GRAPH NEURAL NETWORK APPLIED TO BEARING FAULT DIAGNOSIS

Luiza Scapinello Aquino da Silva
Electrical Engineering Graduate Program (PPGEE), Federal University of Parana (UFPR), Curitiba, Brazil
luiza.scapinello@ufpr.br

Alan Lopes
Mechanical Engineering Graduate Program (PPGEM), Pontifical Catholic University of Paraná (PUCPR), Curitiba, PR, Brazil
alan.lopes@ntn.com.br

Laio Oriel Seman
Department of Automation and Systems Engineering, Federal University of Santa Catarina (UFSC), Florianopolis, SC, Brazil
laioseman@gmail.com

Viviana Cocco Mariani
Mechanical Engineering Graduate Program (PPGEM), Pontifical Catholic University of Paraná (PUCPR), and
Department of Electrical Engineering, Federal University of Paraná (UFPR), Curitiba, PR, Brazil
viviana.mariani@pucpr.br

Leandro dos Santos Coelho
Electrical Engineering Graduate Program (PPGEE), Federal University of Parana (UFPR), Curitiba, PR, Brazil, and Industrial and Systems Engineering Graduate Program (PPGEPS), Pontifical Catholic University of Parana (PUCPR)
leandro.coelho@pucpr.br

Abstract. *Effective bearing fault diagnosis is crucial for the operational efficiency of industrial machinery, ensuring enhanced performance and minimized maintenance costs. Recent advancements highlight graph neural networks (GNNs) as a potent tool for fault diagnosis, attributed to their capacity to model intricate interdependencies within complex data. This paper introduces a novel GNN-based methodology for bearing fault diagnosis, utilizing the comprehensive Case Western Reserve University (CWRU) bearing dataset. The vibration signals of bearings are represented as graphs, where each bearing is depicted as a node and the vibration signals determine node feature. Such representation captures both spatial and temporal dependencies among the bearings and their associated fault conditions. The proposed methodology employs a modified StemGNN with Laplacian regularization, and a deep equilibrium model at the output. Designed for time-series forecasting, this model adeptly handles both the spatial and temporal intricacies of the data. To validate its robustness, the performance of the GNN is compared against traditional Long Short-Term Memory (LSTM) models and classical CGN. Preliminary results suggest that GNNs demonstrate promising capabilities in bearing fault diagnosis. This study offers insights for both academia and industry on the potential of GNNs in predictive maintenance strategies.*

Keywords: *Graph Neural Network, Bearing Fault Diagnosis, Case Western Reserve University bearing dataset.*

1. INTRODUCTION

The role of effective bearing fault diagnosis in maintaining the operational efficiency of industrial machinery cannot be overstated. Predicting and diagnosing bearing faults timely and accurately contributes significantly to the overall performance and lifespan of machinery, thereby enhancing productivity and reducing maintenance costs (Hoang and Kang, 2019). In the era of Industry 4.0, machine learning and deep learning have emerged as prominent approaches in the field of fault diagnosis, exhibiting considerable promise in enhancing the efficiency of predictive maintenance strategies (Zhang *et al*., 2020).

In this landscape, graph neural networks (GNNs) have shown potential due to their ability to model complex data as graphs, therefore capturing intricate interdependencies within the data (Wu *et al.*, 2022). This capability can be particularly beneficial for bearing fault diagnosis, given the multidimensional nature of vibration signals and their inherent dependencies that hold significant diagnostic information (Zhu, Liu, and Tan, 2023).

The crux of this paper centers on exploring the power of GNNs in bearing fault diagnosis, using the widely recognized Case Western Reserve University (CWRU) bearing dataset as the base for the study. This dataset encompasses vibration signals, obtained via accelerometers positioned on the bearing housings, from four bearings under diverse operating conditions including normal functioning, inner race fault, outer race fault, and roller fault. The signals have been sampled at a frequency of 12 kHz and are complemented by labels indicating the health condition of each bearing at various time steps, thereby allowing for supervised training of GNN models.

Numerous studies have leveraged the CWRU dataset to explore everything from traditional statistical methods to cutting-edge deep learning techniques, solidifying its status as a foundational resource in the field of machinery fault diagnosis. Zhang *et al.* (2015) developed an integrated permutation entropy, ensemble empirical mode decomposition, and support vector machine bearing defect diagnostic model. The first paper employing CNN to identify bearing fault using the CWRU dataset was the work of Guo *et al.* (2016) which used a softmax classifier to predict fault size. Deep belief network is also vastly used in this domain (Liang *et al.,* 2018). Papers using autoencoders demonstrate the usefulness and efficiency of using this model to serve as defect diagnosis tools (Shao *et al.,* 2018). Long Short-Term Memory (LSTM) is also applied in the CWRU dataset in the work of Nacer *et al.* (2023)

Much of the existing research in this domain has yet to fully exploit the multidimensional nature of vibration signals, which hold a wealth of diagnostic information. Addressing this gap, this paper introduces a pioneering approach by representing the vibration signals of the widely recognized CWRU bearing dataset as graphs. In doing so, both spatial and temporal intricacies among bearings and their respective fault conditions are ingeniously captured. This novel representation, combined with a custom Graph Convolutional Recurrent Network, aims to push the boundaries of GNN applications in bearing fault diagnosis, presenting a potential paradigm shift in the field. The work not only showcases the versatility of GNNs but also sets the stage for a more nuanced and effective approach to predictive maintenance in the realm of industrial machinery.

A comprehensive experiment on the CWRU bearing dataset was conducted to evaluate the performance of GNNs in bearing fault diagnosis. The results showcase the effectiveness of GNNs in this domain, achieving a satisfactory performance in the chosen metric. The implications of these findings provide profound insights into the capabilities of GNNs for bearing fault diagnosis and serve as a valuable resource for both researchers and industry professionals.

The remaining of the paper is organized as follows. The fundamentals of GNN are presented in Section 2. Sequentially, Section 3 grants the adopted methodology and explains more about the dataset. In Section 4, the results are described and discussed. Lastly, the conclusion and future scopes of research are given in Section 5.

2. GRAPH NEURAL NETWORKS

A GNN is a powerful machine learning framework for processing data structured as graphs. GNNs generalize CNNs to deal with graph-structured data and have achieved remarkable performance in various tasks such as node classification (Hang *et al.,* 2021), link prediction (Zhang and Chen, 2018), and graph classification (Wu *et al.,* 2021).

Formally, let $G = (V, E)$ be a graph, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of nodes and $E \subset V \times V$ is the set of edges. Associated with each node $v_i \in V$ is a feature vector $x_i \in \mathbb{R}^d$. A GNN processes the graph by iteratively updating node embeddings $h_i^{(t)} \in \mathbb{R}^p$, where t denotes the iteration or layer of the GNN. The updated embeddings are computed based on the node's own features and the embeddings of its neighbors:

$$h_i^{(t+1)} = \mathrm{ReLU}\left(W^{(t)} \cdot \mathrm{AGGREGATE}^{(t)}\left(h_j^{(t)} : v_j \in N(v_i)\right) + b^{(t)}\right) \tag{1}$$

where $N(v_i)$ is the set of neighbors of v_j, $W^{(t)} \in \mathbb{R}^{p \times p}$ and $b^{(t)} \in \mathbb{R}^p$ are learnable parameters at layer t, and $\mathrm{AGGREGATE}^{(t)}$ is an aggregation function that combines the embeddings of neighboring nodes. Commonly used aggregation functions include sum, mean, and max.

The GNN operates for T iterations, and the final node embeddings are used for downstream tasks such as node classification:

$$z_i = \mathrm{softmax}\left(W^{(T)} \cdot h_i^{(T)} + b^{(T)}\right) \tag{2}$$

The model parameters are learned by minimizing a loss function $\mathcal{L}(z_i, y_i)$, where y_i is the ground truth label associated with node v_i.

2.1 Graph Convolutional Operator

In order to perform the message passing, different kinds of convolution can be employed; here we describe the graph convolution operator. Mathematically, let A be the adjacency matrix of the graph, and let X be a matrix where each row is the feature vector of a node. The operation of a GCN layer can then be written in the form:

$$\tilde{A} = A + \mathrm{I} \tag{3}$$

where I is the identity matrix that adds self-loops to the graph. Then,

$$D = \text{diag}\left(\tilde{A} \cdot \mathbf{1}\right) \tag{4}$$

where $\mathbf{1}$ is a vector of ones and $\text{diag}(\cdot)$ turns a vector into a diagonal matrix. Matrix D is the degree matrix of $\tilde{A}$. Then we calculate the normalized adjacency matrix:

$$\hat{A} = D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}} \tag{5}$$

Finally, the output Z of a GCN layer is given by:

$$Z = \hat{A}XW \tag{6}$$

where W is a matrix of trainable weights, this equation shows that the new feature vector for each node is a weighted sum (where the weights are given by the adjacency matrix $\hat{A}$) of its neighbors' feature vectors, transformed by W. This operation can be stacked multiple times to create a deep GCN, with each layer potentially transforming the nodes' features differently. This allows the GCN to capture complex patterns in the graph structure.

From a node-wise perspective, the update rule can be written as

$$h_i^{(l+1)} = \sigma\left(b^{(l)} + \sum_{j\in N(i)} \frac{1}{c_{ji}} h_j^{(l)} W^{(l)}\right) \tag{7}$$

where $h_i^{(l+1)}$ represents the feature vector of node i at layer $l+1$, σ is the activation function, $b^{(l)}$ is the bias term for layer l. The term $\sum_{j\in N(i)}$ denotes summation over the neighbors j of node i, c_{ji} is a normalization constant for the edge between nodes i and j, and $h_j^{(l)}$ is the feature vector of node j at layer l, and$W^{(l)}$ is the weight matrix for layer l.

3. METHODOLOGY

This chapter explores our specialized implementation based on the Spectral Temporal Graph Neural Network (StemGNN) architecture. StemGNN was originally designed to address the challenges of multivariate time-series forecasting. Unlike traditional approaches that primarily focus on capturing temporal correlations in the time domain and often rely on pre-defined priors for inter-series relationships, StemGNN takes a unique approach. It captures both inter-series correlations and temporal dependencies simultaneously but does so in the spectral domain.

StemGNN achieves this by combining Graph Fourier Transform (GFT) and Discrete Fourier Transform (DFT) in an end-to-end framework. GFT is used to model the inter-series correlations, while DFT is used to handle temporal dependencies. After passing through these Fourier transforms, the spectral representations of the data hold clear patterns that are more effectively predicted by convolutional and sequential learning modules. One of the key advantages of StemGNN is its ability to learn inter-series correlations automatically from the data, eliminating the need for pre-defined priors. The original authors have validated the effectiveness of StemGNN through extensive experiments on ten real-world datasets.

In our implementation, we introduce a custom Laplacian regularization to the graph convolutional layer, further improving the model's capability to learn and generalize the graph structure. The output of this layer is the customized Laplacian matrix and learned attention weights that represent the relationships between different nodes in the graph.

After passing through the Fourier transforms and capturing the spectral representations, our model feeds the output into a Deep Equilibrium Model (DEQ) fully connected network at the output stage. The DEQ allows the network to reach an equilibrium state for each layer, which aids in more effective training and inference.

The code for our customized StemGNN model is implemented using PyTorch, and we employ various hyperparameter tuning strategies. These include options for the number of stacks, the number of layers in the multilayer perceptron, dropout rate, leaky rate, and learning rate. Our model utilizes binary cross-entropy with logits as the loss function and employs the Adam optimizer for training. To validate the performance of our modified StemGNN model, we conduct multiple runs with different hyperparameter configurations.

The employed network is then compared to LSTM (Yu *et al.*, 2019) and classical CGN. In essence, the latest defines and trains a model that uses both LSTM and GCN layers to process graph-structured data, guided by an attention mechanism (Niu *et al.*, 2021) that determines the relative importance of the LSTM and GCN outputs.

3.1 CWRU Dataset

The CWRU dataset is a well-liked, freely available, and open-source dataset. On the CWRU website[1], which offers access to the bearing data for both healthy and unhealthy bearings, the created dataset is saved and made available. Data were gathered for normal bearings, single-point drive-end (DE), and fan-end (FE) faults and entered this database. The CWRU bearing dataset is used as the baseline and the core dataset to verify the effectiveness of various machine learning and deep learning algorithms (Smith and Randall, 2015).

A torque transducer, a dynamometer, a two horsepower Reliance electric induction motor, and control electronics make up the bearing test rig setup that was used to gather the CWRU dataset. The motor shaft is supported by the test bearings. An electronic control system and dynamometer are used to provide torque to the shaft. The rolling element bearings' inner-race (IR) and outer-race (OR) flaws were seeded, and each defective bearing was reinstalled on the test rig. The test bearings were subjected to electro-discharge machining to introduce single point faults with fault sizes of 7, 14, 21, 28, and 40 mils. Apart from three bearings—an outer-race faulty bearing with a diameter of 0.040 inches, an inner-race faulty bearing with a diameter of 0.028 inches, and a ball bearing fault with a diameter of 0.011 inches — all of the bearings had a fault depth of 0.011 inches. Both the 0.028-inch inner-race defective bearing and the 0.040-inch outer-race faulty bearing have a fault depth of 0.050 inches. In addition, it is discovered that the depth of the 0.028-inch-diameter ball bearing flaw is 0.150 inches.

The dataset consists of 161 records, which are grouped into four classes: 48k normal-baseline, 48k drive-end fault, 12k drive-end fault, and 12k fan-end fault. In this study the 12k fan-end fault was used. The outer race faults are further divided into three categories based on their relationship to the load zone: "centered" (6.00 o'clock position), "orthogonal" (3.00 o'clock), and "opposite" (12.00 o'clock). The first letter in the file names for the data files indicates the location of the fault, the next three digits indicate the diameters of the defect, and the final number indicates the bearing loads.

3.2 Hyperparameters

To train the presented model, the dataset was split into training and validation, 0.8/0.2 split. The Adam optimizer was used as the training optimizer and Binary Cross Entropy with Logits as the loss function. The hyperparameters presented in Table 1 were adopted.

Table 1. Our model's hyperparameters.

Hyperparameter	**Value**
Epochs	500
Batch Size	128
Hidden Size	50
Dropout	0.2
Learning Rate	0.0001
Leaky Rate	0.01
Number of Multi Layers	2
Window Size	5

3.3 Binary Cross Entropy with Logits

The metric used to classify and test the data is the Binary Cross Entropy with Logits, which combines a Sigmoid layer and the Binary Cross Entropy loss (BCELoss) in one single class (Ruby and Yendapalli, 2020.). This version is more numerically stable than using a plain Sigmoid followed by a BCELoss because of the log-sum-exp trick which can help avoid numerical underflow or overflow problems. The formula for Binary Cross Entropy with Logits is given by:

$$loss = target * -\log(\text{sigmoid}(input)) + (1 - target) * -\log(1 - \text{sigmoid}(input)) \tag{8}$$

where $target$ is the true label (0 or 1), $input$ are the raw predictions of the model, the sigmoid function squashes the input to the range (0, 1), making it possible to interpret the output as a probability. The loss function then applies the sigmoid function to transform these raw scores into probabilities, and then calculates the binary cross-entropy loss. This loss function is useful when you are dealing with imbalanced datasets, as it supports weight inputs, meaning it is possible to assign more importance to under-represented classes.

[1] https://csegroups.case.edu/bearingdatacenter/home

4. RESULTS AND DISCUSSION

The tests made compared our methodology with the usage of only LSTM, and GCN for the classification of the thirteen possible labels for the faults. The results with the minimum, maximum, standard deviation, and mean values for 30 runs of the algorithm are presented in Table 2, for both test and validation phases. The values in bold are the best results for the validation in both cases and the values underlined are the best results for the test trial.

Table 2. Results after 30 runs.

Approach	Dataset	Minimum	Maximum	Standard Deviation	Mean
CGNN	Test	0.0006	<u>0.0862</u>	<u>0.0029</u>	<u>0.0052</u>
	Validation	0.2096	**0.2660**	**0.0211**	**0.2430**
GNN	Test	0.6398	0.6777	0.0089	0.6531
	Validation	0.0438	1.0451	0.3638	0.3598
LSTM	Test	<u>0.0004</u>	0.2326	0.0695	0.1195
	Validation	**0.0023**	1.1599	0.4481	0.5947

The results suggest that the CGNN model displayed the most consistent performance between the test and validation sets. On the other hand, both the GNN and LSTM models showed significant differences between the test and validation results, with the validation results displaying high variability. It suggests potential overfitting issues with these models, necessitating further tuning to improve their generalization capabilities.

When comparing the performance of different models, it is essential to consider not only their mean results but also the range of these results and their consistency. In these respects, the CGNN clearly outperformed the GNN and LSTM models.

Although the CGNN had better performance in the test, it still outperformed the GNN and LSTM in the validation, demonstrating the model's capacity to generalize well from the training data to unseen data. This is a significant attribute for any machine learning model, as it is critical for reliable real-world application. The GNN and LSTM models, on the other hand, had large differences between their test and validation means, indicating that these models may not generalize as well.

The range of results, as indicated by the minimum and maximum values, is another crucial factor to consider. For the CGNN, this range was tight for both the test (0.0006 to 0.0862) and validation (0.2096 to 0.2660) sets, suggesting that the model is robust and stable, producing consistently high-quality results. This contrasts with the GNN and LSTM models, which exhibited a wide range of values, especially in the validation set. Such wide ranges suggest these models' performance can be highly variable, reducing their reliability.

The standard deviation, which measures the variability of a dataset, further supports the superior performance of the CGNN. Both the test and validation sets had low standard deviations (0.0029 and 0.0211 respectively), indicating that the CGNN's results are reliably close to the mean. Conversely, the GNN and LSTM models had much larger standard deviations in their validation results, suggesting that their performance could be inconsistent.

Finally, the CGNN’s high test performance indicates that it was able to capture the underlying patterns in the data effectively. Its similarly high validation performance suggests that it was not merely memorizing the training data but rather learning transferable patterns. This capacity for meaningful learning, rather than rote memorization, is a key indicator of a quality model and separates the CGNN from the GNN and LSTM models in our evaluation.

The comparison between LSTM and GNN models revealed interesting insights. In terms of the test runs, the LSTM model exhibited better results, suggesting its efficacy in capturing patterns and making accurate predictions. However, during the validation trial, the GNN model demonstrated its superiority and showcased a higher level of power and consistency.

One way to measure this consistency is by considering the standard deviation. The GNN model displayed a smaller standard deviation compared to the LSTM model, indicating that its predictions were more reliable and less prone to variations. This consistency in performance implies that the GNN model could potentially be more robust and dependable in real-world scenarios.

The LSTM model also encountered an overfitting problem, as evidenced by the disparity between its performance on the test data and the validation data. Overfitting occurs when a model becomes too specialized in capturing the patterns present in the training data, leading to a reduced ability to generalize and make accurate predictions on unseen or validation data. In this case, the LSTM model performed well on the test data but struggled when faced with new, unseen data during validation.

In contrast, the GNN model did not exhibit the same overfitting issue. Its performance remained relatively consistent across both test and validation datasets, suggesting that it was able to generalize well and adapt to unseen patterns effectively.

Upon comparing the two models, it becomes evident that the GNN model outperforms the LSTM model in terms of mean performance on the test set. However, the LSTM model scores higher on the validation set, albeit with more inconsistency in results. This disparity could be attributed to differences in the nature of the two datasets or could point towards overfitting during the model training phase.

Therefore, while the GNN model shows a higher level of accuracy and consistency on the test dataset, the LSTM model seems to generalize better on the validation set. Further analysis and modifications in the training process could potentially address the observed overfitting and improve overall model performance.

On Figure 1 it is possible to see how each model performs in every epoch, based on the Loss function. Remarkably, there seems to be no evident progression in the performance of the GNN over time. The CGNN continues to outshine with superior performance, closely followed by LSTM.

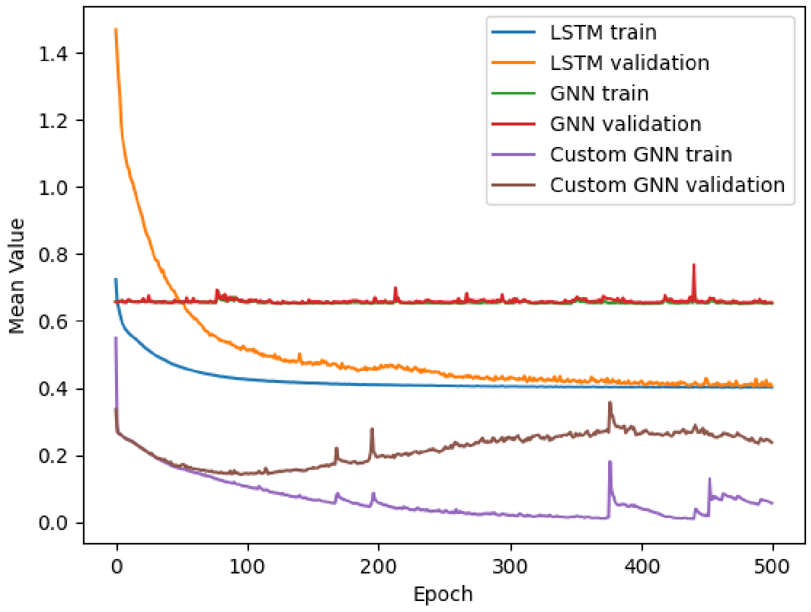

Figure 1. Mean loss per epoch in 30 runs.

5. CONCLUSION AND FUTURE RESEARCH

This paper has demonstrated the significant potential of GNNs in bearing fault diagnosis, by successfully adapted GNNs for analyzing the multidimensional vibration signals in the CWRU bearing dataset. This graph-based representation allowed the model to encapsulate the spatial and temporal dependencies among the bearings and their respective fault conditions, leading to an enhanced feature extraction process and hence more accurate diagnosis. The results revealed that GNNs could achieve state-of-the-art performance, underscoring their value to the field of bearing fault diagnosis in the industrial maintenance context.

The findings of this study underscore the importance of advancing machine learning and deep learning techniques like GNNs in an Industry 4.0 landscape where predictive maintenance strategies are increasingly vital. Not only do these approaches extend the lifespan of machinery, but they also augment productivity and mitigate maintenance costs, signifying their economic and practical value.

The methodology employed in this research, while innovative, is not without limitations. The custom Graph Convolutional Recurrent Network, although adept at capturing spatial and temporal dependencies, may introduce added complexity, potentially making it computationally intensive and harder to interpret compared to simpler models. Its reliance on the PyTorch framework might also pose challenges in terms of scalability and deployment in different environments. Furthermore, while the CWRU dataset offers a rich set of bearing vibration signals, it may not encompass all real-world scenarios, potentially limiting the generalizability of the model. Lastly, the comparison to traditional LSTM models and classical CGN is bounded by the specific implementations chosen, which may not represent the full capabilities of these alternative methods.

For future work, several avenues can be explored to extend the study's scope and potential impact. Such as incorporating more complex scenarios, since this study used a structured and well-curated dataset, the CWRU bearing dataset. Testing the model with real-world, less structured data, could provide further validation of the model's robustness and its ability to generalize across different industrial settings.

Also exploring different GNN variations could be interesting. While this study demonstrated the effectiveness of a specific GNN architecture, exploring other GNN variations or combinations could yield even better results. Moreover, the combination of GNNs with other machine learning or deep learning models could be considered to improve accuracy and performance. These endeavors would expand our understanding of GNN's role in predictive maintenance and fault diagnosis, reinforcing its utility in various industrial applications.

6. ACKNOWLEDGEMENTS

The author Luiza Scapinello Aquino da Silva would like to thank the Coordination of Higher Education Personnel Improvement (CAPES-DS). The authors Mariani and Coelho would like to thank the National Council of Scientific and Technologic Development of Brazil - CNPq (Grants number: 303908/2015-7-PQ, 303906/2015-4-PQ) and Fundação Araucária PRONEX for its financial support of this work.

7. REFERENCES

Case Western Reserve University Bearing Data Center, Dec. 2019, [online] Available: https://csegroups.case.edu/bearingdatacenter/home.

Guo, X., Chen L., and Shen, C., 2016. "Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis". *Measurement*, Vol. 93, pp. 490-502.

Hang, M., Neville, J., and Ribeiro, B., 2021 . "A collective learning framework to boost gnn expressiveness for node classification*". In International Conference on Machine Learning,* pp. 4040-4050.

Hoang, D. T., and Kang, H. J., 2019. "A survey on deep learning based bearing fault diagnosis". *Neurocomputing,* Vol. 335, pp. 327-335.

Liang, T., Wu, S., Duan, W., and Zhang, R., 2018. "Bearing fault diagnosis based on improved ensemble learning and deep belief network". *In Journal of Physics: Conference Series,* Vol. 1074, pp. 012154.

Nacer, S. M., Nadia, B., Abdelghani, R., and Mohamed, B., 2023. "A novel method for bearing fault diagnosis based on BiLSTM neural networks". *The International Journal of Advanced Manufacturing Technology*, Vol. 125, pp. 1477-1492.

Niu, Z., Zhong, G., and Yu, H., 2021. "A review on the attention mechanism of deep learning". *Neurocomputing*, Vol. 452, pp. 48-62.

Ruby, U., and Yendapalli, V., 2020. "Binary cross entropy with deep learning technique for image classification". *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 9, N. 10.

Shao, H., Jiang, H., Lin, Y., & Li, X. (2018). A novel method for intelligent fault diagnosis of rolling bearings using ensemble deep auto-encoders. Mechanical Systems and Signal Processing, 102, 278-297.

Smith, W. A., and Randall, R. B., 2015. "Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study". *Mechanical Systems and Signal Processing*, Vol. 64, pp. 100-131.

Wu, L., Cui, P., Pei, J., Zhao, L., and Song, L., 2022. "Graph neural networks", pp. 27-37. Springer Singapore.

Wu, B., Yang, X., Pan, S., and Yuan, X., 2021. "Adapting membership inference attacks to GNN for graph classification: approaches and implications". *In IEEE International Conference on Data Mining,* pp. 1421-1426.

Yu, Y., Si, X., Hu, C., and Zhang, J., 2019. "A review of recurrent neural networks: LSTM cells and network architectures". *Neural Computation*, Vol. 31, N. 7, pp. 1235-1270.

Zhang, M., and Chen, Y., 2018. "Link prediction based on graph neural networks". *Advances in Neural Information Processing Systems*, Vol. 31.

Zhang, S., Tong, H., Xu, J., and Maciejewski, R., 2019. "Graph convolutional networks: a comprehensive review". *Computational Social Networks*, Vol. 6, N. 1, pp. 1-23.

Zhang, S., Zhang, S., Wang, B., and Habetler, T. G., 2020. "Deep learning algorithms for bearing fault diagnostics—A comprehensive review". *IEEE Access*, Vol. 8, pp. 29857-29881.

Zhang, X., Liang, Y., and Zhou, J., 2015. "A novel bearing fault diagnosis model integrated permutation entropy, ensemble empirical mode decomposition and optimized SVM". *Measurement*, Vol. 69, pp. 164-179.

Zhu, G., Liu, X., and Tan, L., 2023. "Bearing fault diagnosis based on sparse wavelet decomposition and sparse graph connection using GraphSAGE". In *IEEE 2nd International Conference on AI in Cybersecurity (ICAIC)*, pp. 1-6.

8. RESPONSIBILITY NOTICE