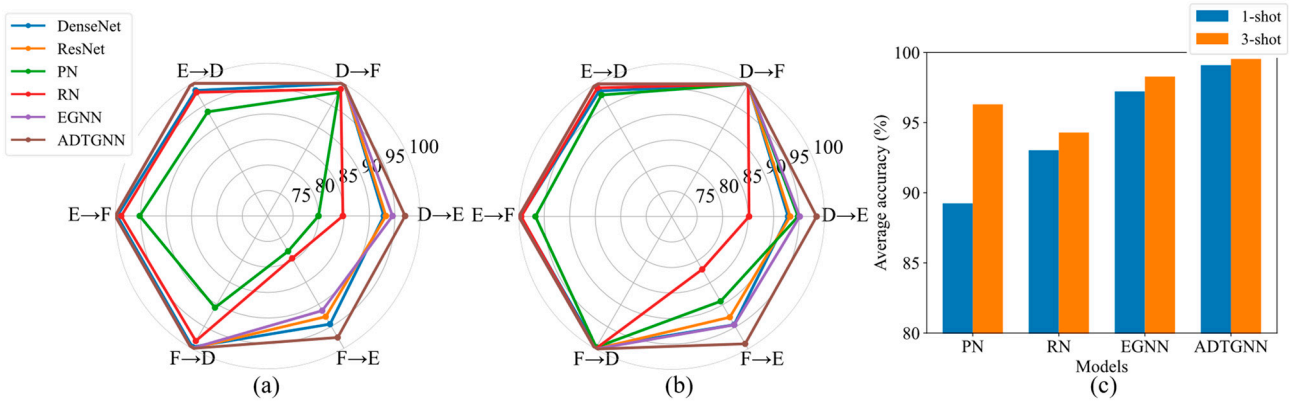

Figure 10. The accuracy of various models on the PU dataset. (**a**) The accuracy of each model under the 1-shot condition; (**b**) the accuracy of each model under the 3-shot condition; (**c**) comparison of average accuracy of few-shot learning models.

3.5. Analysis of Experimental Results on the DDS Dataset

Table 8 and Figure 11 display the experimental results of various models on the DDS dataset. It can be observed that under 1-shot and 3-shot conditions, the proposed ADTGNN outperforms the other comparison models across all six tasks. Under 1-shot conditions, ADTGNN achieves an average accuracy of 94.20% across the six tasks, which is 5.5% higher than EGNN. Under 3-shot conditions, ADTGNN achieves an average accuracy of 96.25%, surpassing EGNN by 4.02%. This can be attributed to the proposed ATCM and DTAS, which enable ADTGNN to adaptively set thresholds, thereby discarding some irrelevant edges and improving accuracy. From Figure 11a, it can be observed that the performance of various models on the G→H and G→I tasks is relatively poor. The reason for this may be the significant distribution difference between dataset G and dataset H, resulting in the knowledge learned by the model in the source domain not effectively transferring to the target domain, especially in cases with small sample sizes. It is observed that under the 1-shot condition, the average accuracy of PN is only 82.03%, which is lower than DenseNet with 83.58% and ResNet with 84.47%. The potential reason for this could be that when there is only one sample per class, PN struggles to derive effective class prototypes, leading to relatively poorer performance. Moreover, ADTGNN maintains relatively high accuracy under the 1-shot condition. This is attributed to the transformation of one-dimensional vibration signals into a graph, enabling ADTGNN to effectively leverage relational information among samples during classification.

Table 8. Experimental results on the DDS dataset.

Classification Task	Model	Fault Diagnosis Task						
		G→H	G→I	H→G	H→I	I→G	I→H	Average
/	DenseNet	88.41%	78.45%	86.01%	90.12%	73.66%	84.83%	83.58%
	ResNet	86.40%	74.41%	87.66%	94.76%	70.76%	92.83%	84.47%
5-way 1-shot	PN	77.59%	73.54%	76.79%	92.31%	77.13%	94.79%	82.03%
	RN	81.59%	78.79%	85.10%	93.59%	80.79%	92.39%	85.38%
	EGNN	82.95%	83.60%	85.43%	98.24%	86.62%	95.35%	88.70%
	ADTGNN	90.73%	93.58%	88.19%	98.86%	96.69%	97.13%	94.20%
5-way 3-shot	PN	86.79%	82.33%	87.63%	95.19%	83.14%	94.81%	88.32%
	RN	83.49%	80.99%	88.79%	97.99%	83.19%	95.59%	88.34%
	EGNN	88.73%	85.96%	89.72%	99.03%	92.77%	97.18%	92.23%
	ADTGNN	93.89%	96.50%	92.81%	99.12%	97.15%	98.04%	96.25%

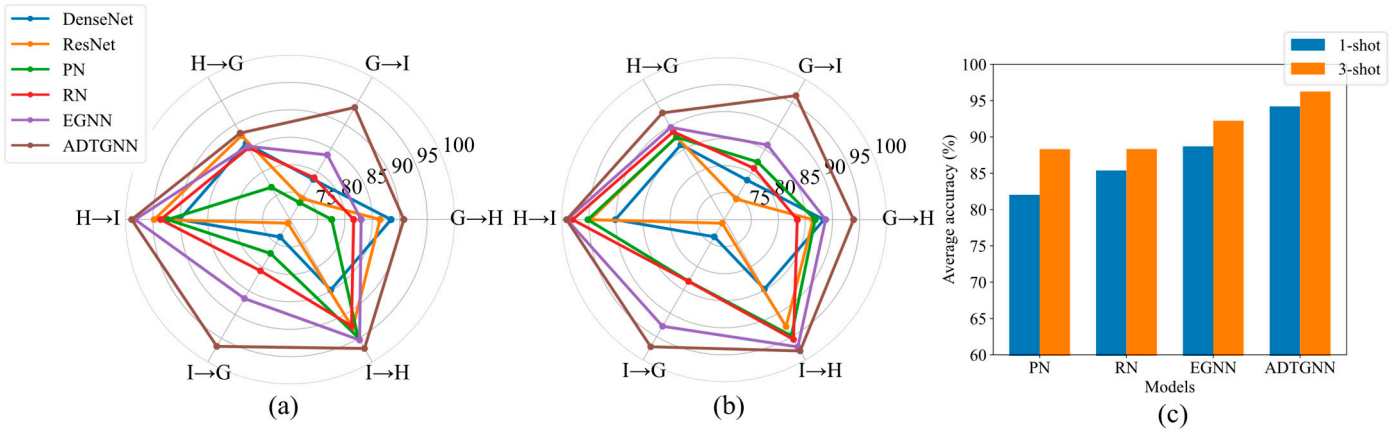

Figure 11. The accuracy of various models on the DDS dataset. (**a**) The accuracy of each model under the 1-shot condition; (**b**) the accuracy of each model under the 3-shot condition; (**c**) comparison of average accuracy of few-shot learning models.

3.6. Analysis of Over-Smoothing Issue

Chen et al. [35] proposed that the over-smoothing issue in GNNs may result from the excessive mixing of information and noise. In GNNs, interactions between nodes can bring useful information or irrelevant noise. For instance, interactions between nodes of the same class bring valuable information, making their representations closer to each other. Conversely, interactions between the nodes of different classes introduce noise, leading to learned representations that are indistinguishable. As the number of network layers increases, the acquired noise gradually outweighs the useful information, resulting in the over-smoothing problem. To validate the effectiveness of the proposed ATCM and DTAS in addressing over-smoothing issues, this study investigated the performance of EGNN and ADTGNN with different model layers in the challenging tasks of D→E and F→E. The experimental results under the 1-shot condition are presented in Table 9 and Figure 12.

Table 9. The accuracy of EGNN and ADTGNN under different model layers.

Fault Diagnosis Task	Model	Model Layer				
		1	2	3	4	5
D→E	EGNN	91.7%	95.28%	90.87%	88.34%	84.49%
	ADTGNN	95.48%	97.01%	96.65%	95.07%	94.23%
F→E	EGNN	82.65%	91.43%	87.26%	84.24%	82.13%
	ADTGNN	93.93%	97.51%	96.82%	95.71%	94.13%

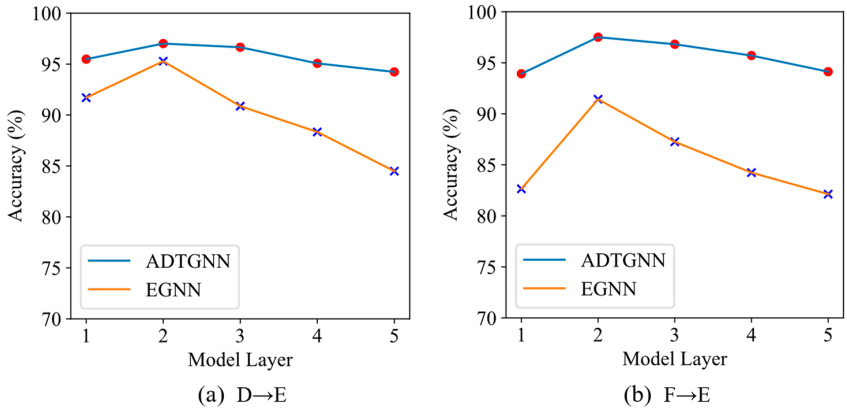

Figure 12. Accuracy comparison of EGNN and ADTGNN under different model layers. (**a**) Task D→E; (**b**) Task F→E.

It can be observed that for tasks D→E and F→E, when the model has two layers, both EGNN and ADTGNN achieve their highest accuracy. As the number of model layers increases, there is a varying degree of decrease in accuracy for EGNN and ADTGNN. However, compared to EGNN, ADTGNN exhibits a more gradual decline. For instance, in the D→E task, when the model has five layers, ADTGNN's accuracy only decreases by 2.78% compared to its peak accuracy. In contrast, EGNN's accuracy drops by 10.79%. This indicates that the proposed strategy in this paper effectively mitigates over-smoothing issues.

3.7. Selection of Parameter

In ADTGNN, a critical parameter, the global threshold τ, has a substantial impact on the model's performance. The choice of τ directly affects the upper limit of the adaptive threshold learned by the model. To explore the performance of ADTGNN under different τ values, we conducted experiments under the 1-shot condition, focusing on the challenging tasks of D→E and F→E. In these experiments, ADTGNN had a model depth of two layers, and the δ value was set to 1.005. The results of the experiments are depicted in Table 10 and Figure 13.

Table 10. The accuracy of ADTGNN under different τ values.

Fault Diagnosis Task	The Value of Global Threshold τ				
	0.1	**0.2**	**0.3**	**0.4**	**0.5**
D→E	94.01%	93.93%	94.71%	97.01%	93.35%
F→E	93.09%	93.38%	95.45%	97.51%	94.34%

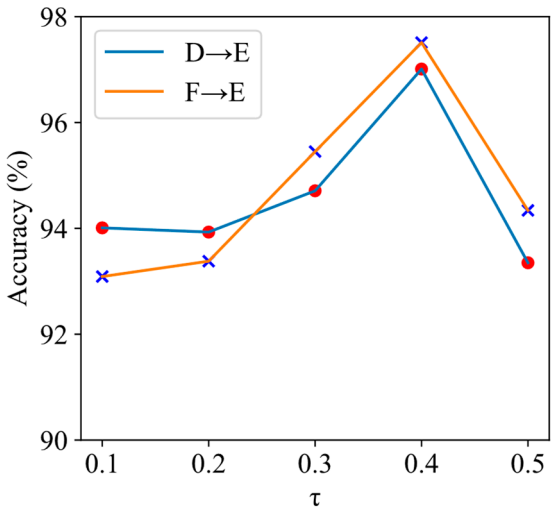

Figure 13. Performance of ADTGNN under different τ values.

It can be observed that ADTGNN performs best when $\tau = 0.4$. An excessively high τ may lead to the erroneous discarding of some useful edges, while an excessively low τ may result in the ineffective filtering of irrelevant edges.

3.8. Ablation Experiment

To assess the impact of the proposed ATCM and DTAS on the overall performance of the model, this study conducted ablation experiments on four challenging tasks. Here,

ADTGNN_1 indicates that the model does not use ATCM and DTAS. ADTGNN_2 indicates the model with a fixed threshold of 0.3 and without DTAS, while ADTGNN_3 indicates that the model only uses ATCM and not DTAS, with a τ value set to 0.4. The experimental results are presented in Table 11. It can be observed that ADTGNN_3 exhibits significantly higher accuracy across all four tasks compared to ADTGNN_2. This is attributed to the proposed ATCM, which adaptively sets thresholds for each edge based

on confidence, reducing the likelihood of erroneously discarding important edges. To demonstrate the effectiveness of DTAS more distinctly, this paper plotted the testing accuracy curves for ADTGNN_3 and ADTGNN in a specific experiment in Figure 14. As seen in Figure 14, ADTGNN achieves higher accuracy more rapidly than ADTGNN_3, particularly in tasks C→A and F→E. This is because during the early stages of training, the model's performance may be insufficient, and setting high thresholds could result in erroneously discarding useful edges, leading to training instability. DTAS, by constructing dynamic thresholds, effectively enhances training stability. Finally, it can be observed that ADTGNN_1 exhibits the poorest performance among the four models. In summary, we can conclude that both the proposed ATCM and DTAS contribute to the overall performance of the model and are indispensable.

Table 11. The results of the ablation experiment.

Model	Fault Diagnosis Task			
	C→A	**C→B**	**D→E**	**F→E**
ADTGNN_1	95.26%	92.17%	94.54%	91.43%
ADTGNN_2	95.67%	92.92%	94.73%	94.14%
ADTGNN_3	96.15%	94.56%	96.12%	96.37%
ADTGNN	96.89%	95.24%	97.01%	97.51%

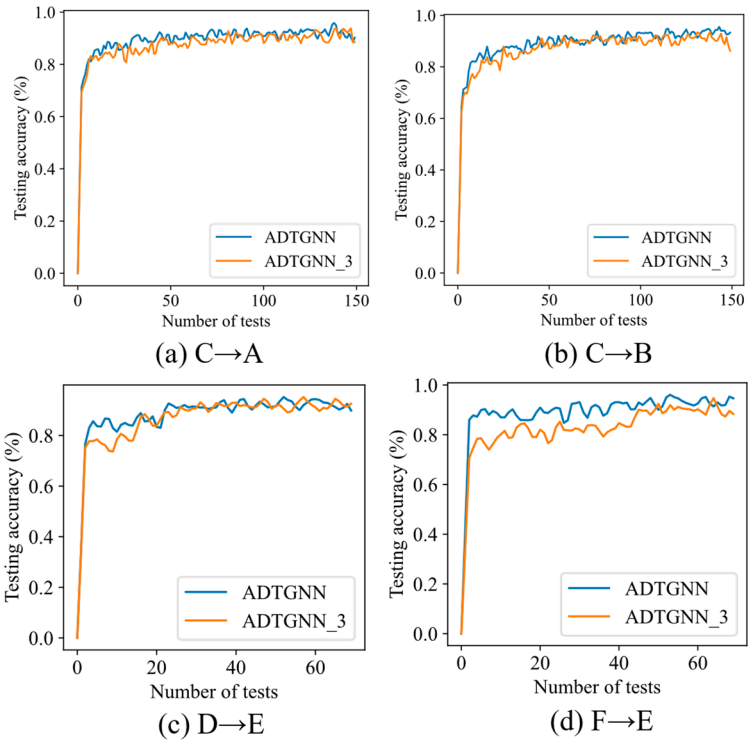

Figure 14. The testing accuracy curves for ADTGNN_3 and ADTGNN. (**a**) Task C→A; (**b**) Task C→B; (**c**) Task D→E; (**d**) Task F→E.

4. Conclusions

This paper proposes a novel cross-condition few-shot fault diagnosis method based on ADTGNN. ADTGNN is primarily composed of four modules: the GIM, the NFUM, the EFUMand the ATCM. Firstly, the one-dimensional vibration signal is transformed into a graph using the GIM. The graph is then input into the NFUM and the EFUM for feature transformation. The ATCM dynamically assigns thresholds to each edge based on its confidence, adapting and optimizing the graph structure, thereby effectively alleviating the over-smoothing issue in the graph. Furthermore, this paper proposed a DTAS, creating

a dynamic threshold that gradually increases with the number of training iterations. This approach aims to prevent the model from prematurely discarding crucial edges in the early stages of training due to insufficient performance. Finally, fault diagnosis is achieved through a weighted voting mechanism. The proposed method's superiority is validated through the construction of 18 cross-condition few-shot fault diagnosis tasks on three bearing datasets.

However, in order to achieve fault diagnosis across different working conditions, ADTGNN still relies on collecting a small number of labeled samples for each fault in advance. In future work, we will endeavor to address this issue.

Author Contributions: Conceptualization, L.Z. and Y.J.; methodology, L.Z. and Y.J.; software, L.Z.; validation, L.Z., Y.J. and H.J.; formal analysis, H.J.; investigation, C.T.; resources, W.J.; data curation, Z.S.; writing—original draft preparation, L.Z.; writing—review and editing, L.Z.; visualization, A.U.R.; supervision, Y.J.; project administration, Y.J. and H.J.; funding acquisition, Y.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (grant numbers: 51405449 and 51575497) and the Zhejiang Provincial Natural Science Foundation of China (grant number: LZ22E050001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Acknowledgments: The authors give thanks to CWRU and PU for providing this research with the open experiment data of rolling bearing faults.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Pu, H.; Zhang, K.; An, Y. Restricted Sparse Networks for Rolling Bearing Fault Diagnosis. *IEEE Trans. Ind. Inf.* **2023**, *19*, 11139–11149. [CrossRef]
2. Zhang, X.; Wang, H.; Ren, M.; He, M.; Jin, L. Rolling Bearing Fault Diagnosis Based on Multiscale Permutation Entropy and SOA-SVM. *Machines* **2022**, *10*, 485. [CrossRef]
3. Mikic, D.A.; Desnicab, E.; Asonjac, A.; Stojanovicd, B.; Epifanic-Pajic, V. Reliability Analysis of Ball Bearing on the Crankshaft of Piston Compressors. *J. Balk. Tribol. Assoc.* **2016**, *22*, 2060–5070.
4. Patil, A.A.; Desai, S.S.; Patil, L.N.; Patil, S.A. Adopting Artificial Neural Network for Wear Investigation of Ball Bearing Materials Under Pure Sliding Condition. *Appl. Eng. Lett.* **2022**, *7*, 81–88. [CrossRef]
5. Vasić, M.; Stojanović, B.; Blagojević, M. Failure Analysis of Idler Roller Bearings in Belt Conveyors. *Eng. Fail. Anal.* **2020**, *117*, 104898. [CrossRef]
6. Xiong, J.; Liu, M.; Li, C.; Cen, J.; Zhang, Q.; Liu, Q. A Bearing Fault Diagnosis Method Based on Improved Mutual Dimensionless and Deep Learning. *IEEE Sens. J.* **2023**, *23*, 18338–18348. [CrossRef]
7. Lin, T.; Zhu, Y.; Ren, Z.; Huang, K.; Gao, D. CCFT: The Convolution and Cross-Fusion Transformer for Fault Diagnosis of Bearings. *IEEE/ASME Trans. Mechatron.* **2023**, 1–12. [CrossRef]
8. Meng, Z.; Luo, C.; Li, J.; Cao, L.; Fan, F. Research on Fault Diagnosis of Rolling Bearing Based on Lightweight Model with Multiscale Features. *IEEE Sens. J.* **2023**, *23*, 13236–13247. [CrossRef]
9. Zhang, K.; Li, Z.; Zheng, Q.; Ding, G.; Tang, B.; Zhao, M. Fault Diagnosis with Bidirectional Guided Convolutional Neural Networks Under Noisy Labels. *IEEE Sens. J.* **2023**, *23*, 18810–18820. [CrossRef]
10. Wang, X.; Hua, T.; Xu, S.; Zhao, X. A Novel Rolling Bearing Fault Diagnosis Method Based on BLS and CNN with Attention Mechanism. *Machines* **2023**, *11*, 279. [CrossRef]
11. Zhang, Y.; Zhou, T.; Huang, X.; Cao, L.; Zhou, Q. Fault Diagnosis of Rotating Machinery Based on Recurrent Neural Networks. *Measurement* **2021**, *171*, 108774. [CrossRef]
12. Kim, H.; Lee, H.; Kim, S.; Kim, S.W. Attention Recurrent Neural Network-Based Severity Estimation Method for Early-Stage Fault Diagnosis in Robot Harness Cable. *Sensors* **2023**, *23*, 5299. [CrossRef] [PubMed]
13. Zhang, Z.; Yang, Q.; Zi, Y.; Wu, Z. Discriminative Sparse Autoencoder for Gearbox Fault Diagnosis Toward Complex Vibration Signals. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 3522611. [CrossRef]
14. Yu, S.; Wang, M.; Pang, S.; Song, L.; Zhai, X.; Zhao, Y. TDMSAE: A Transferable Decoupling Multi-Scale Autoencoder for Mechanical Fault Diagnosis. *Mech. Syst. Signal Process.* **2023**, *185*, 109789. [CrossRef]

15. Li, F.; Wang, L.; Wang, D.; Wu, J.; Zhao, H. Transfer Multiscale Adaptive Convolutional Neural Network for Few-Shot and Cross-Domain Bearing Fault Diagnosis. *Meas. Sci. Technol.* **2023**, *34*, 125002. [CrossRef]
16. Jiang, X.; Zheng, J.; Zhuang, X.; Ge, Z. Ensemble Data Augmentation for Imbalanced Fault Diagnosis. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 3528312. [CrossRef]
17. Wang, D.; Dong, Y.; Wang, H.; Tang, G. Limited Fault Data Augmentation with Compressed Sensing for Bearing Fault Diagnosis. *IEEE Sens. J.* **2023**, *23*, 14499–14511. [CrossRef]
18. Huang, R.; Li, J.; Liao, Y.; Chen, J.; Wang, Z.; Li, W. Deep Adversarial Capsule Network for Compound Fault Diagnosis of Machinery Toward Multidomain Generalization Task. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 3506311. [CrossRef]
19. Zeng, M.; Li, S.; Li, R.; Li, J.; Xu, K.; Li, X. A Transfer-Learning Fault Diagnosis Method Considering Nearest Neighbor Feature Constraints. *Meas. Sci. Technol.* **2023**, *34*, 015114. [CrossRef]
20. Wang, S.; Wang, D.; Kong, D.; Wang, J.; Li, W.; Zhou, S. Few-Shot Rolling Bearing Fault Diagnosis with Metric-Based Meta Learning. *Sensors* **2020**, *20*, 6437. [CrossRef]
21. Lei, Z.; Zhang, P.; Chen, Y.; Feng, K.; Wen, G.; Liu, Z.; Yan, R.; Chen, X.; Yang, C. Prior Knowledge-Embedded Meta-Transfer Learning for Few-Shot Fault Diagnosis under Variable Operating Conditions. *Mech. Syst. Signal Process.* **2023**, *200*, 110491. [CrossRef]
22. Wang, H.; Wang, J.; Zhao, Y.; Liu, Q.; Liu, M.; Shen, W. Few-Shot Learning for Fault Diagnosis with a Dual Graph Neural Network. *IEEE Trans. Ind. Inf.* **2023**, *19*, 1559–1568. [CrossRef]
23. Yang, C.; Liu, J.; Xu, Q.; Zhou, K. A Generalized Graph Contrastive Learning Framework for Few-Shot Machine Fault Diagnosis. *IEEE Trans. Ind. Inf.* **2023**, 1–10. [CrossRef]
24. Li, Q.; Han, Z.; Wu, X.-M. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2018.
25. Zhao, L.; Akoglu, L. PairNorm: Tackling Oversmoothing in GNNs. ICLR'20. *arXiv* **2020**, arXiv:1909.12223.
26. Rong, Y.; Huang, W.; Xu, T.; Huang, J. Dropedge: Towards Deep Graph Convolu—Tional Networks on Node Classification. ICLR'20. *arXiv* **2020**, arXiv:1907.10903.
27. Xiao, X.; Li, C.; Huang, J.; Yu, T.; Wong, P.K. An Improved Graph Convolutional Networks for Fault Diagnosis of Rolling Bearing with Limited Labeled Data. *Meas. Sci. Technol.* **2023**, *34*, 125109. [CrossRef]
28. Case Western Reserve University Bearings Vibration Dataset. Available online: http://csegroups.case.edu/bearingdatacenter/home (accessed on 12 November 2022).
29. Lessmeier, C.; Kimotho, J.K.; Zimmer, D.; Sextro, W. 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 *European Conference of the Prognostics and Health Management Society*; Prognostics and Health Management Society: Utrecht, The Netherlands, 2016.
30. Kim, J.; Kim, T.; Kim, S.; Yoo, C.D. Edge-Labeling Graph Neural Network for Few-Shot Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019.
31. Snell, J.; Swersky, K.; Zemel, R.S. Prototypical Networks for Few-Shot Learning. *arXiv* **2017**, arXiv:1703.05175.
32. Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P.H.S.; Hospedales, T.M. Learning to Compare: Relation Network for Few-Shot Learning. *arXiv* **2018**, arXiv:1711.06025.
33. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. *arXiv* **2018**, arXiv:160806993.
34. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
35. Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; Sun, X. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Association for the Advancement of Artificial Intelligence: Washington, DC, USA, 2020.