Optimizing Machine Learning Algorithms for Fault Classification in Rolling Bearings: A Bayesian Optimization Approach

Muhammad Zain Yousaf, Zhejiang University, China

Josep M Guerrero, Valladolid University, Spain

Muhammad Tariq Sadiq, University of Essex

Accepted for publication in Engineering Applications of Artificial Intelligence.

Optimizing Machine Learning Algorithms for Fault Classification in Rolling Bearings: A Bayesian Optimization Approach.

Abstract

Modern power machinery is inherently complex and operates under dynamic operating conditions, so they demand advanced solutions based on deep learning to diagnose bearing faults inside rotating equipment that cause unplanned downtime and safety issues, leading to operational challenges. However, most deep learning approaches aim to improve performance by incorporating hybrid neural networks that rely on multiple convolutional and temporal units, often overlooking optimizing the large number of hyperparameters that define the structure and performance of hybrid models along with the associated computational constraints. To address this gap, this study presents an innovative approach for the detection and classification of bearing faults by integrating an optimized sparse deep autoencoder (DAE) with a Bidirectional Long Short-Term Memory model (Bi-LSTM). The optimal network structure and hyperparameters are determined through Bayesian optimization (BO) with parallel settings, which automatically searches for network configurations that improve the feature extraction ability of the DAE and the generalization ability of the Bi-LSTM for more efficient fault classification in rolling bearings. Parallel optimization accelerates network structure and hyperparameter tuning by evaluating multiple configurations at once. It leverages the full potential of available multi-core Central Processing Units (CPUs)/Graphics Processing Units (GPUs) in conjunction with a lightweight BO surrogate model. This autonomous and user-friendly framework generates inputs from principal component analysis for linear and BO-DAE for non-linear feature extraction and selection, which are then used to train a BO-enhanced Bi-LSTM. This three-stage optimized method effectively captures spatial and temporal dependencies in vibrational signals, achieving superior efficiency, accuracy, and reliability compared to shallow and deep learning models. Evaluation metrics, including macro precision (99.50%), recall (99.60%), F1-Score (99.57%), and Cohen's Kappa metric ($C\kappa$ = 99.53%), demonstrate the efficacy of our approach for bearing fault classification in industrial applications.

Keywords: Rolling Element Bearings, Modern Power Machinery, Deep Autoencoder, Bidirectional Long Short-Term Memory, Bayesian Optimization.

1) Introduction

By 2030, the global GDP is anticipated to exceed $125 trillion, fueled by a compound annual growth rate (CAGR) of approximately 3.5%, primarily driven by rapid technological advancements and the widespread adoption of Industry 4.0 [1]. There is an ongoing integration of automation in industrial manufacturing through smart manufacturing and the Industrial Internet of Things (IIoT), which leads to a 30% increase in the deployment of rotating machinery across different critical sectors. In the power sector, energy companies are prime examples that use IIoT systems to monitor the performance of rotating machinery like gas turbines and compressors in real-time. This integration optimizes energy consumption, minimizes downtime, and extends the operational lifespan of gas turbines and compressors. However, the reliability of rotating machines depends on the accurate detection and classification of faults in rolling bearings—a vital component in rotary machines that is responsible for almost 40% of all machinery failures [2]. The failures result in unexpected downtime, expensive maintenance costs, and catastrophic failures in some cases that could jeopardize human safety. Therefore, advances in fault classification techniques are essential to prevent these severe consequences, ensuring seamless operation and supporting the global GDP growth targets set for 2030.

It is evident from reviewing the literature that vibrational signals from modern machinery are vital for detecting and classifying bearing faults, and the methodologies for fault diagnosis can be broadly categorized into four main domains: 1) model-based, signal processing-based, machine learning-based (ML), and deep learning-based (DL) methods [3]. Model-based are the earliest approaches that depend upon physical principles and mathematical models to describe equipment behaviour under normal and faulty conditions. However, despite the valuable information present within vibrational signals, model-based methods like spectral analysis [4] and envelope demodulation [5] have their limitations. These conventional methods depend on deep domain knowledge to develop accurate models, which can be a barrier in complex systems. Moreover, the complex operating conditions typically add non-linearity to the system with heavy background noises, which affects the model's robustness and leads to misdiagnosis. Depending upon the operating conditions, specific signal processing methods emerged to complement model-based approaches by extracting fault-related features from the non-linearity and non-stability of recorded vibration signals. For this purpose, techniques like time-domain analysis (such as

kurtosis, skewness, mean, and crest factor), frequency-domain analysis (such as Fourier transform (FT) and power spectral density), and time-frequency analysis (employing wavelet transform (WT), Hilbert transform, short-time Fourier transform (STFT)) were adopted [6]. In recent attempts, *C. Hu et al.* s [7] used FT and discrete WT to isolate specific frequency components and transient signals associated with faults. Adaptive threshold settings were then applied to metrics such as mean and kurtosis to detect faults. While these settings provide a straightforward approach for fault detection in vibration signals, their limitations, such as sensitivity to noise, static nature, inability to analyze multivariate data, and frequent recalibration under complex operating conditions, require addressing.

As a branch of data-driven techniques, ML models can adapt to varying operational conditions without manual recalibration and capture the intricate relationship between multivariate data for modern fault diagnosis applications in complex industrial environments [8]. Several traditional ML models have been reported for rolling bearing fault diagnosis in recent years [8]. A fault diagnosis process is divided into denoising, feature extraction, feature selection (FS), and classification for traditional ML models. For the fault classification, different well-known classifiers, such as support vector machine (SVM) [9], artificial neural network (ANN)[10], ***k***-nearest neighbour (***k***-NN)[11], Naïve Bayes (NB)[12], neural fuzzy logic [13] and random forest (RF) have been applied. Whereas to capture the bearing characteristics under dynamic operating conditions, signal processing techniques are often employed to denoise and extract features from the time-domain, frequency-domain, and time-frequency domain, which are then used to design a high-dimensional feature vector. However, a significant challenge arises because some features may be redundant or irrelevant to the diagnostic target. Therefore, the FS step for different diagnostic tasks is crucial, but it is subjective, time-consuming, and inefficient without sufficient engineering expertise [14]. These aforementioned limitations stem from the shallow architectures often used in traditional ML models. As a result, there is a pressing need to develop autonomous end-to-end solutions based on deep learning (DL), as these methods extract meaningful information from high-dimensional feature vectors without any expertise.

DL models represent a significant advancement in artificial intelligence (AI) for fault diagnosis in advanced systems. Convolutional or temporal layers within DL architectures extract meaningful information either from raw vibrational signals or processed multi-domain features. This extracted information is then passed through dense layers to effectively differentiate output classes [15]. This capability makes DL models a suitable option for detection and classification tasks in the fault diagnosis of rolling bearings. A summary of the most used DL models, including convolutional neural networks (CNN), long short-term memory (LSTM), bidirectional LSTM (Bi-LSTM), and deep auto-encoder (DAE) are elaborated in Table 1.

Table 1. Related work summary of rolling bearing fault diagnosis for detection and classification using Deep Learning methods

Related work summary of rolling bearing fault diagnosis for detection and classification using Deep Learning methods						
Metrics	Methods used Feature selection and classification	Methods used for Feature Extraction	Noise Immunity	Calculation Rate	Explanation	Limitations
Single Deep Learning Models of CNN, LSTM, Bi-LSTM, and DAE						
[16]	CNN	Continuous Wavelet Transform (CWT)	Good	Good	Eliminates manual feature extraction. The algorithm is strong under noise via Time-frequency (TF) images.	The adaptive algorithm requires computational resources and human expertise for CWT processing.
[17]	CNN	Raw Vibration Signals (RVS)	Good	Moderate	Raw vibration signals are transformed into two-dimensional grayscale images for feature extraction.	Data preprocessing requires computational resources.
[18]	CNN	CWT for RVS	Excellent	Good	Incorporates spatial and channel attention modules to focus on representative features within the TF images.	Significant computational resources and human expertise are required for CWT preprocessing.
[19]	CNN	RVS	Excellent	Good	The proposed framework simplifies fault diagnosis by preprocessing raw vibration signals into 2D grayscale images for input into attention-based CNN.	Denosing with kernel Principle Component Analysis (PCA) removes subtle important details along with noise.
[20]	CNN	Short-Time Fourier Transform (STFT)	Moderate	Moderate	Using inception blocks enables the model to capture at different frequency bands.	ICN architecture involves multiple layers, increasing the computational burden.
[21]	CNN	RVS	Excellent	Moderate	Utilizing two convolution layers, max-pooling, fully connected layers, and a Softmax layer for multiclass fault classification.	CNN architecture involves multiple layers, increasing the computational burden.
[22]	CNN	Raw Sound Signals	Good	Good	The end-to-end CNN model combines the advantages of multi-channel signal fusion and automatic feature learning to achieve superior fault detection accuracy.	Requires controlled environments for acoustic signal acquisition, which may limit real-world applications.
[23]	CNN	Extracts Periodic Pulses	Good	Moderate	This work proposes to combine optimized signal preprocessing with CNN-based classification.	Human expertise is required to optimize the signal preprocessing model.
[24]	LSTM	Periodic Sparse Attention + LSTM Units for RVS	Good	Moderate	Extracts long-term dependencies from fault signals, capturing time-series correlations in vibration data. Periodic, where sparse attention minimizes the impact of random interference.	Increase the number of parameters for training.

[25]	LSTM	Raw Vibration Signals + LSTM Units	Good	Moderate	Enhances standard LSTM by iteratively optimizing the number of hidden layer nodes to balance information from input and forget gates.	Single dataset used for testing.
[26]	LSTM	Raw vibration signals + LSTM Units	Good	Moderate	The stacked LSTM architecture extracts temporal features at multiple abstraction levels, effectively capturing the inherent structure of vibration signals.	Requires careful hyperparameter tuning.
[27]	Bi-LSTM	Wavelet Transform	Excellent	Good	This method integrates advanced signal preprocessing with Bi-LSTM to enhance fault diagnosis.	The single dataset used for testing and human expertise for WT.
[28]	Bi-LSTM	Sliding Window Input	Excellent	Good	This method tackles the challenge of limited failure using transfer learning combined with Bi-LSTM.	Tendency for overfitting.
[29]	Deep Auto Encoders (DAE)	Frequency Spectrum Input	Excellent	Excellent	DAE excels at discriminating complex spatial characteristics. These extracted features help pinpoint intricate structural characteristics of bearing faults.	Performance depends on several hyperparameters optimization.
Hybrid Deep Learning Models of CNN, LSTM, Bi-LSTM, and DAE						
[30]	CNN-LSTM	Convolutional and Temporal Features from RVS	Moderate	Moderate	BiConvLSTM architecture removes information loss that occurs in conventional CNN-LSTM pipelines by simultaneously processing spatial and temporal features.	Accuracy for fault direction classification remains lower (84.72%) compared to fault type and location.
[31]	CNN-LSTM	Convolutional + Temporal Features from RVS	Good	Good	The blend of dilated convolutions, residual networks, and LSTM gates results in a model capable of handling noise and fluctuating conditions.	Performance depends on several hyperparameters optimization.
[32]	CNN/Bi-LSTM	Convolutional and Bi-Temporal Features from RVS	Excellent	Good	PCA improves computational efficiency and reduces noise, while the CNN/Bi-LSTM combination boosts the model's ability to process complex, time-series data.	A single dataset was used for testing.
[33]	CNN-SVM	CWT for RVS	Good	Moderate	CNN extracts deep features from vibration signals, and the SVM classifier provides fault classification.	Accuracy decreases with highly noisy datasets.
[34]	CNN-SVM	Raw Vibration Signals	Excellent	Good	Feature representations extracted via CNN are mapped to a cleaner space, enabling SVM to classify faults.	Needs broader validation
[35]	Deep Neural Network (DNN)	DAE	Good	Moderate	The paper proposes a novel DNN-based method for fault diagnosis of rotating machinery.	The DNN requires significant training time compared to shallow networks.
[14]	Softmax Classifier	Ensemble DAE	Good	Good	This study addresses the limitations of individual deep learning models by proposing an ensemble-based approach for rolling bearing fault diagnosis.	Performance depends on several hyperparameters optimization.
[36]	Ensemble Classification	DAE	Good	Good	The final classification relies on combining the outputs of multiple classifiers using a majority voting strategy,	No state-of-the-art shallow and DL models in comparative studies

Table 1 showcases recent studies on rolling bearing fault diagnosis using deep learning (DL) algorithms, which can be divided into two main categories: single and hybrid models. Single models rely on one specific DL architecture(such as CNN, LSTM, Bi-LSTM, or DAE) for fault diagnosis, whereas hybrid models combine two or more techniques to improve classification accuracy and performance. However, the studies summarized in Table 1 often overlook the general limitations inherent in single DL models, which are briefly outlined below:

- **CNN:** overfitting (low generalization) and struggle to capture temporal features in vibration signals [37].
- **LSTM:** can extract temporal features of vibration signals and improve model generalization through gate structures [38]. However, with large datasets, it struggles to capture non-linear characteristics and faces a slow convergence rate [38]. Moreover, it seeks information in one direction.
- **Bi-LSTM:** can process vibration signal sequences in bi-direction, which helps capture non-linear characteristics and complex temporal information in large datasets. The overall generalization is also better with faster convergence rates. However, complexity arises due to an increase in the number of hyperparameters for training.
- **DAE** is a simple and easy-to-train DL model that can reduce the dimensionality of high-dimensional feature vectors and extract meaningful information [29]. However, hyperparameter tuning is required for efficient performance.

These challenges can be addressed by leveraging the individual strengths of single models for feature extraction and pattern classification. **1)** While some studies above have explored hybrid models and highlighted their enhanced fault classification accuracy, these models are more or less limited to a maximum of two techniques. Nevertheless, this integration often overlooks challenges such as the increased number of parameters associated with each model, the requirement for precise tuning in these deeper networks to encounter obstacles such as exploding or vanishing gradients, and deterioration along with the computational burden [39, 40]. Therefore, selecting parameters with efficient optimization in deeper neural networks serves as an instrumental tool that enhances the true efficacy of the hybrid models while considering computational time constraints. This is because traditional approaches to hyperparameter tuning often face time constraints due to high computational costs. This study pioneers the application of an optimized hybrid model by integrating sparse DAE and Bi-LSTM. Moreover, to maximize the performance of each DL model, a meticulous optimization process is employed via Bayesian Optimization (BO). **2)** In addition, many single and hybrid models above utilize signal processing techniques to extract high-dimensional features instead of working direct with raw datasets. This reliance on signal processing integration

does not present a genuine end-to-end or off-the-shelf solution for fault diagnosis in rolling bearings that any user can readily apply. It still requires domain expertise for feature extraction and preprocessing, which limits the accessibility and practicality of these models in modern equipment with dynamic operating conditions. The contributions of this paper are summarized as follows.

a) To develop autonomous and user-friendly fault diagnosis systems, a deep neural network with optimized parameters is designed to avoid complex signal processing and manual feature extraction.

b) Sparse DAE with an optimized structure enhances the ability to extract non-linear and complex features without prior experience or domain knowledge.

c) This sophisticated feature extraction and selection methodology integrates PCA and BO-based sparse DAE to minimize computational burden and extract linear and non-linear features from vast raw datasets.

d) To solve the exploding or vanishing gradients problem and use the full potential of available multi-core CPUs and GPU processors such as NVIDIA GeForce RTX Series, BO with parallel settings is implemented to enhance computation and model training.

e) This solution can give adaptive and optimal configurations for any type of dataset due to the proposed BO's global optimization ability. This is useful when dealing with complex, black-box functions in fault diagnosis. Besides, BO's auto-tuning ability saves time and money because setting up parameters for complex deep neural networks before training requires human expertise [41].

f) These extracted features, when fed into BO-based Bi-LSTM architecture, the proposed system with fine-tuned parameters for specific data proficiently captures and interprets both forward and backward temporal information. The improved spatial and temporal data analysis enables more precise detection and classification of unseen failures.

g) Unlike prior works that use BO for shallow models or limit its application to a small number of hyperparameters, our framework applies the proposed BO to optimize 15 hyperparameters across both the sparse DAE and Bi-LSTM for superior performance, with search ranges spanning 4–5 orders of magnitude [42-45].

h) In contrast to prior sequential BO methods that rely on the expected improvement (*EI*) acquisition function, our approach introduces the Expected Improvement per Second Plus (*EIps+*) acquisition function and parallel settings, reducing the optimization time by 63.33% for hyperparameter tuning.

i) Experiments are carried out on two different datasets based on bearings and gearboxes. The proposed algorithm achieves an average fault classification accuracy of 99.60% and 99.99%, respectively. Furthermore, the proposed algorithm reaches an average accuracy of 99.12% when tested on noisy data (SNR = 10 dB). It outperforms ***state-of-the-art shallow*** and ***DL models*** in comparative studies.

This paper consists of the following sections: In Section (2), the theoretical and motivation background for PCA, DAE, Bi-LSTM, and BO is presented, and in Section (3), the proposed algorithm is implemented. Section (4) presents experimental and analytical results applicable to evaluation metrics, followed by a conclusion in Section (5).

2) Fundamentals for the Proposed Methodology

This section introduces the dimensionality reduction concepts for applying linear PCA to vibrational signals. It also covers the basics of BO, DAE, and BO-DAE approaches, which are crucial for extracting non-linear features. Finally, we will explore the theoretical framework of BO tuned Bi-LSTM. This paved the way for a robust fault classification system.

2.1 Principal Component Analysis (PCA)

In comparison to counterparts such as linear discriminant analysis (LDA), ICA, and t-distributed Stochastic Neighbor Embedding (t-SNE), PCA can capture large datasets more simply with better interpretability. It is an unsupervised learning technique that helps to transform the dimension of large datasets represented by $n \times p$ matrix. The goal is to transform original p variables into a smaller set of q variables called principal components. Principal components are orthogonal projections (perpendicular) of data onto lower-dimensional space. Ensure that the resulting components capture significant variation within the data and are uncorrelated [46]. Basically, principal components are linear combinations of the original variables and can be written as:

$$PC_k = a_{k1}Y_1 + a_{k2}Y_2 + \cdots\cdots + a_{kp}Y_p \quad (1)$$

Y_1, Y_2 ,…, and Y_p are the vibrational signals in the above equation, and PC_k represents the k number of principal components, coefficients a_{k1}, a_{k2},…, a_{kp} forms the k^{th} eigenvector of the covariance matrix S [46]. Since eigenvectors and

their corresponding eigenvalues are crucial for determining how much variance different principal components capture. Thereupon, singular value decomposition (SVD) is employed to compute these principal components by decomposing the data matrix Y into three matrices $Y = U\Sigma V^T$. For very large datasets of vibrational signals, we have incorporated SVD alongside PCA. This is because SVD helps to decrease computational burden while maintaining the integrity of the data [46]. Now, the equation has two orthogonal matrices, U and V. In the case of Σ, it is a diagonal matrix and contains singular values of Y. The columns of the V matrix represent principal components, and the singular values in Σ correspond to the square root of the eigenvalues of the covariance matrix of Y. This relationship is vital because it helps to determine how well variance is captured from the vibrational signals. By doing so, variance is measured by the eigenvalues λ_i of the covariance matrix. Subsequently, as shown in Table 2, the optimal number of principal components is selected by including the cumulative variance $(C.V)$ criterion with a 95% threshold, as shown below:

$$C.V = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \tag{2}$$

For optimal projection dimensionality reduction, identify the minimum number k as:

$$\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{p} \sigma_i^2} \geq 0.95 \tag{3}$$

According to Equation 3, in Figure 1, k =276 principal components achieve a 95% variance for designated vibrational signals and also help reduce noise while retaining the most essential and interpretable patterns [47]. This initial step also helps reduce the computational load for subsequent tasks.

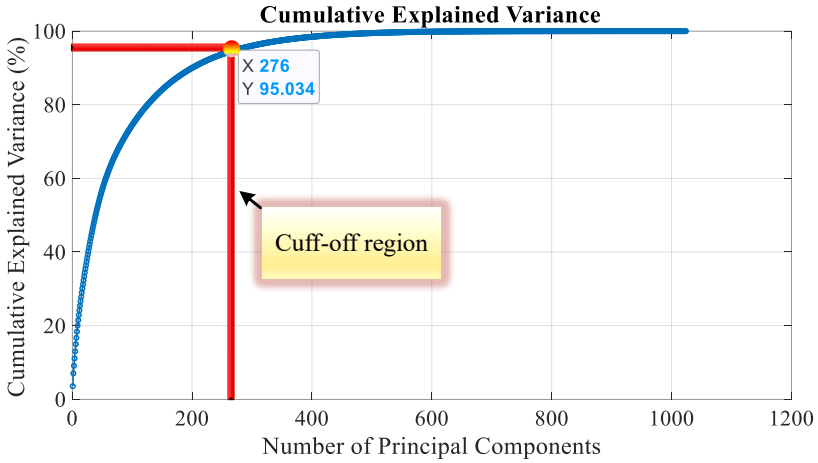


Figure 1. The vibrational signals total variance with a 95% threshold.

Table 2. SVD-based PCA with linear feature reduction.

Algorithm 1: Principal component analysis

Input: Data matrix $\boldsymbol{Y}$, size $n \times p$.

Output: Principal Components matrix $\boldsymbol{P}$.

1: Standardize Data Matrix:

-Standardize data matrix $\boldsymbol{Y}$ for equal contribution to the analysis.

2: Perform SVD:

-Perform SVD on data matrix $\boldsymbol{Y}$ to attain $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{V}$, as $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}$

3: **Variance Assessment:**

- Variance for every individual singular value calculated: $\frac{\sigma_i^2}{n-1}, for\, i = 1 to\, p$

- Sigmoid function is represented as σ .

4: Select Number of Components:

- Determine the optimal k via: $\frac{\sum_{i=1}^{k}\sigma_i^2}{\sum_{i=1}^{p}\sigma_i^2} \geq 0.95$

5: Extract Principal Components:
- Select the first k vectors from $\boldsymbol{V}$.
- From the Principal Components matrix $\boldsymbol{P}$ using these vectors.

6: **End.**

2.2 Deep Autoencoder (DAE)

The deep sparse autoencoder helps to capture complex non-linear patterns that the linear PCA misses. This hybrid combination helps to capture a broad and complementary spectrum of features in vibration data that can be used for clustering, classification, and anomaly detection.

a) Basic-Autoencoder

The three-layer architecture of a basic autoencoder is shown in Figure 2. Autoencoders are generative models that capture non-linear, complex relationships [48]. As the autoencoder trains with unlabelled input data X, it compresses the high-dimensional data from the first layer to the second layer into a lower-dimension space. A decoder tries to reconstruct input data from the second to third layer after the encoding process has been completed. Since the autoencoder captures essential features in the bottleneck layer Z (latent space), it can be perceived as a clever method to transform dimensions. In addition to determining the reduced space dimension and capturing essential information, this layer assists in reconstructing the original data at the output $\widehat{X}$. Achieving efficient compression is impossible if the autoencoder is not properly trained while minimizing reconstruction errors. In other words, inadequate training results in poor learning of prominent features.

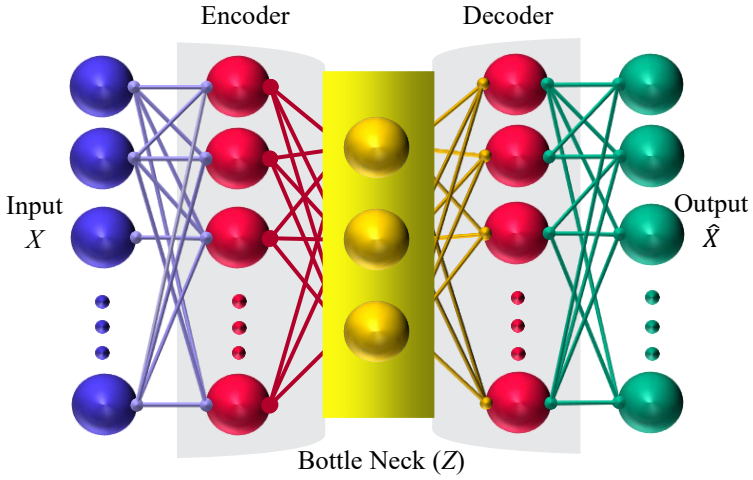


Figure 2. Autoencoder basic structure

With this in mind, iterative backpropagation is used to find the best encoding and decoding weights and biases for the basic autoencoder with a single hidden layer during training. This unsupervised training process requires N samples without labels, which are obtained by high-dimensional input data as $X = \{X_1, X_2, , \ldots, X_t, \ldots, X_N\}$, each X_t is a v-dimensional vector in $\mathcal{R}^{D_x}$. The encoder maps input vector X to a corresponding encoded vector $Z \in \mathcal{R}^{D^{(1)}}$ via a transformation defined by the weights matrix $W^{(1)} \in \mathcal{R}^{D^{(1)}D_x}$ and bias vector $b^{(1)} \in \mathcal{R}^{D^{(1)}}$. Encoding function for each input vector can be written as:

$$Z = h^{(1)}(W^{(1)}X + b^{(1)}) \tag{4}$$

$\mathcal{R}^{D_x}$denotes high-dimensional space. (1) indicates the first layer. $h^{(1)}$ is the non-linear encoder transfer function. A positive saturating linear transfer function (*satlin*) is proposed in this study to help capture complex patterns in high-dimensional vibrational data. In mathematical terms, the generic *'satlin'* can be written as follows:

$$f(Z) = \begin{cases} 0, & if\ Z \leq 0 \\ Z, & if\ 0 < Z < 1 \\ 1, & if\ Z \geq 1 \end{cases} \tag{5}$$

When using '*satlin*', Equation 4 latent feature representation for X_t becomes:

$$Z_t = f(W_t^{(1)}X_t + b_t^{(1)}) = \begin{cases} 0, & if\ (W_t^{(1)}X_t + b_i^{(1)}) \le 0 \\ W_t^{(1)}X_t + b_t^{(1)}, & if\ 0 < (W_t^{(1)}X_t + b_t^{(1)}) < 1 \\ 1 & if\ (W_t^{(1)}X_t + b_t^{(1)}) \ge 1 \end{cases} \quad (6)$$

Within the interval [0, 1] is the output that can be used for feature learning. The decoder then reconstructs the latent space Z back to an approximation $\hat{X}$ of the original input vector, using its own set of weights $W^{(2)} \in \mathcal{R}^{D_\varkappa \times D^{(1)}}$ and bias vector $b^{(2)} \in \mathcal{R}^{D_\varkappa}$. Decoding for the latent features is defined as follows:

$$\widehat{X} = h^{(2)}(W^{(2)}Z + b^{(2)}) \quad (7)$$

(2) indicates the second layer. $h^{(2)}$ is the transfer function for the decoder. We have employed a pure line, represented as $f(Z) = Z$ and $\hat{X} = W^{(2)}Z + b^{(2)}$. Meanwhile, the scaled conjugate gradient '*transcg*' function updates weight and biases during training to minimize the cost function. Cost functions measure the error between inputs X and outputs $\hat{X}$ [49].

2.2.1 Sparse Autoencoders

Sparse autoencoders introduce regularization techniques that enhance feature recognition [48]. Sparsity regularization ensures that a few neurons are active for each vibrational input set in the hidden layer. For this reason, neurons can recognize specific, possibly unique, characteristics of input data to learn more meaningful and distinct features. This objective is achieved by adding a regularizer to the cost function. It is based on the average activation value of a neuron. For a neuron 'α' in the hidden layer, the average activation over the training set is $\hat{p}_\alpha$. Here, it is calculated as:

$$\widehat{p}_\alpha = \frac{1}{\beta}\sum_{\gamma=1}^{\beta} h(w_\alpha{}^{(1)T}X_\gamma + b_\alpha{}^{(1)}) \quad (8)$$

Equation 8 has $w_\alpha{}^{(1)T}$, which is the α^{th} row of the weight matrix $W^{(1)}$, $b_\alpha{}^{(1)}$ is the α^{th} entry of the bias vector $b^{(1)}$, and X_γ is the γ^{th} training example. β is the total number of training examples. For this study, the cost function is augmented with a regularization term to induce low average activation levels ($\hat{p}_\alpha$).

b) Sparsity Regularization

To add such a regularization term in this study, Kullback-Leibler (*KL*) divergence is used for sparsity that works to keep the $\hat{p}_\alpha$ close to a small value. This is achieved by comparing $\hat{p}_\alpha$ with a small predefined sparsity parameter $\boldsymbol{\rho}$. Sparsity proportion $\boldsymbol{\rho}$ is a hyperparameter that can be used to set the average activation value [50]. In doing so, the sparsity regularization term can be written as follows:

$$\Omega_{sparsity} = \sum_{\alpha=1}^{D^{(1)}} KL(\rho \,\|\, \widehat{p}_\alpha) = \sum_{\alpha=1}^{D^{(1)}} \rho \log(\frac{\rho}{\widehat{p}_\alpha}) + (1-\rho)\log(\frac{1-\rho}{1-\widehat{p}_\alpha}) \quad (9)$$

The idea behind this penalty function is that $\Omega_{sparsity}$ increases monotonically as $\hat{p}_\alpha$ diverges from ρ otherwise $KL(\rho \| \hat{p}_\alpha) = 0$ if $\hat{p}_\alpha = \rho$. As a result of *KL* divergence, it is added to the cost function to help the neurons in the hidden layer focus on and respond to certain features in the training data.

c) L_2 regularization

Furthermore, this study includes L_2 regularization term to prevent excessive weights during training sparse autoencoder. This modification controls overfitting and computes as follows:

$$\Omega_{weights} = \frac{1}{2}\sum_{l=1}^{L}\sum_{\gamma=1}^{\beta_l}\sum_{\alpha=1}^{\kappa_l}(w_{\gamma\alpha}^{(l)})^2, \quad (10)$$

Hidden layers are denoted by L. The input size of layer l is κ_l, and β_l is the output size of layer l. $w_{\gamma\alpha}^{(l)}$ denotes elements of the weight matrix of layer l. The overall cost function integrates these terms to balance reconstruction accuracy, sparsity, and weight magnitude. The adjusted cost function for the proposed algorithm is based on mean-squared error (*MSE*), and the overall cost function is now:

$$E=\underbrace{\underbrace{\frac{1}{N}\sum_{\beta=1}^{N}\sum_{\kappa=1}^{K}\left(X_{\kappa\beta}-\widehat{X}_{\kappa\beta}\right)^{2}}_{mean\,squared\,error}+\lambda\times\underbrace{\Omega_{weights}}_{\substack{L_2\\regularization}}+\delta\times\underbrace{\Omega_{sparsity}}_{\substack{sparsity\\regularization}}}_{adjusted\ \ mean\ \ squared\ \ error\ \ function} \tag{11}$$

Here, K denotes the input dimension. In this modification, $\boldsymbol{\lambda}$ is the

L_2 weight regularization coefficient, and $\boldsymbol{\delta}$ is the sparsity regularization coefficient. To incorporate optimum sparsity and control weight values, the proposed BO algorithm will optimize potential hyperparameters (such as $\boldsymbol{\lambda}$, $\boldsymbol{\delta}$, and $\boldsymbol{\rho}$) to control the influence of regularization terms. As a result, the autoencoder can be trained to learn compact data representations effectively, making it useful for tasks such as dimensionality reduction and feature extraction.

The basic methods for extracting features from high-dimensional vibrational data have been introduced so far. After feature reduction and extraction, the next step is to train the model on the extracted features. To effectively train our proposed model with the extracted features, we briefly introduce Bi-LSTM in the next section.

2.3 Bidirectional Long Short-Term Memory (Bi-LSTM)

LSTM possesses advanced design, and it is an exclusive type of RNN. This exclusive type incorporates a forget gate, an input gate, an output gate, and a cell state. These gating mechanisms help mitigate the common problems of exploding and vanishing gradients often met in RNNs when dealing with complex data identical to vibrational signals from bearings. The architecture of the LSTM is illustrated in Figure 3.

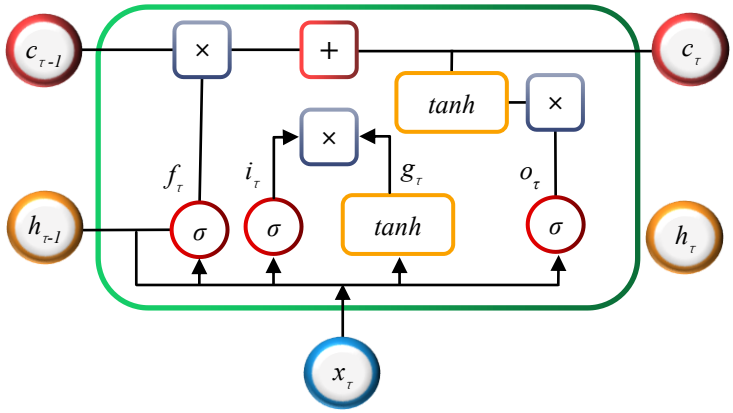


Figure 3. Structure of LSTM cell.

At the time step τ, $x(\tau)$ indicate the LSTM cell input data, $h(\tau)$ indicates the LSTM cell current output, whereas $h(\tau-1)$ is the output from the previous time step $(\tau-1)$. $c(\tau)$ represent the cell state of the LSTM. Within the LSTM cell, the following computation is executed:

$$h(\tau)=f(x(\tau),h(\tau-1),w)$$

$$=\begin{cases} i(\tau)=\sigma(w_{xi}x(\tau)+w_{hi}h(\tau-1)+b_i) \\ f(\tau)=\sigma(w_{xf}x(\tau)+w_{hf}h(\tau-1)+b_f) \\ g(\tau)=\tanh(w_{xg}x(\tau)+\vec{w}_{hg}\vec{h}(\tau-1)+b_g) \\ c(\tau)=f(\tau)\odot c(\tau-1)+i(\tau)\odot g(\tau) \\ o(\tau)=\sigma(w_{xo}x(\tau)+w_{ho}h(\tau-1)+b_o) \\ h(\tau)=o(\tau)\odot\tanh(c(\tau)) \end{cases} \tag{12}$$

The variables w_{xi} ,w_{xo} ,w_{xf}, and w_{xg} indicate the weight matrices for the input gate, output gate, forget gate, and cell gate. Moreover, b_i, b_o, b_f, and b_g are the bias vectors corresponding to the input gate, output gate, forget gate, and cell state. σ symbolize the sigmoid function, respectively.

As mentioned earlier, LSTM models have shown better performance than RNNs in addressing problems related to long-term dependency. At the same time, LSTM cannot consider past and future contextual information due to its only one-direction sequence processing. A bi-directional mechanism is incorporated into the Bi-LSTM structure to overcome LSTM

constraints. As illustrated in Figure 4, this structure comprises two LSTMs: forward LSTM manages the sequence from the past to the future and another from the future to the past (i.e., backward LSTM). The Bi-LSTM structure output is formed by cascading vectors from both the forward and backward sequence outputs, as shown below:

$$h(\tau) = h^{f}(\tau) \oplus h^{b}(\tau) \tag{13}$$

$h^{f}(\tau)$ and $h^{b}(\tau)$ are the forward and backward sequence outputs. In this study, the Adam optimizer adaptively tunes Bi-LSTM network interior parameters (θ), including gradient moments, weights, and biases for both forward and backward LSTM cells [41]. For optimal convergence and loss function (ϱ) minimization, Adam updates parameters via $\theta_{\tau+1} = \theta_{\tau} - \frac{\eta}{\sqrt{\hat{v}_{\tau}}+\epsilon}\hat{m}_{\tau}$. Here, $\hat{m}_{\tau}$ and $\hat{v}_{\tau}$ are adaptive moments of the gradients that help to improve weights and biases, ϵ is a constant value, and η is the learning rate. This proposed mechanism with the concept of gradient clipping in Bi-LSTM structure, helps decode complex dependencies in the vibrational signals of bearing [41]. For clarity, Adam optimizes model parameters during training while BO tunes hyperparameters.

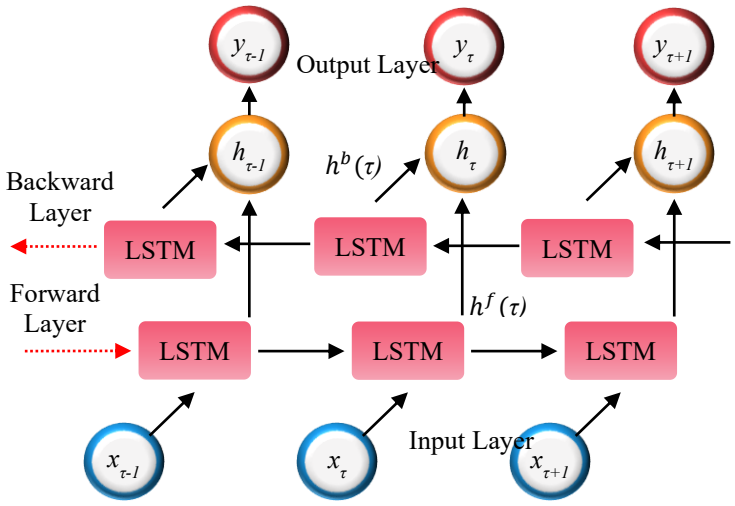


Figure 4. Description of the Bi-LSTM.

2.4 Bayesian Optimization (BO)

Deep learning models such as DAE and Bi-LSTM for optimal performance depend upon fine-tuned hyperparameters. However, the global optimization of hyperparameters in the high-dimensional, black-box system is time-consuming and a high-cost evaluation. BO tackles this challenge by leveraging a probabilistic model that explores the hyperparameter space in an intelligent manner, lowering evaluation costs and improving model performance [51].

Gaussian Process (GP): BO leverages GP to model the unknown objective function probabilistically $f(\dot{x})$. GP is a non-parametric model defined by a mean function $\dot{m}(\dot{x})$ and covariance function $\dot{k}(\dot{x},\dot{x}')$, known as the kernel. These elements facilitate accurate prediction of the function distribution that best fits the observed data. In general, for new input $\dot{x}$, GP assumes the underlying function values have a multivariate normal distribution as:

$$f(\dot{x}) \sim \dot{N}(\dot{m}, \dot{K}) \tag{14}$$

Where $\dot{m}$ represents the mean vector and $\dot{K}$ is the covariance matrix. For a dataset $\dot{D}=\{(\dot{x}_i,\dot{y}_i)\}_{i=1}^{\dot{n}}$, where $\dot{y}_i$ are the observed values of $f(\dot{x}_i)$, GP calculates the posterior mean and variance predictions for a new point $\dot{x}_{new}$ by utilizing the observed inputs $\dot{X}$, outputs $\dot{Y}$, and the noise term σ_n^2, as follows:

$$\mu_{posterior}(\dot{x}_{new}) = \dot{k}(\dot{x}_{new},\dot{X})\left[\dot{K}+\sigma_n^2\dot{I}\right]^{-1}\dot{Y} \tag{15}$$

$$\sigma^2_{posterior}(\dot{x}_{new}) = \dot{k}(\dot{x}_{new},\dot{x}_{new}) - \dot{k}(\dot{x}_{new},\dot{X})\left[\dot{K}+\sigma_n^2\dot{I}\right]^{-1}\dot{k}(\dot{X},\dot{x}_{new}) \tag{16}$$

Where $\dot{I}$ is the identity matrix. ***Mean Function:*** The mean function in our GP model is set to a constant mean, which assumes that the objective function has a consistent baseline value across the hyperparameter space. This choice is common in BO applications because it simplifies the model while still allowing the GP to capture complex patterns through

the covariance function. The constant value for the mean in our work is estimated from the observed point. ***Choice of Kernel:*** The covariance function $\dot{k}(\dot{x},\dot{x}')$ defines the relationship between points in the hyperparameter space. We employ the Matérn 5/2 kernel, which is a popular choice for BO due to its flexibility and ability to model smooth but non-linear functions. Moreover, the Matérn 5/2 kernel is less sensitive to small fluctuations, reducing the risk of overfitting to noisy observations. The Matérn 5/2 kernel is defined as:

$$\dot{k}(\dot{x},\dot{x}') = \sigma^2\left(1+\frac{\sqrt{5}.d}{l}+\frac{5.d^2}{3l^2}\right)\exp\left(-\frac{\sqrt{5}.d}{l}\right) \tag{17}$$

Where:

- $d = \left\|\dot{x}-\dot{x}'\right\|$ is the Euclidean distance between two points,
- σ^2 is the variance parameter,
- ℓ is the length-scale parameter, controlling the smoothness of the function.

This kernel is well-suited for our application as it balances smoothness and adaptive nature, allowing the GP to model the complex, non-linear relationships in the hyperparameter space of the sparse DAE and Bi-LSTM in an effective manner.

Acquisition Functions: This decision-making process is based on a predefined strategy and auxiliary optimization to find the next query point [51]. Employing the mean and variance predictions from the GP generates a scalar metric that indicates the potential utility or improvement expected from evaluating candidate points. ***Expected Improvement per Second Plus (EIps+):*** Expected Improvement (*EI*) and its variant, Expected Improvement per Second (*EIps*), are useful acquisition functions that can be significantly improved to make computations more efficient. *EI* and *EIps* guide the selection of the next query point to evaluate by balancing the trade-off between exploration (uncertain regions of the search space, where $\sigma^2_{posterior}(.)$ is high) and exploitation (areas already identified as promising, where $\mu_{posterior}(.)$ is high). At the same time, to execute this task and enhance decision-making under computational resource constraints, lightweight *EIps+* is incorporated. The formula for standard *EI* is:

$$EI(\dot{x}) = (\mu_{posterior}(\dot{x}) - f(\dot{x}_{best}) - \xi)\Phi(\dot{Z}) + \sigma_{posterior}(\dot{x})\phi(\dot{Z}) \tag{18}$$

$$\dot{Z} = \frac{\mu_{posterior}(\dot{x}) - f(\dot{x}_{best}) - \xi}{\sigma_{posterior}(\dot{x})} \;\; if \sigma_{posterior}(\dot{x}) > 0, else\, \dot{Z} = 0, \tag{19}$$

$f(\dot{x}_{best})$ is the best point observation so far. Φ and ϕ are the cumulative distribution function and probability density function of the normal distribution, respectively, and ξ is a hyperparameter encouraging exploration. For *EIps*, the expected evaluation time $s(\dot{x})$ factored as:

$$EIps(\dot{x}) = \frac{EI(\dot{x})}{s(\dot{x})} \tag{20}$$

Lightweight *EIps+* introduces an adjustment for the uncertainty in computational cost:

$$EIps+(\dot{x}) = \frac{EI(\dot{x})}{s(\dot{x}) + \dot{\beta}\sigma_s(\dot{x})} \tag{21}$$

$\sigma_s(\dot{x})$ quantifies the uncertainty in evaluation time and $\dot{\beta}$ is a hyperparameter balancing expected improvement against time uncertainty. ***Advantages Over EI and EIpS:*** *EIps+*has the following benefits over *EI* and *EIps*.

1. While previous methods, such as *EI* and *EIps*, focus on enhancing the objective function, they do not account for time and its associated uncertainties. In contrast, the *EIps+* acquisition function incorporates time-aware exploration and quantifies the associated uncertainties. This approach prioritizes the exploration of hyperparameter configurations that offer high improvement potential and predictable evaluation times, avoiding overcommitment to slow or uncertain evaluations.
2. By penalizing uncertain time estimates $\sigma_s(\dot{x})$, *EIps+* reduces its sensitivity to noisy or unstable evaluations.
3. The introduction of $s(\dot{x}) + \dot{\beta}\sigma_s(\dot{x})$ the denominator in Equation 21 facilitates batch evaluations, thereby maximizing hardware utilization (e.g., multi-core CPUs/GPUs).

Integration of Parallel Settings: The extension of *EIps+* to parallel settings allows for simultaneous evaluation of multiple promising hyperparameter configurations. A batch method selects a group of promising options $\{\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n\}$ for evaluation together. Various workers (independent computational units) ran each evaluation to save time, the GP model is updated once all assessments are done. This step helps to find better options in the next iteration and ensures faster convergence while maintaining model performance and reducing optimization time.

3) Implementation of the Proposed Methodology

This section presents an implementation of the algorithm based on the aforementioned basic methods. The proposed algorithm based on the PCA, BO-DAE, and BO-Bi-LSTM is shown in Figure 5.

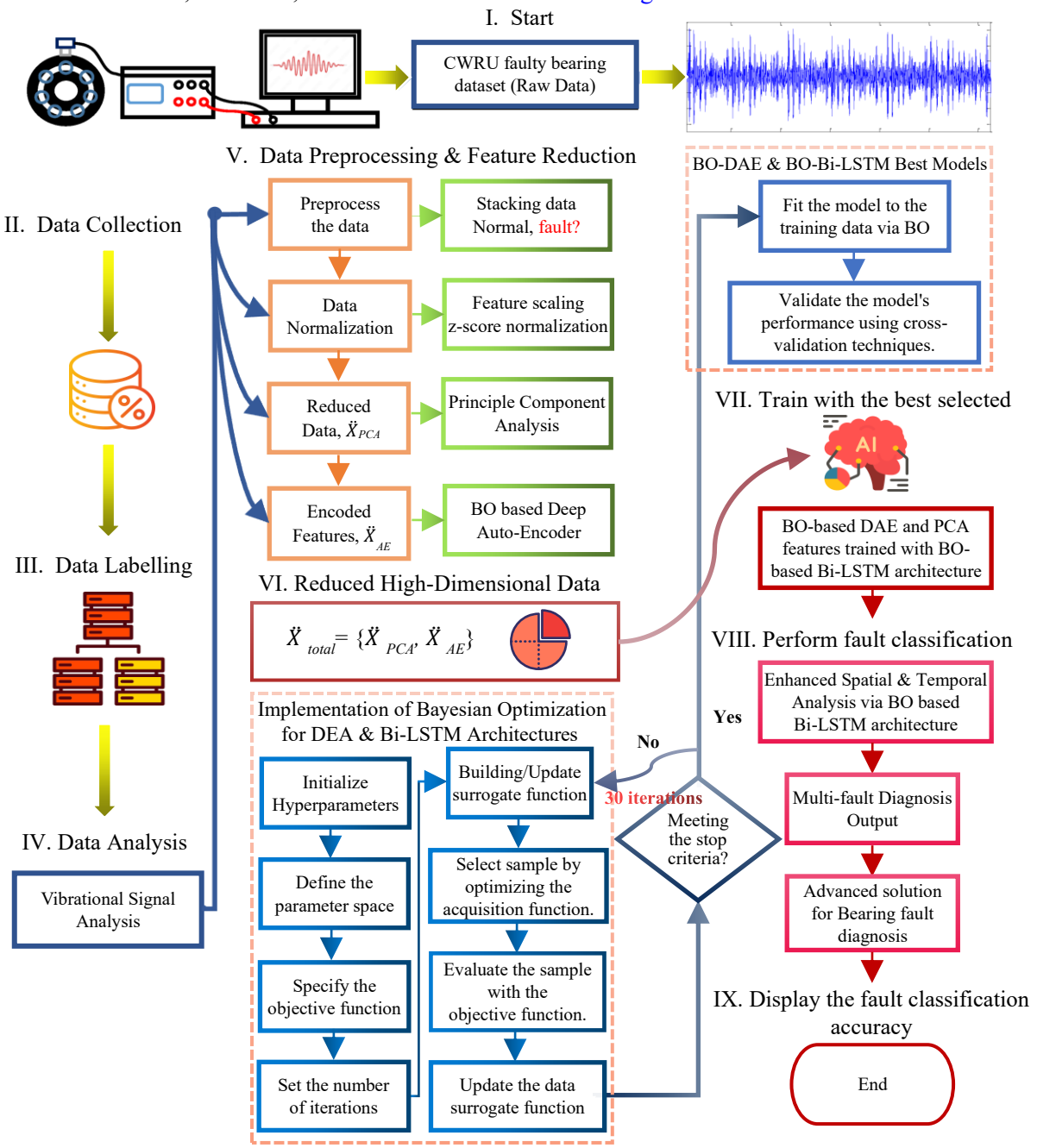


Figure 5. Implementation of the proposed algorithm based on supervised learning

3.1 Data Collection

At first, Case Western Reserve University's (CWRU) faulty bearing dataset is sorted to collect significant data for further analysis. The dataset is categorized into three main types: 1) Inner race faults: The fault occurs on the inner race of the bearing, which provides a path for rolling elements and is closest to the shaft; 2) Outer race faults: These faults are at the outer ring of the bearing and interfaces with housing, 3) Ball faults: These faults are on the ball in ball bearing. These

faults are artificially generated via electro-discharge machining (EDM) on the bearing components with fault diameters ranging from 0.007 to 0.021 inches at different degrees of severity, as tabulated in Table 3. The dataset has ten data groups, including faulted and normal bearings. In this test, a 3-horsepower (HP) reliance electric motor is mounted on the left-hand side of the test bench, while the fan and drive end are equipped with 6205-RS JEM SKF deep groove ball bearings. We collected vibration data at a sampling frequency of 12 kHz by mounting accelerometers (encoders) at particular points on the motor housing, typically near its drive and fan ends, to capture vibration signals from bearings. Load conditions applied to the motor shaft are 3 HP and a speed of 1772 revolutions per minute (rpm), which simulate common operational conditions for such motors.

Table 3. Rolling bearing data set description

Rolling bearing data set description				
Bearing State	Fault Degree	Number Label	Letter Labels	Remarks
Rolling Ball faults 1	0.007	Label 1	RE007	Ball defects 07
Rolling Ball faults 2	0.014	Label 2	RE014	Ball defects 014
Rolling Ball faults 3	0.021	Label 3	RE021	Ball defects 021
Inner race defect 1	0.007	Label 4	IR007	Inner defects 07
Inner race defect 2	0.014	Label 5	IR014	Inner defects 014
Inner race defect 3	0.021	Label 6	IR021	Inner defects 021
Outer race defect 1	0.007	Label 7	OR007	Outer defects 07
Outer race defect 2	0.014	Label 8	OR014	Outer defects 014
Outer race defect 3	0.021	Label 9	OR021	Outer defects 021
Normal	NA	Label 10	Nor	Normal

3.2 Standardization

Considering a dataset with dimension $\ddot{N} \times \ddot{M}$, $\ddot{N}$ gives total samples and $\ddot{M}$ gives the number of features (e.g., CWRU measurements). After sorting the raw signal $\ddot{X}_{raw} = \{\ddot{x}_1, \ddot{x}_2, \ldots, \ddot{x}_{(\ddot{M})}\}$ into ten different types and setting $\ddot{N}$ = 10×117 =1170, where each $\ddot{x}_i$ =1024 are features of the vibrational signal in each sample that correspond to fault type or healthy state. We utilized different standardization techniques such as min-max scaling, robust scaling, decimal scaling, and L_2 normalization. However, z-score normalization leads to more effective compressed representations for training deep learning models as:

$$\ddot{X}_{std} = \frac{\ddot{X}_{raw} - \mu}{\sigma} \tag{22}$$

Where μ and σ represent mean and standard deviation vectors. Equation 22 ensures that features in each sample have small gaps and a consistent range for reduction techniques like PCA and BO-DAE.

3.3 Feature Extraction

PCA transforms standardized data $\ddot{X}_{std}$ into a reduced dimensional space $\ddot{X}_{PCA} = U_k \Sigma_k V_k$, where k =276 components are chosen to achieve a cumulative variance of 95%. With reference to Section 2.1, SVD-based PCA reduces the dimensionality from 1024 to 276 components while maintaining the CWRU dataset's core characteristics. SVD-based PCA aims to collect linear spatial features, whereas BO-DAE is utilized to capture non-linear spatial features from $\ddot{X}_{std}$. Therefore, the sparse autoencoder architecture defined in Section 2.2 is designed. At its core, the bottleneck layer produces a compressed representation $Z^{(i)}$ for the input data $\ddot{X}_{std}^{(i)}$. For ideal features at the bottleneck output, the primary objective is to minimize the total loss function while integrating the reconstruction loss, the sparsity penalty, and L_2 regularization as:

$$E_{Total} = \underbrace{\underbrace{\frac{1}{N}\sum_{\beta=1}^{N}\sum_{\kappa=1}^{K}\left\|\ddot{X}_{std_{\kappa\beta}}^{(i)} - \hat{\ddot{X}}_{\kappa\beta}^{(i)}\right\|^2}_{mean\,squared\,error} + \lambda \times \underbrace{\Omega_{weights}}_{\substack{L_2 \\ regularization}} + \delta \times \underbrace{\Omega_{sparsity}}_{\substack{sparsity \\ regularization}}}_{adjusted\ \ mean\ \ squared\ \ error\ \ function} \tag{23}$$

By optimizing E_{Total}, the proposed DAE learns to encode the essential non-linear features of the data into a bottleneck layer ($\ddot{X}_{AE}$= *Encoder* ($\ddot{X}_{std}$)) with fewer dimensions. E_{Total} can be improved by selecting optimal hyperparameter values before training that balance complexity and generalization, capturing key features for high-dimension vibrational signal reduction,

plus avoid overfitting. To achieve this, BO is employed to find a set of hyperparameters Θ^*, including $\boldsymbol{\lambda}$, $\boldsymbol{\delta}$, and other hyperparameters mentioned in Table 4, by minimizing the aggregated validation loss. Implementation of the proposed BO-DAE is shown in Table 5. The BO models the loss function (MSE) as a GP and uses the acquisition function *EIps+* to decide where to sample next:

$$\Theta^* = arg\min_{\Theta} E_{val}(R(\ddot{X}_{val}; \Theta)) \quad (24)$$

Delved into Equation 24, BO's primary objective is to minimize the objective function, where E_{val} represents the validation loss, and R represents the reconstructed output of the validation data. Θ represent selected values from a search range of hyperparameters.

Table 4. Optimized DAE Hyperparameters

Optimized DAE Hyperparameters		
Hyperparameters	Search Range	BO Selected Parameters
L_2 Weight regularization coefficient (λ)	[1×10^{-5} to 1×10^{-2}]	0.009690540632201
Sparsity regularization coefficient (δ)	[1×10^{-3} to 1]	0.070407786152138
Hidden size	[100 to 200]	198
Sparsity proportion ($\boldsymbol{\rho}$)	[0.01 to 0.5]	0.451387076690407
Max epochs	[100 to 500]	485
Scale data	NA	0

The following sub-section gives the results obtained by implementing BO-DAE.

Table 5. Training DAE with BO and k-Fold CV

Algorithm 2: Proposed BO-DAE

Input: Data matrix $\ddot{\boldsymbol{X}}_{\boldsymbol{std}}$ size $\ddot{\boldsymbol{N}} \times \ddot{\boldsymbol{M}}$.

Output: Optimized Sparse Autoencoder model with hyperparameters Θ^*.

1: Initialization:

-Define a range for hyperparameters, including Max epochs, $\boldsymbol{\delta}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\rho}$, to initiate the BO process.

2: k-Fold Cross-Validation (CV) Setup:

- Integrate k-Fold CV within the BO loop with k=5 (Dividing the dataset $\ddot{\boldsymbol{X}}_{\boldsymbol{std}}$ into five distinct subsets)

3: **Evaluation with k-Fold CV:**

-For each set of hyperparameters Θ

1) Train Sparse Autoencoder with Θ on 5-fold CV training data.

2) Evaluate each fold's validation set.

3) Calculate MSE for the reconstruction error on the validation subset.

4: Results Aggregation:

-Aggregate the MSEs from 5 folds to calculate an average MSE for the set of hyperparameters Θ.

5: Optimization:

- Apply BO to analyze the aggregated MSE results from the 5-fold CV

- Determine the next set of hyperparameters Θ for evaluation in parallel to evaluate via maximizing the acquisition function (Parallel *EIps+*), guided by the GP model.

6: **Iteration:**

- Utilize the parallel setup to evaluate multiple configurations simultaneously, reducing overall optimization time. Repeat steps 3 to 5 until the **30th** BO iterations are done

7: **Final Selection (Best Model):**

- Find the optimal set of hyperparameters Θ^* that minimize the objective function (i.e., aggregated MSE), indicating the best model performance on unseen data subsets.

8: End.

During training, BO uses 30 iterations to fine-tune a set of critical hyperparameters. Based on our experiments, 30 iterations provided a good balance between computational cost and convergence to optimal hyperparameters. Meanwhile, BO is integrated with k-fold CV to lower the MSE (objective function) on a validation dataset. The process took 10,598.6 sec, and the best MSE of 0.10083 was observed on the 24th iteration, demonstrating model accuracy in reconstructing data from compressed representations with optimized hyperparameters. The integration of parallel BO reduced the total

optimization time by approximately 35% compared to traditional BO. The coefficient for the L_2 weight regularization is $\boldsymbol{\lambda}$, which avoids overfitting by applying a penalty proportional to the square of the magnitude of the weights. The search range is set between [1×10^{-5} to 1×10^{-2}], with BO selecting 0.009690540632201. The number suggests that a generalizable model with strong regularization is chosen. This is because it is close to the upper limit of the range. The sparsity regularization coefficient ($\boldsymbol{\delta}$) encourages the model to learn sparse representations by implementing a penalty for non-zero activations in the hidden layers. The search range spanned from [1×10^{-3} to 1]. BO suggests the value of 0.070407786152138. Under those circumstances, an optimal level of sparsity enforcement allows the model to maintain a flexible approach to learning from the data. The total number of neurons in the bottleneck layer of the autoencoder is specified by the hidden size, which determines the ability of the model to compress input. For instance, the selection of 198 reflects a balanced approach to model complexity, allowing for feature extraction without overfitting. From a range of [0.01, 0.5], $\boldsymbol{\rho}$ =0.451387076690407 is selected, indicating a high level of neuron activation in the bottleneck layer that supports the model's comprehensive representation of features. With a search range from [100, 500], 485 epochs are chosen via BO, which gives an optimized value of E_{Total}= 0.2555. These results prove robust training with better convergence and in-depth learning without overfitting.

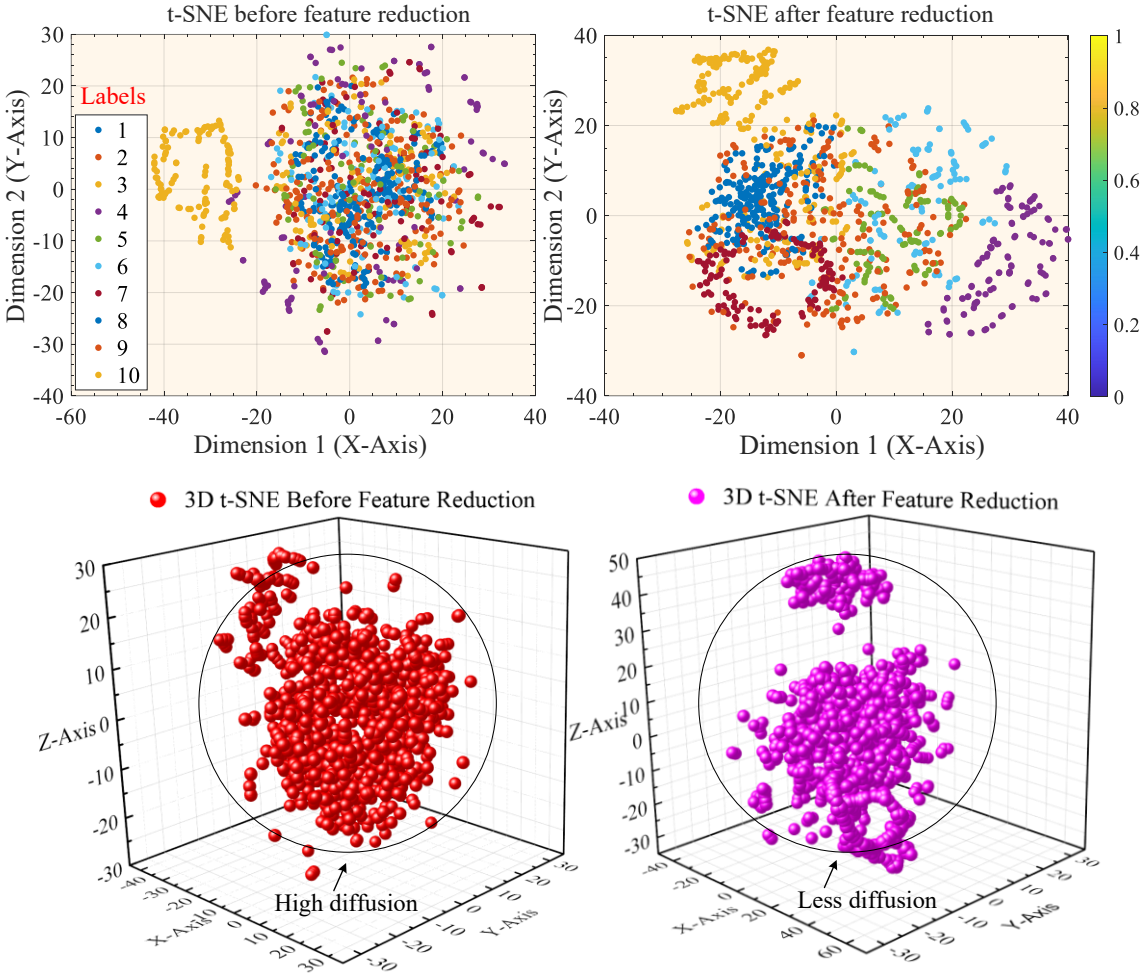


Figure 6. t-SNE representation of before and after feature reduction $\ddot{X}_{Total}$.

3.3.1 Feature Selection

After collecting features from $\ddot{X}_{PCA} = PCA(\ddot{X}_{std})$ and BO-DAE ($\ddot{X}_{AE}$= *Encoder* ($\ddot{X}_{std}$)), the datasets are organized together for each of the fault types as $\ddot{X}_{Total} = \{\ddot{X}_{PCA}, \ddot{X}_{AE}\}$. These reduced features represent the vibrational signal's linear and non-linear contents in the bearing structure. We employ a correlation-based feature selection method to optimize feature selection for data processing further. The theoretical foundation is based on the Pearson correlation coefficient, which evaluates the relationship between two variables:

$$corr(\ddot{X}_i,\ddot{Y}) = \frac{\sum(\ddot{X}_i - \ddot{X}_i)(\ddot{Y} - \ddot{Y})}{\sqrt{\sum(\ddot{X}_i - \ddot{X}_i)^2 \sum(\ddot{Y} - \ddot{Y})^2}} \quad (25)$$

Where $\ddot{X}_i$ and $\ddot{Y}$ present features and targets. $\hat{\ddot{X}}_i$ and $\hat{\ddot{Y}}$ present mean values. In this model, we compute an absolute correlation for each feature with the label. Features are then ranked based on their correlation scores, with higher values indicating a stronger connection to the label. Figure 6 with the help of t-SNE, clearly illustrates how features that are significant to the target variable are identified and prioritized to expedite learning while improving spatial interpretability for future modelling [52].

3.4 Network Mapping

In the next step, the proposed model learns the relation between input $\ddot{X}_g$ and $\ddot{Y}_g$ output with the help of BO-based Bi-LSTM architecture in the offline training mode to classify bearings faults. It is mathematically expressed based on the following equation:

$$\ddot{Y}_g = \mathbb{R}(\ddot{X}_g) \quad (26)$$

Here, $\mathbb{R}$ presents intelligent architecture attained to leverage the temporal characteristics of BO-Bi-LSTM. In this context, the input sequence expanded as $\ddot{X}_g = [\ddot{x}_g^1, \ddot{x}_g^2, \ddot{x}_g^3, \ddot{x}_g^4, \ldots \ddot{x}_g^e, \ldots, \ddot{x}_g^L]$. Each $\ddot{x}_g^e$ is a column vector representing feature inputs with a total length L of data (also the length of time) that feeds into the input layer (range of e is 1 to L). Now, the input data is arranged as follows:

$$\ddot{X} = \ddot{X}_1, \ddot{X}_2, \ddot{X}_3, \ddot{X}_4, \ldots, \ddot{X}_n = \left[\ddot{x}_1^1, \ddot{x}_1^2, \ldots, \ddot{x}_1^L; \ddot{x}_2^1, \ddot{x}_2^2 \ldots, \ddot{x}_2^L; \ddot{x}_g^1, \ddot{x}_g^2, \ldots, \ddot{x}_g^L\right] \quad (27)$$

With reference to Table 2, there are 10 target labels, namely different types of inner, outer along with bearing faults and normal operations. Under those circumstances, the target labels are organized as follows:

$$F = \ddot{Y}_1, \ddot{Y}_2, \ddot{Y}_3, \ddot{Y}_4, \ldots, \ddot{Y}_n \quad (28)$$

Each $\ddot{Y}_j$ corresponds to the classified fault type for the j^{th} sequence, and n represents the total number of sequences in the dataset. After mapping, preprocessed data is segmented for training and validating the BO-optimized Bi-LSTM model across defined folds. Under those circumstances, the best-selected intelligent architecture will classify bearing faults. Table 6 and Table 7 present the BO-based Bi-LSTM best-selected intelligent architecture and optimized parameters.

Table 6. BO-based Bi-LSTM architecture ($\mathbb{R}$)

BO-based Bi-LSTM architecture		
Layer Architecture	Description	Remarks
1) Sequence Input	Initiates processing of variable-length sequences.	Matching input feature size
2) Bi-LSTM Layer	Crucial for capturing complex temporal patterns.	Bi- temporal learning
3) Normalization Layer	Stabilizes learning via normalizing layer outputs.	Improve training efficiency
4) Dropout Layer	Prevents overfitting via randomly omitting units.	Ensure model generalization
5) Bi-LSTM Layer	Bi-LSTM Layer 2	*Bi-lstmLayer-02*
6) Normalization Layer	Normalization Layer 2	*batchNormalizationLayer-02*
7) Dropout Layer	Dropout Layer 2	*dropoutLayer-02*
8) Bi-LSTM Layer	Bi-LSTM Layer 3	*Bi-lstmLayer-03*
9) Normalization Layer	Normalization Layer 3	*batchNormalizationLayer-03*
10) Dropout Layer	Dropout Layer 3	*dropoutLayer-03*
11) Fully Connected	Transforms Bi-LSTM features for classification.	Dense layer with 100 neurons
12) ReLU Layer	Introduce non-linearity to help the model learn complex patterns.(*reluLayer-01*)	
13) Dropout Layer	Mitigate overfitting post-feature extraction and enhance generalization.	
14) Fully Connected	Align output dimension with target classes for classification.	
15) Softmax Layer	Converts outputs to probabilities.	$\ddot{Y}_g$=softmax (W. $\mathbb{R}(\ddot{X}_g)$+b)
16) Classification Layer	Assign the most probable class to each sequence for fault classification.	

3.4.1 BO-based Bi-LSTM architecture ($\mathbb{R}$)

BO incorporates three Bi-LSTM layers to capture temporal dependencies in both forward and backward directions of the input sequence. However, vibration sequence can vary significantly in amplitude depending on factors such as sensor placement, operating conditions, and even equipment age. Therefore, normalization layers ensure that features extracted by the model are a true reflection of the underlying faults and are not skewed by signal variations that are not relevant. In addition to improving convergence, it reduces the risk of becoming stuck in local optima and enhances overall performance. To further improve the performance, the ReLU layer introduces non-linearity into the model, allowing it to learn these intricate relationships and effectively differentiate between different fault types. This is because vibration sequences exhibit complex relationships between features. Especially with Bi-LSTM layers, it enhances model computation compared to Sigmoid and Tanh. After capturing patterns through the Bi-LSTM layers, the hidden layer output is sent to a fully connected layer as an input, where the refined information is categorized. We can distinguish between normal and abnormal rolling element-bearing states by combining softmax and classification layers. However, given the dynamic nature of fault operating environments in rotary machines, it is difficult to include all possible fault scenarios; therefore, training samples are usually too small. This may expose the conventional Bi-LSTM model to overfitting states, plus the hit-and-trial method of selecting hyperparameters is inefficient. Therefore, the following sub-section provides a comprehensive overview of these problems and the criteria used to address them.

3.5 Overfitting Problem

It is understood that overfitting is a critical issue in fault classifications, as rolling bearing datasets are limited, and deep neural networks often overfit over these limited training datasets [41]. It is possible to fit each neural network on the same dataset and average the prediction from all models. However, it is impossible to do so on the scale of rotary machines. Therefore, dropout is a good solution for overfitting. It is a regularization tool that includes training Bi-LSTM with various non-repetitive sub-networks and averaging them. It eliminates neurons from an initial network with the probability $\ddot{\boldsymbol{P}}$. This probability rate is enhanced with the help of BO, presented in Table 7. As a result, the proposed architecture ($\mathbb{R}$) generalizes well to new data, reduces overfitting, and improves the model stability to diagnose faults accurately on unseen vibration signals.

Table 7. Optimized Bi-LSTM Hyperparameters

Optimized Bi-LSTM Hyperparameters		
Hyperparameters	Search Range	BO Selected Parameters
Hidden units for Bi-LSTM Layer	[50 to 200]	89
Initial Learning Rate (ILR)	[1×10^{-4} to 1×10^{-2}]	0.0099
Mini-Batch Size (MBS)	[20 to 128]	94
Learn Rate Drop Factor (LRDF)	[0.1 to 0.9]	0.1943
Learn Rate Drop Period (LRDP)	[1 to 50]	38
Gradient Threshold (GT)	[0.5 to 2]	1.5937
Number of Epochs (NE)	[100 to 300]	173
Number of Bi-LSTM Layers (NBL)	[1 to 5]	3
Dropout Rate ($\ddot{\boldsymbol{P}}$)	[0.1 to 0.5]	0.2230

3.6 Hyperparameter Selection and Training

During training with extracted features, BO aims to build a Bi-LSTM architecture and improve learning by exploring a predefined hyperparameter space to maximize validation performance. This process involves carefully setting the search space and tailoring the objective function. The BO objective function incorporates *k*-fold cross-validation, which trains the model on diverse subsets of data and computes the average validation accuracy to guide the optimization process. The accuracy metric for multi-class is defined as:

$$\text{AvgValAccuracy} = \frac{1}{u}\sum_{z=1}^{u}\frac{\sum_{d=1}^{\ddot{S}}(\ddot{Y}_{\text{Pred}_d} == \ddot{Y}_{val_d})}{\ddot{S}} \tag{29}$$

$\ddot{S}$ is the total number of samples in the validation set for fold z. $\ddot{Y}_{\text{Pred}_d}, \ddot{Y}_{val_d}$ are the predicted and actual labels of the

d^{th} sample in the validation set. The expression == generates 1 when the expression within it is true and 0 otherwise.

The negated value obtained via Equation 29 assists BO in pinpointing the optimal set of hyperparameters, as presented in Table 7. This meticulous adjustment of hyperparameters—including hidden units, initial learning rate (ILR), mini-batch size (MBS), learning rate drop factor (LRDF), learning rate drop period (LRDP), gradient threshold (GT), number of epochs (NE), number of Bi-LSTM layers (NBL), and dropout rate ($\dot{P}$) [41]. Specifically, hidden units help to learn the complex patterns of the vibrational signal. BO selected 89 hidden units per Bi-LSTM layer, which helps to maintain an optimal balance between model complexity and mitigating overfitting issues. ILR = 0.0099 and MBS = 94 aim for fast convergence with acceptable noise. This is because ILR controls model weights during training, whereas MBS affects the stability and speed of the learning process. To improve this learning process during training, LRDF controls the size of the decrease in the learning rate, while the LRDP specifies how frequently (in epochs). GT prevents exploding gradients. Gradients are allowed larger updates (GT =1.5937 threshold) for faster learning, balanced by the learning rate factor, and drop (LRDF =38 epochs, LRDP = 0.1943 factor) in the training cycle. Incorporating three Bi-LSTM layers helps capture complexity without excessive burden. As a result, the model achieved robust performance metrics, including a training accuracy of 100% in every iteration. On the validation set, the model exhibited the best accuracy of 90.89 % at the 28th iteration with an evaluation time of 1.82 seconds.

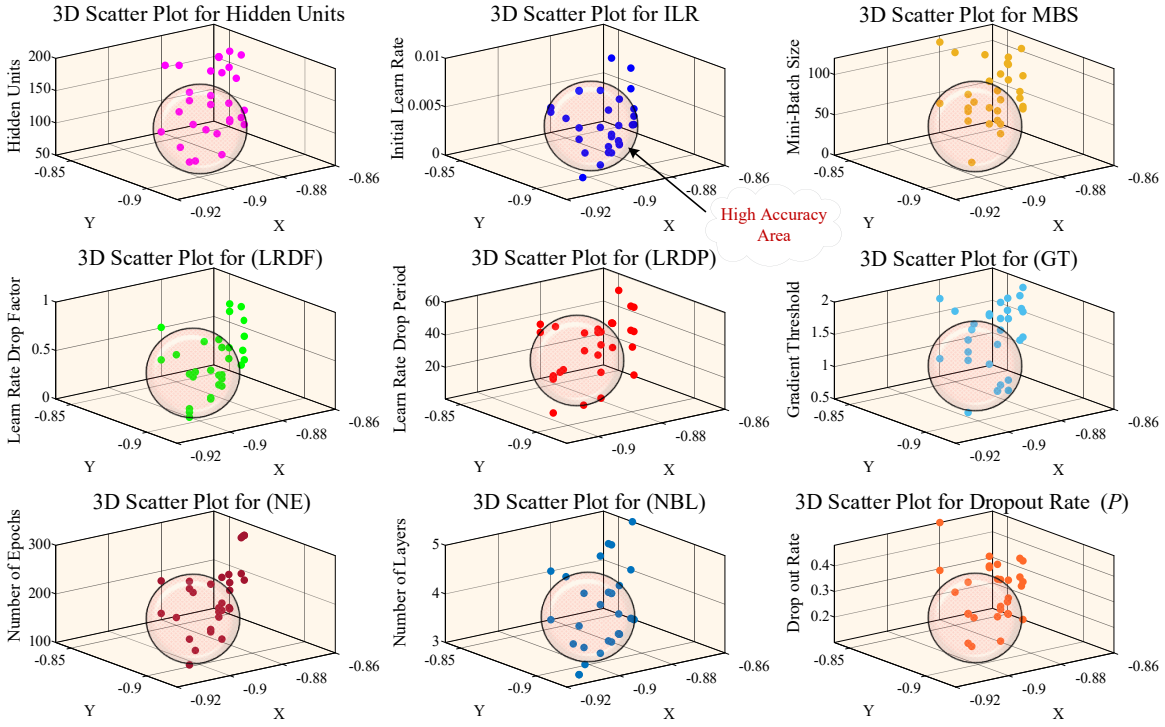


Figure 7. Hyperparameter variation across observed accuracy (X) on validation sets and BO estimated accuracy (Y).

Figure 7 shows the estimated and observed accuracy of the model for 30 iterations. As the accuracy metric increases, BO tends to pick hyperparameter values that are not too big or too small to ensure that the model is balanced and works best without overfitting or underfitting. After optimizing hyperparameters via BO and 5-fold CV for efficiency, the best model undergoes further exhaustive 10-fold CV to assess its performance across several metrics. Thus ensuring robust validation and testing.

4) Results and Discussion

This section meticulously assesses the proposed fault classification algorithm across diverse conditions, including noise settings, comparison with shallows and DL models, and critical feature extraction techniques. It also has confusion matrix analysis and ablation studies (removing PCA, BO-DAE, and BO-based Bi-LSTM) that confirm the model's reliance on both spatial (space) and temporal (time) information.

4.1 Fault Classification Accuracy

In this section, we evaluate the fault classification accuracy of the best selected via different metrics. Apart from the accuracy metric (29), precision, recall, F1-scores, and Cohen's Kappa metrics are used to validate the model performance.

These metrics are mathematically represented as follows.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{30}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{31}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{32}$$

$$\text{Cohen's Kappa} = \frac{p_o - p_e}{1 - p_e} \tag{33}$$

TP and FP represent true positive and false prediction, whereas FN stands for false negative. p_o is observed agreement and p_e is expected agreement by chance.These metrics are used to monitor and measure the model's classification accuracy.

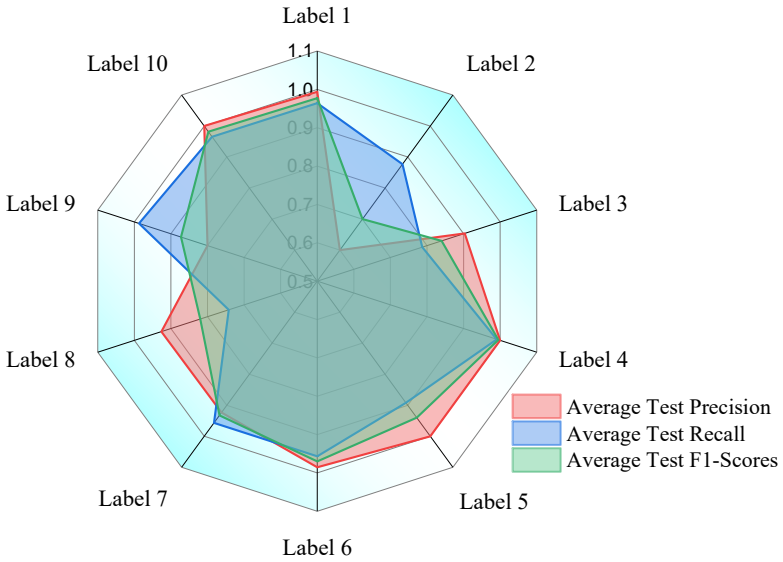


Figure 8. A radar chart presents precision, recall, and F1 scores for each class with a 10-fold CV (Label info Table 3)

After segmenting the selected features into training and validation sets $\ddot{X}_{train}$, $\ddot{X}_{val}$, $F_{train}$and F_{val}, as described in subsection 3.3.1; we proceed to train and evaluate the best model ($\ddot{Y}_g = \mathbb{R}(\ddot{X}_g, \Theta^*)$), g present any specific sequence. Thereupon, metrics are recorded and averaged across folds to determine the model's overall effectiveness. While analyzing the radar chart in Figure 8, Label 1 exhibited the highest precision at 0.9938, indicating a solid model accuracy. In contrast, Label 2 had the lowest precision but a satisfactory recall of 0.9, suggesting a balance between these evaluation metrics. The highest recall was observed for Label 4 at 0.9909, demonstrating the model's effectiveness in identifying true positives. Label 10 achieved the highest F1-Score of 0.9819 for steady-state values, with Label 4 leading in non-steady-state scenarios with an F1-Score of 0.9952.

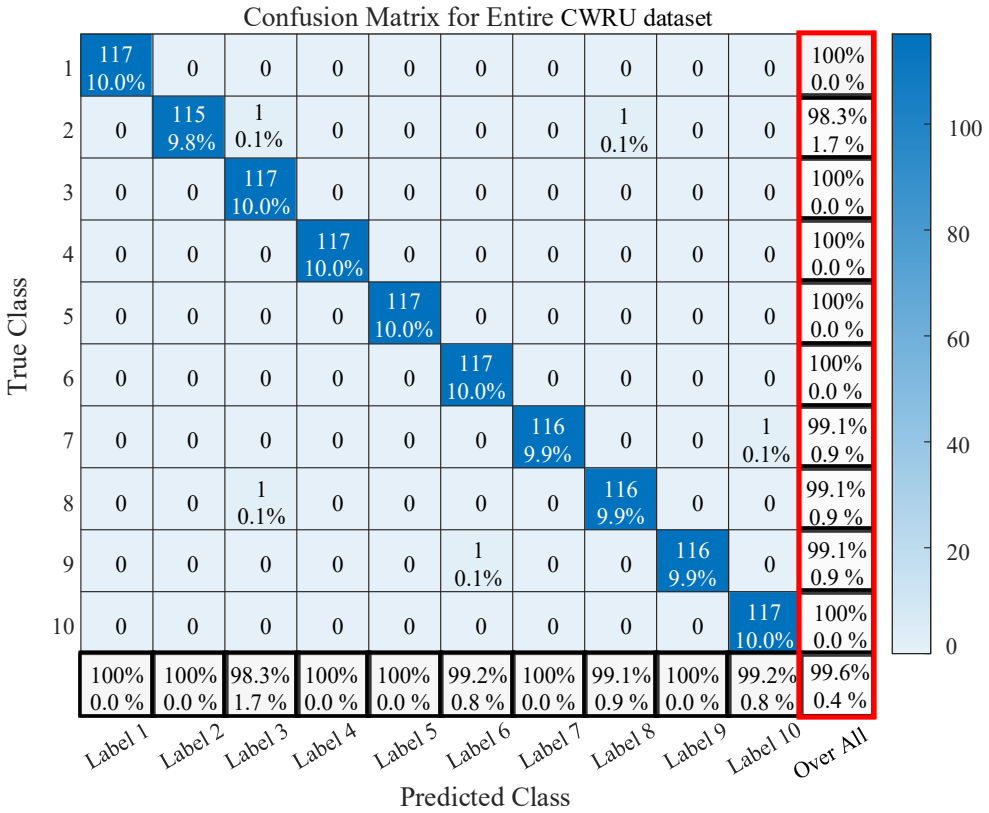


Figure 9. Confusion matrix of the proposed algorithm. (Label info Table 3)

Using predicted and actual classes, Figure 9 illustrates the overall accuracy of the dataset, demonstrating the efficacy of a confusion matrix tailored to each fault type to evaluate the efficacy of the proposed algorithm. Using this matrix, we have assessed how effective the algorithm is in distinguishing between fault types, which achieves 99.60% classification accuracy for the CWRU dataset. These findings, with an average 10-fold CV training accuracy of 100%, highlight the model's robust performance across diverse testing scenarios.

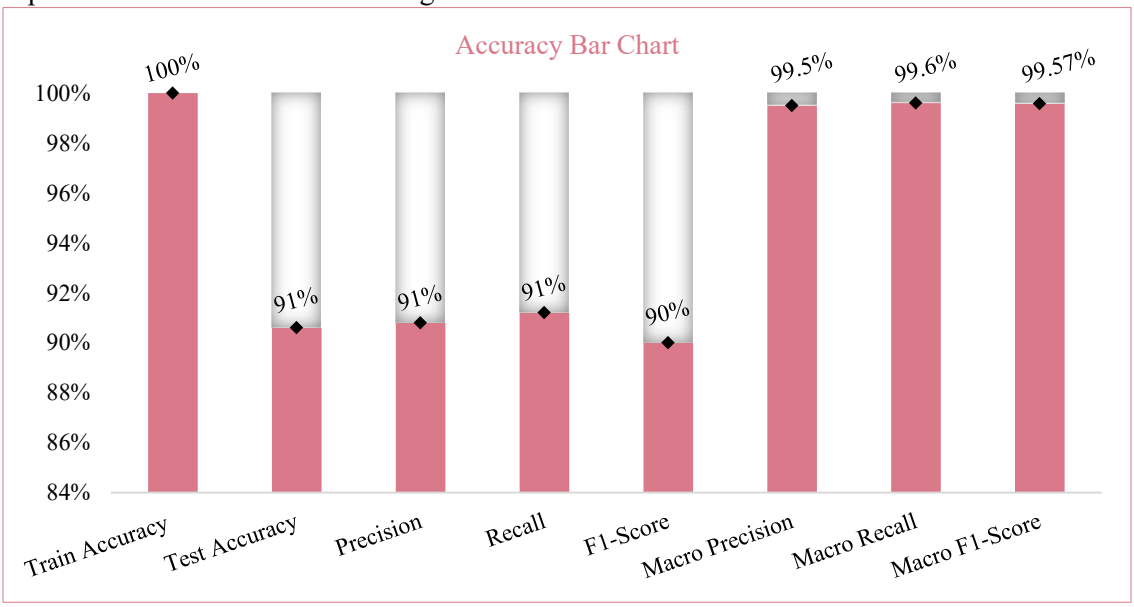


Figure 10. Accuracy Bar Chart

A bar chart in Figure 10 illustrates the importance of macro values when evaluating a classification model's performance in a variety of fault categories, especially in datasets with uneven class distributions. According to its macro precision score of 99.50%, macro recall score of 99.60%, and macro F1-Score of 99.57%, this model is reliable and well-balanced in classification. To further highlight model robustness in complex classification scenarios, training and testing accuracies represent an average of 100% for training and 91% for a 10-fold cross-validation test. These well-tested metrics

ensure that all classes are assessed uniformly. Thus preventing any single metric from skewing the overall performance assessment.

4.2 Evaluation under Noisy Events

Considering the working conditions under which mechanical equipment operates, data collected under variable conditions might be contaminated with severe noise conditions. To rigorously evaluate the proposed algorithm's robustness, it is deliberately injected with Gaussian white noise under extreme noise conditions at signal-to-noise ratios (SNRs) of 10 dB, 6 dB, and 2 dB. This way, the algorithm's performance is tested under realistic and challenging acoustic conditions, ensuring it can handle diverse acoustic scenarios. The classification results under different noise levels are presented in Table 8.

Table 8. The recognition rate of faults under noise conditions

The recognition rate of faults under noise conditions				
Noise (dB)	Entire Dataset Accuracy (%)	Macro Precision (%)	Macro Recall (%)	Macro F1-Score (%)
10 dB	99.12	99.23	99.24	99.23
6 dB	98.86	98.63	98.66	98.64
2 dB	97.10	97.10	97.20	97.12
Average (%)	98.36	98.32	98.40	98.33

The high fault classification accuracy of 98.36% for the entire (CWRU) dataset demonstrates the robustness of the proposed feature against noise. In comparison, when the denoising Wiener Filtering (WF) technique is applied [53], defined as WF $\left(H(f) = \frac{P_{xx}(f)}{P_{xx}(f)+S_{vv}(f)}\right)$ and trained for 10 dB noise levels, dataset accuracy was 98.12%. This approach removes noise using a periodogram to estimate and simplify noise power spectrums $S_{vv}(f)$ using the signal's median power $P_{xx}(f)$. We have enhanced signal clarity by mitigating additive Gaussian noise by leveraging FFT with WF filtering and IFFT conversion. We also compared with the robust discrete wavelet transform-based denoising technique (DWT-DT) [54]. However, samples contaminated with 10 dB noise levels received a classification accuracy of 97.95%, macro precision of 97.95%, macro recall of 97.97%, and macro F1-Score of 97.95% for the DWT-DT model. Keeping this in mind, the proposed features perform well.

To further highlight the resistance of our proposed framework, we have tested the baseline shallow ML and DL models under low SNR conditions by injecting Gaussian white noise into the vibration signals. The results are tabulated in Table 9. Decrement in accuracy (%) measures the average performance drop under noise (10 dB, 6 dB, 2 dB) compared to clean data. These additional experiments and analyses demonstrate that our framework achieves state-of-the-art performance with minimal decrement (1.3%), outperforming SVM (14.55%) and KNN (18.63%) due to robust features and optimized hyperparameters.

Table 9. Comparisons of proposed framework and baseline models under low SNR conditions

Comparisons of proposed framework and baseline models on the CWRU dataset under low SNR conditions (10 dB, 6 dB, and 2 dB)						
Algorithms	Selected Features	Entire Dataset Accuracy	Accuracy @ 10 dB	Accuracy @ 6 dB	Accuracy @ 2 dB	Decrement in Accuracy (%)
SVM	Normalized features	80.68%	78.55%	73.10%	55.18%	↓14.55%
KNN	Normalized features	75.21%	68.50%	62.30%	52.80%	↓18.63%
LSTM	Proposed Features	98.03%	97.76%	94.20%	88.50%	↓04.60%
Bi-LSTM	Proposed Features	99.32%	98.46%	96.98%	94.12%	↓02.82%
Proposed Algorithm +	Proposed features	99.60%	99.12%	98.86%	97.10%	↓01.30%

4.3 Comparison with Shallow Networks

In this section, we compared traditional shallow networks such as Support Vector Machine (SVM) with a Medium Gaussian kernel, Cosine K-Nearest Neighbors (KNN), Logistic Regression kernel (LRK), and Naïve Bayes Classifier (NBC) against a proposed algorithm using PCA & BO-DAE features on the CWRU dataset. The assessment is based on classification accuracy under normalized and proposed feature sets with robust 10-fold CV. The classification accuracy, macro precision, recall, and F1-Score for all shallow networks are tabulated in Table 10. Considering the dynamics and complicated operating environments for bearing, these shallow networks are prone to mis-convergence and lower accuracy rates.

Table 10. Comparisons of diagnostic results with shallow networks with 10-fold CV.

Comparisons of diagnostic results with shallow networks						
Algorithms	Selected Features	Entire Dataset Accuracy (%)	Drop off in Accuracy (%)	Macro Precision	Macro Recall	Macro F1-Score
SVM	Normalized features	80.68%	↓ 18.92%	80.68%	84.43%	79.91%
KNN	Normalized features	75.21%	↓ 24.39%	75.39%	79.82%	75.28%
Logistic Regression	Normalized features	77.26%	↓ 22.34%	77.26%	78.51%	77.08%
Naïve Bayes	Normalized features	51.54%	↓ 48.06%	51.54%	57.42%	52.11%
SVM	Proposed features	50.10%	↓ 49.50%	50.10%	53.34%	49.71%
KNN	Proposed features	77.18%	↓ 22.42%	77.44%	80.50%	76.84%
BO-SVM	Proposed features	84.20%	↓ 15.40%	87.10%	84.20%	83.62%
BO-SVM	Normalized features	93.76%	↓ 05.84%	94.42%	93.76%	93.73%
Proposed Algorithm	Proposed features	99.60%		99.50%	99.60%	99.57%

We tested 12-15 shallow networks for given scenarios and presented models that stood out, but our proposed model still outperformed them all. Figure 11 presents the confusion matrix for the SVM model with normalized features. It provides better results than other shallow networks. However, a 49.5% decrease in accuracy is observed for SVM with the proposed features. On the contrary, despite being less adept at handling reduced features, the KNN performed well with the proposed features. It underlines the importance of feature selection for improving classification.

The choice of hyperparameters, such as the kernel function, box constraint, and kernel scale, plays a crucial role in enhancing SVM classification performance [55]. For validation, BO-optimized SVM models using the proposed features achieved 80.68% accuracy with a Gaussian kernel function, a kernel scale of 29.26, and a box constraint of 156.86. The box constraint determines the trade-off between maximizing generalization and minimizing classification errors. A high value of 156.86 makes the model more complex and imposes stronger penalties for misclassifications. However, when BO employed a polynomial kernel function with a scale of 1 and a box constraint of 9.93, it suggested that SVM used a non-linear boundary to separate the data better. The lower box constraint allowed the model to generalize better with low penalties for misclassifications, leading to a substantial improvement in accuracy of 93.76%.

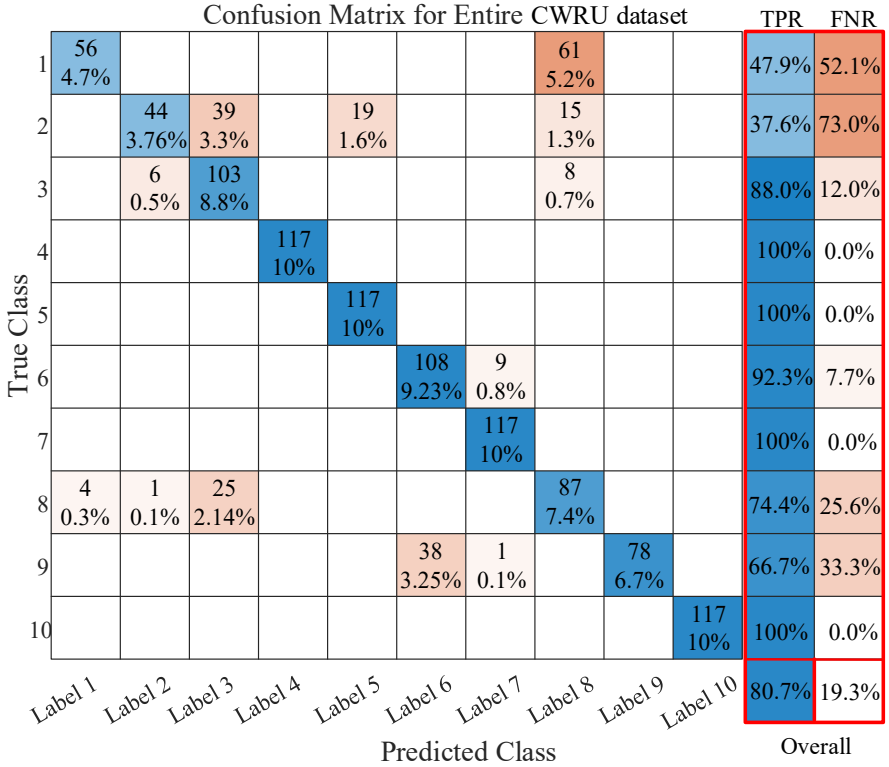
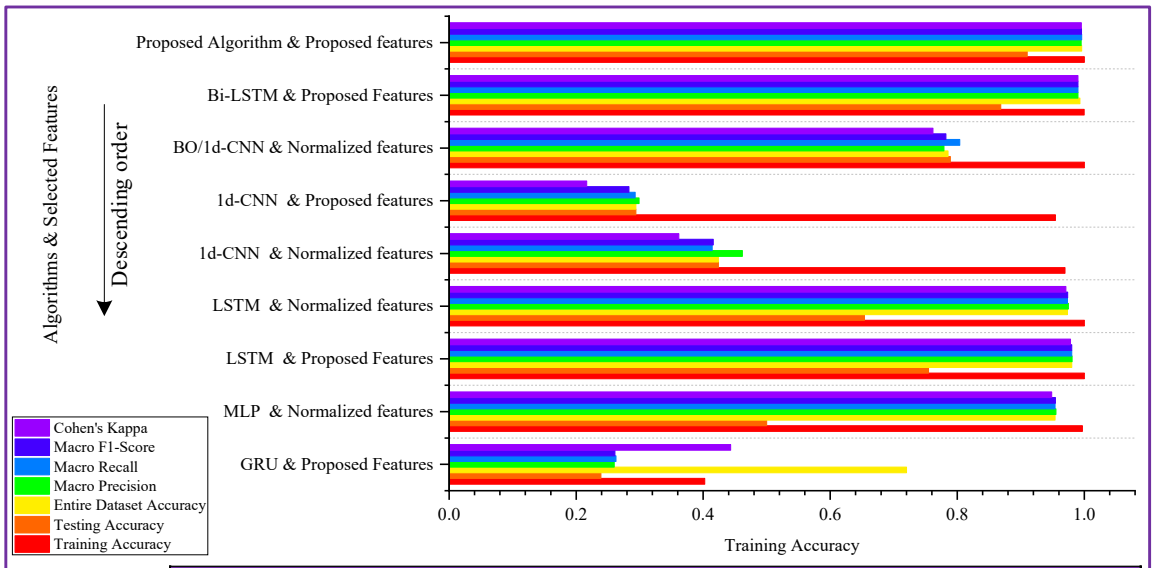


Figure 11. Confusion matrix of the SVM with normalized vibrational features.



Comparisons of diagnostic results with deep neural networks

Ascending order

Algorithms	Selected Features	Training Accuracy	Testing Accuracy	Entire Dataset Accuracy	Macro Precision	Macro Recall	Macro F1-Score	Cohen's Kappa
GRU	Proposed Features	40.23%	23.90%	72%	26.00%	26.27%	26.10%	44.33%
MLP	Normalized features	99.70%	50.00%	95.3846%	95.539%	95.385%	95.462%	94.872%
LSTM	Proposed Features	100%	75.47%	98.0342%	98.081%	98.034%	98.037%	97.816%
LSTM	Normalized features	100%	65.40%	97.351%	97.50%	97.35%	97.37%	97.10%
1d-CNN	Normalized features	96.952%	42.393%	42.393%	46.161%	41.414%	41.578%	36.162%
1d-CNN	Proposed features	95.44%	29.402%	29.402%	29.886%	29.275%	28.311%	21.667%
BO-1d-CNN	Normalized features	100%	78.923%	78.547%	77.919%	80.405%	78.219%	76.186%
Bi-LSTM	Proposed Features	99.99%	86.8376%	99.32%	99.00%	99.00%	99.00%	99.00%
Proposed Algorithm	Proposed features	100%	91%	99.60%	99.50%	99.60%	99.57%	99.53%

Figure 12. Comparison of diagnostic results with deep neural networks with 10-fold CV.

4.4 Comparison with Deep Neural Networks

In this evaluation, we compared the performance of various deep neural networks for fault classification tasks. The results are visualized in Figure 12, "Descending order" and "Ascending order" in the table to indicate how the algorithms are arranged according to their stepwise implementation. Apart from other evaluation metrics employed, we included Cohen's Kappa coefficient ($C\kappa$) as an additional measure, Kappa assesses how much two observers agree on categorizing items, effective for subjective and unordered categories. With $C\kappa$ = 99.53%, the proposed algorithm with proposed features demonstrates robust performance.

In models like the gated recurrent unit (GRU) and 1d-CNN, performance falters when applied with proposed features, even after incorporating ReLU activation, normalization, and dropout layers. For instance, the GRU model achieved an average testing accuracy of 23.90%, with $C\kappa$ = 44.33%, reflecting significant challenges in generalizing the learned patterns. The Multi-Layer Perceptron (MLP) model with normalized features and LSTM (both with proposed and normalized features) presented a dichotomy in performance. For illustration, MLP with epoch =100 and batch size =150 obtained via the hit and trial hyperparameters settings cannot be generalized well with a testing accuracy of 50%. LSTM, especially with proposed features, showed superior performance (average training accuracy of 100%, CWRU dataset classification of 98.03%, $C\kappa$ = 97.82%). This indicates that the proposed spatial features integrate well with temporal learning mechanisms.

To further enhance performance, we introduced the BO-1d-CNN model, with 12 optimized hyperparameters and multiple relevant layers. This model outperforms conventional 1d-CNN models with an average testing accuracy of 78.92% and $C\kappa$ = 76.19% [56]. In another case, it is observed that conventional Bi-LSTM (similar architecture to BO tuned Bi-LSTM without optimized hyperparameters) with proposed features also showed notable results, closely following the proposed algorithm with an entire dataset accuracy of 99.32%, testing accuracy of 86.84%, and uniform scores of 99% across precision, recall, and F1-score, along with $C\kappa$ = 99.00%. While analyzing Figure 13, the proposed algorithm underscores the Bi-LSTM model in terms of evaluation metrics, and the average training time for each fold is 50 sec, which is three times less than the Bi-LSTM model (125 sec). This indicates a robust capacity for spatial and temporal feature representation with increased convergence speed and superior fault classification.

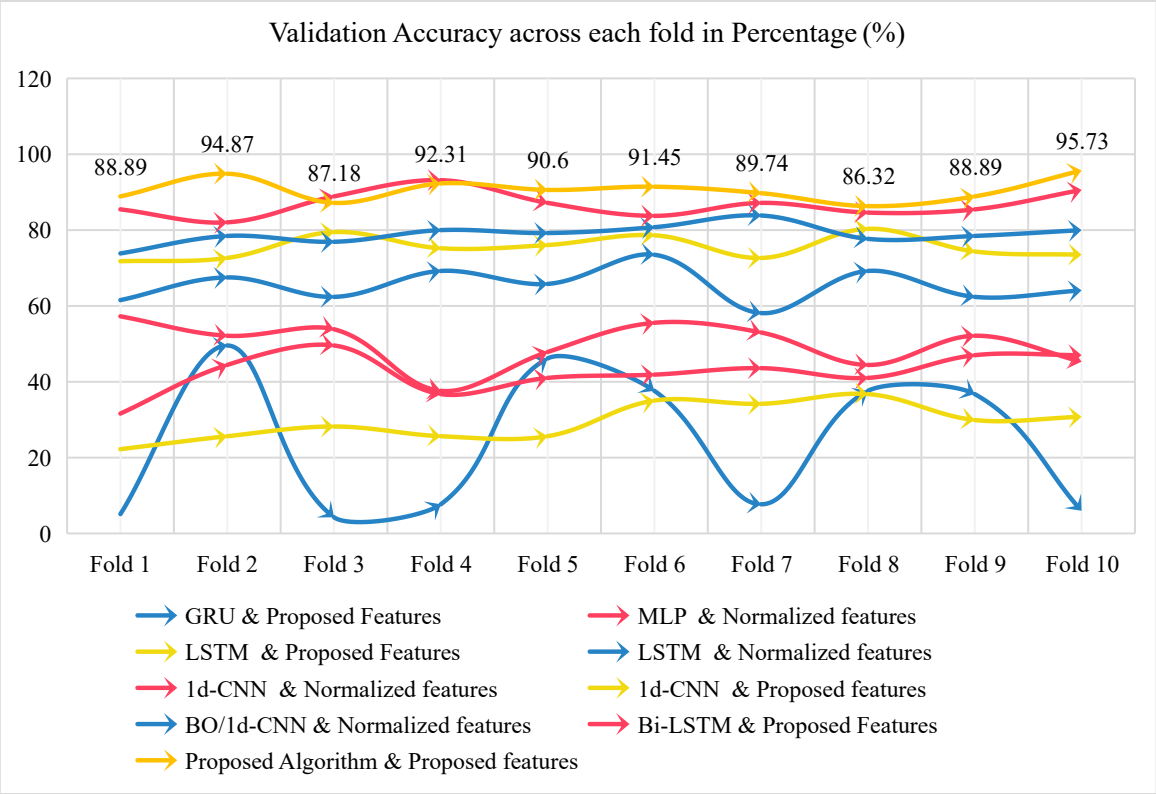


Figure 13. Validation accuracy (testing accuracy) across each deep neural network fold.

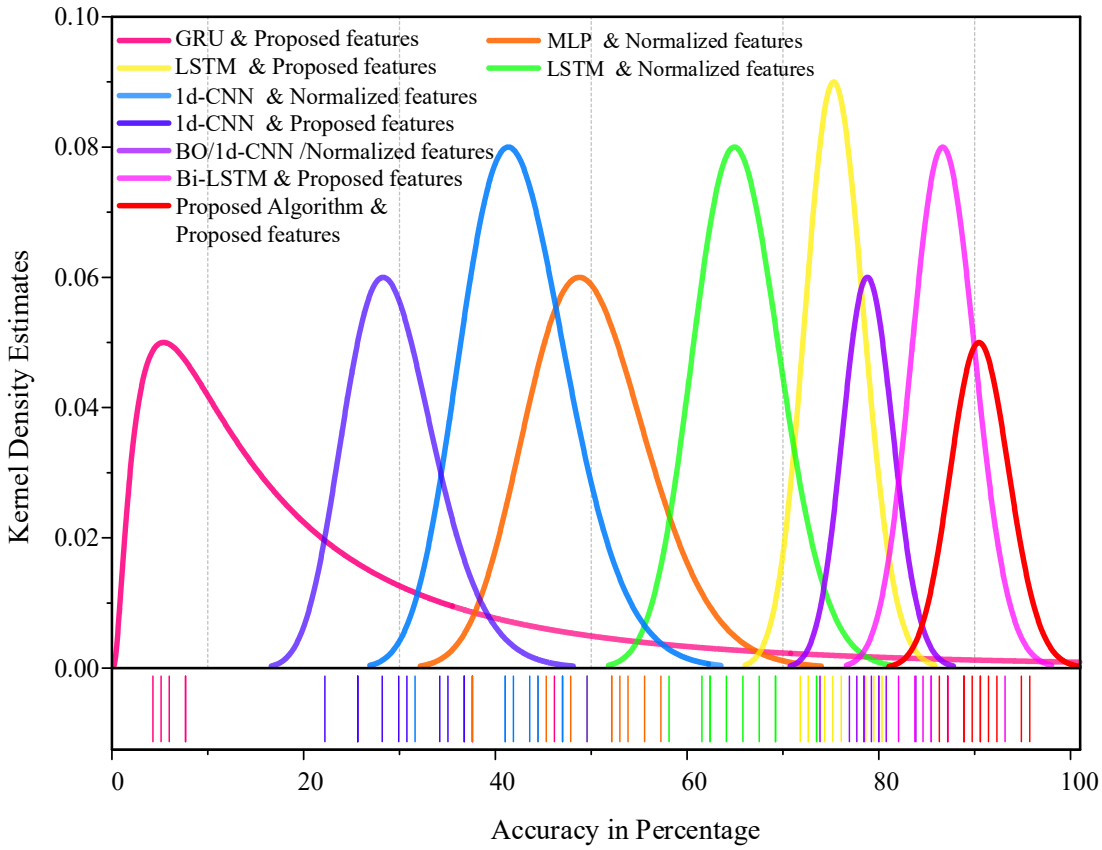


Figure 14. Kernel density estimation of errors for different networks

Kernel density estimation helps visualize algorithm performance across various metrics. Based on Figure 14, the accuracy distribution graph demonstrates that the proposed algorithm with tailored features excels. Kernel distribution is

approximately 100% with sharp peaks, showing high accuracy and consistency. In contrast, the GRU model with the proposed feature exhibits a peak at lower accuracy levels, dipping below 60%, showing poor performance and inconsistency. Meanwhile, the LSTM model with proposed features has a median accuracy of around 80%. However, the LSTM model for the normalized features has a median peak of around 65%, illustrating the importance of feature selection. In addition to a tighter distribution of around 90% with better performance, the Bi-LSTM model with proposed features surpasses the aforementioned LSTM models. On the other hand, 1d-CNN and BO-1d-CNN models with standard features have shown a broader accuracy distribution. It determines that the proposed BO enhances the accuracy of the 1d-CNN model. Secondly, the median accuracy falls below 50% for the MLP model, emphasizing the need for feature selection. Overall, this comparative analysis highlights the importance of feature learning and optimization.

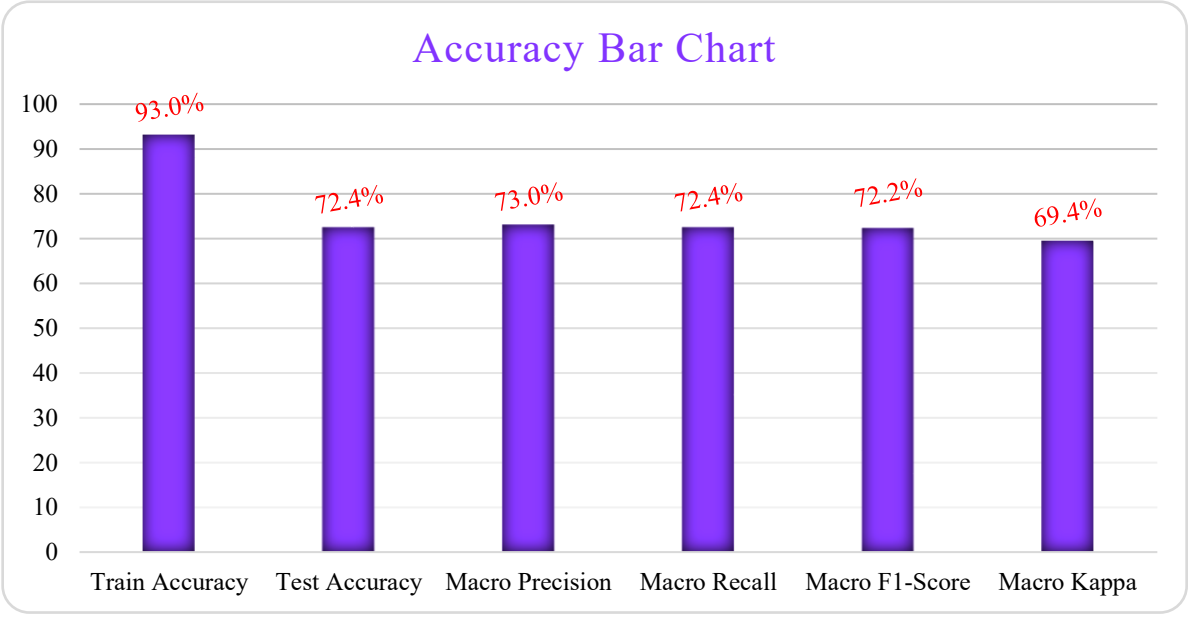


Figure 15. Accuracy Bar Chart with the replacement of the PCA component with 1d-CNN for 10-fold CV.

While DL models like 1d-CNN excel at hierarchical feature learning, PCA was preferred due to no requirement for training and avoids the risk of overfitting with small and limited datasets. To evaluate the trade-offs between PCA and deep hierarchical extractors, PCA was replaced with 1d-CNN's latent features and combined with BO-DAE's outputs ($\ddot{X}_{Total} = \{\ddot{X}_{1d-CNN}, \ddot{X}_{AE}\}$). With an initial learning rate of 0.001, 100 epochs, and a mini-batch size of 32, the 1d-CNN took 32 seconds to train and extract features. Figure 15 shows that the proposed algorithm achieved training accuracy of 93%, testing accuracy of 72.4%, and macro precision, recall, and F1-Score of 73.0%, 72.4%, and 72.2%. Despite excelling at hierarchical feature learning, the 1d-CNN model demonstrated signs of overfitting to extract features. This analysis highlights the challenge of implementing 1d-CNN to small datasets.

Table 11. Proposed algorithm training and test results with PCA DAE for CWRU datasets with a 10-fold CV

BO-based Bi-LSTM algorithm training and test results with PCA and DAE for CWRU datasets with a 10-fold CV.									
Optimization	**Feature Selection Method**	**Classifier**	**Values**						
			Train Accuracy	Test Accuracy	Macro Precision	Macro Recall	Macro F1 Score	Macro Kappa	Entire Dataset Accuracy (%)
BO	PCA	Bi-LSTM	100%	83.76%	98.90%	98.89%	98.89%	98.77%	98.89%
	BO-DAE	Bi-LSTM	94.00%	62.10%	92.00%	92.00%	92.00%	91.00%	91.71%
	PCA + BO-DAE	Bi-LSTM	100%	91.00%	99.50%	99.60%	99.57%	99.53%	99.60%

4.5 Comparison with Different Feature Selection Models

This section presents the study to compare different feature selection and dimension reduction techniques as individuals and in combination with the proposed encoder. Techniques like t-SNE [52], ICA [46], and random projection (RP) [57] offered slightly improved results when used with the proposed encoder and trained with the proposed algorithm. PCA, along with the encoder, helps to capture variance much better than ICA and preserve better information than t-SNE, as shown in Figure 16. Subsequently, techniques like isometric feature mapping (ISO MAP) [57] and kernel PCA [47] tend

to overfit, whereas sequential feature selection (FS) underperforms for all metrics [58]. All these results were obtained using MATLAB® 2023b, a 13th Generation Intel core(TM) i9-13900H@ 2.60GHz (24 Cores) with 32 GB RAM and NVIDIA RTX 2000 Ada Generation Laptop GPU running on Windows 11 Pro 22H2. Under these circumstances, the proposed model does not overfit while avoiding sensitivity and mal-operation issues. This is because it contains a sophisticated feature selection routine with optimized hyperparameters, which prevents maloperation from occurring.

Table 11 compares the performance of PCA and BO-DAE in terms of their impact on proposed model performance. It was observed that PCA performed better than BO-DAE features; however, it tends to struggle to extract non-linear patterns. Therefore, when BO-DAE is assembled with PCA, it leads to better and balanced model performance.

Comparisons of diagnostic results with different features

Selected Features	Training Accuracy	Testing Accuracy	Entire Dataset Accuracy	Macro Precision	Macro Recall	Macro F1-Score	Cohen's Kappa
Encoder+ICA	88.70%	45.90%	79.74%	79.74%	80.90%	79.90%	77.51%
Encoder +tNSE	100%	66.33%	95.98%	95.98%	96.00%	95.98%	95.54%
Encoder + RP	100%	66.50%	95.64%	95.64%	95.70%	95.63%	95.16%
Sequential FS	31.71%	22.65%	30.86%	32.22%	30.85%	29.27%	23.33%
ISO MAP	99.92%	40.43%	93.50%	93.25%	93.25%	93.23%	92.50%
kernel PCA	98.24%	40.90%	91.03%	91.03%	91.28%	91.09%	90.03%

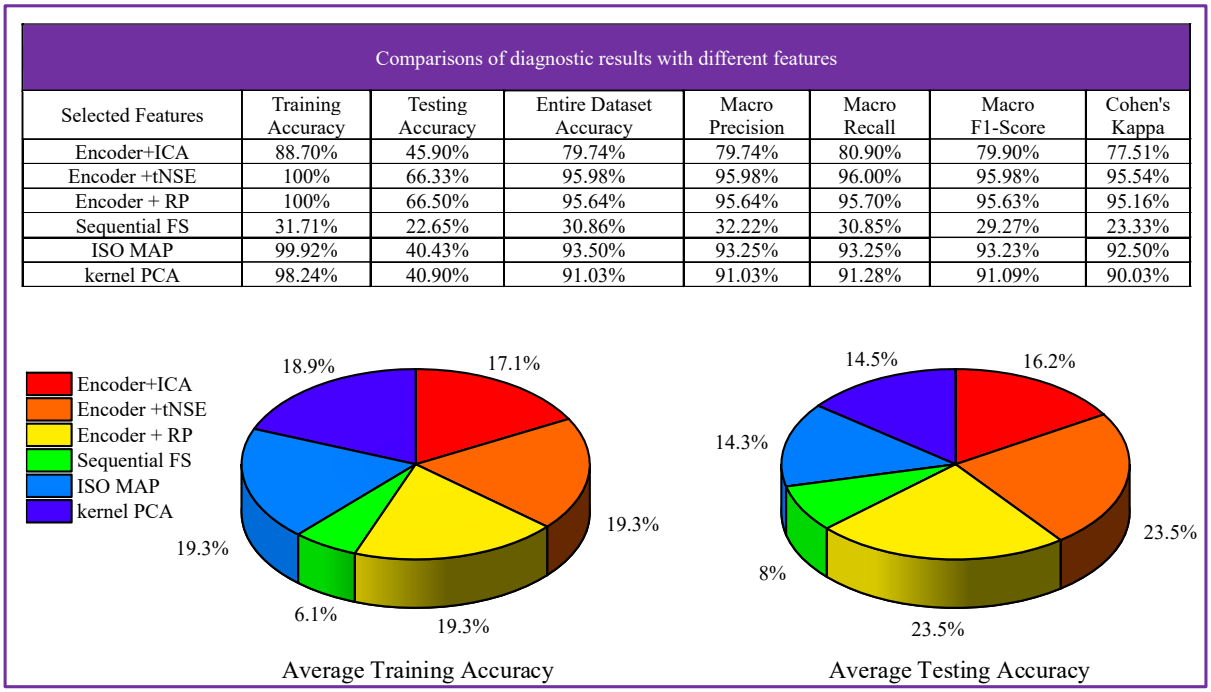
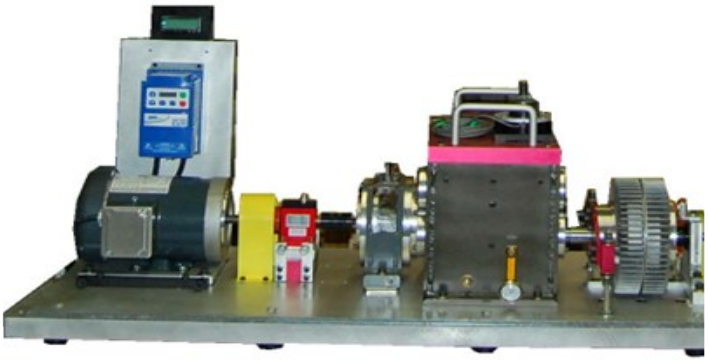


Figure 16. Comparison of Feature selection models trained with 10-fold CV and the proposed algorithm.



DDS experimental platform

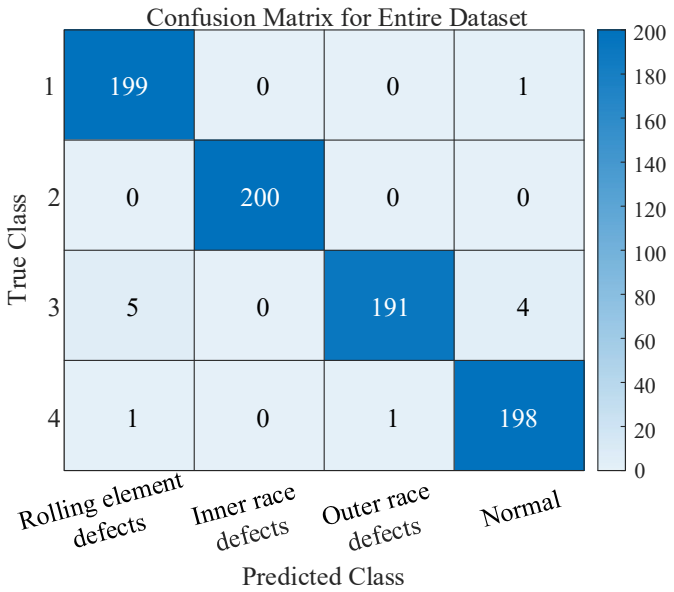


Figure 17. Confusion matrix of the proposed algorithm with 10-fold CV.

4.6 Bearing Fault Diagnosis using Drivetrain Dynamics Simulator (Test 2)

This work also used SpectraQuest's Drivetrain Dynamics Simulator (DDS) to verify the proposed algorithm's efficacy [47]. This simulator simulates industrial drivetrains for educational and experimental purposes. It uses two parallel-

shaft gearboxes with ER-16K bearing models, connected to opposing shafts through magnetic brakes by a bearing loader. The wheel has a diameter of 15.16 mm and a diameter of 3.125 mm for the rolling elements, a total of nine, and a zero-contact angle. We conducted experiments with a sampling frequency of 12.8 kHz, a motor of 20 Hz, and no load condition, and we generated a matrix size of 200,000 *1. There are 800 samples, 200 for each condition, with a sample size of 1024 for normal, inner, outer, and rolling element defects. Figure 17 illustrates the overall classification accuracy of 99% (with 99.50%, 100%, 95.5%, and 99% for each class). Hence, with training accuracy of 100%, the macro average for precision = 98.50%, recall =98.52%, F1-Score=98.50%, and $C\kappa$ = 98%, the proposed algorithm demonstrates proficient results.

The proposed algorithm helps to capture long-term dependencies and identify characteristic variations at optimal learning rates. Despite the complexity of the rolling bearing network, it fully exploits the intrinsic characteristics of Bi-LSTM, such as pattern classification and high generalization capability. In comparison, the earlier mentioned Bi-LSTM model that closely mimics the proposed algorithm achieves 98% classification accuracy of the entire dataset, with the macro average for precision = 98.00%, recall =98.02%, F1-Score=98.00%, and $C\kappa$ = 97%.

4.7 Computational Complexity Breakdown

The proposed framework incorporates three main components: PCA, BO-DAE, and BO-based Bi-LSTM. For the CWRU datasets, PCA performs the SVD of the input matrix to reduce the features from 1024 to 276 while keeping 95% of the variance. This step is computationally light, with a processing time of approximately 0.05 ms per sample on a personal computer.

However, optimizing a large set of hyperparameters to adjust the DL architecture based on the data characteristics can be computationally intensive. This process often requires days or even weeks of computation on specialized high-performance hardware [59]. The BO-DAE process completed 30 BO iterations (parallelized) in 10,598.6 seconds for 6 hyperparameters, including a 5-fold CV for each iteration of BO. In addition, rigorous BO-driven optimization was applied to optimize 9 hyperparameters for the Bi-LSTM, with an extensive search range. The proposed parallel BO achieves 63.33% faster convergence than the sequential BO method by evaluating multiple configurations simultaneously. The final model involves training the Bi-LSTM model for 173 epochs (selected by BO) with a 10-fold CV and an evaluation time of 1.82 seconds per iteration. Once the BO training with parallel setting was completed, the optimized sparse DAE and Bi-LSTM models took approximately 1.32×10^{-3} seconds to process a single 1024-point raw signal, making the system suitable for real-time fault diagnosis.

In the literature, as tabulated in Table 12, particle swarm optimization (PSO) [42], grid search (GS) [44], and BO [43] have been used to optimize the architecture of shallow machine learning algorithms with a limited number of hyperparameters. However, PSO and GS are not well-suited for handling a large number of hyperparameters with an extensive search range [55]. For deep learning algorithms, such as the deep convolution-gated recurrent unit [45], BO has been employed to optimize five hyperparameters with a limited search range and without extensive validation. In contrast, to the best of the authors' knowledge, this is the first attempt where extensive validation is conducted with a broad search range of hyperparameters for rolling bearing fault classification. In future work, lightweight surrogate models for BO will be explored further to reduce computational overhead while retaining accuracy. Plus, adaptive and semi-supervised studies will be explored as well [60-62].

Table 12. Comparative Efficiency of BO vs. Existing Methods for Hyperparameter Tuning in Fault Diagnosis

Comparative Efficiency of BO vs. Existing Methods for Hyperparameter Tuning in Fault Diagnosis		
Metric	**Proposed Framework**	**Existing Methods ([42], [43], [44], [45])**
Hyperparameters	15 parameters (BO-DAE+ BO based Bi-LSTM)	⩽5 parameters (e.g., SVM kernels, shallow networks)
Search Range	4–5 orders of magnitude	Narrow
Validation Rigor	5-fold + 10-fold CV	Limited validation (e.g., single hold-out)
Training Time	Higher	Faster but lower accuracy

Table 13 compares the benchmark performance of the BO-driven framework against existing models. Liao *et al.* [63] achieve higher accuracy than the proposed framework, but it requires twice the input size (2048 data points vs. 1024 in our framework). When compared to other state-of-the-art models, our framework stands out for its full autonomy and reliability than [64] and [43]. This makes it more efficient, reliable, and user-friendly as an off-the-shelf device through its autonomous feature extraction and end-to-end optimization, setting a new standard for predictive maintenance and fault diagnosis in industrial applications.

Table 13. Comparison with Existing State-of-the-Art Models

Accuracy Benchmark: Autonomous BO-Driven Framework vs. State-of-the-Art Models on CWRU Bearing Data						
No. of Series	**Classification Models**	**Signal Decomposition (SD)**	**Extracted Features**	**Fully Autonomous**	**Hyperparameter Tuning**	**Values (%)** *Dataset Accuracy*
Ref [63]	CNN, Transfer Learning	Wavelet Convolution	Autonomous	×	×	99.73%
Ref [64]	ANN	Variational Mode SD	Energy Features	×	×	99.30%
Ref [43]	BO-SVM	—	Normalized Features	×	×	93.76%
Proposed	PCA, BO-DAE/Bi-LSTM	Autonomous	Autonomous	✓	✓	99.60%

4.8 Experimental Validation of BO Advantages

Table 14 compares the performance of the proposed BO method with baseline optimization techniques for the CWRU datasets. As an initial step, we employed a Bi-LSTM model for grid search (GS) to evaluate in sequence all potential combinations of hyperparameters within a predefined search space. Although GS guarantees that no potential configuration is overlooked, it becomes high resource-intensive when dealing with a large set of hyperparameters. Subsequently, our framework was unable to utilize this approach [55]. BO and random search (RS) are more computationally efficient in high-dimensional space. With a random approach, RS provides a balance of exploration and exploitation. However, the probabilistic nature of BO (30 iterations) gives better results than RS (30 iterations) in our case. Furthermore, the proposed BO with a batch size of 8, GP kernel of Matérn 5/2 and $\dot{\beta}$ =0.75 reduces optimization time by 63.33 % compared to *EI* and 57.63 % compared to *EIpS* with better entire dataset accuracy.

Table 14. Proposed BO method vs. Baseline Methods with 10-fold CV

Proposed BO method vs. Baseline Methods						
Optimization Method	**Time of Convergence (Hours)**	**Decline in Time of Convergence**	**Number of Iterations**	**Training Accuracy**	**Testing Accuracy**	**Entire Dataset Accuracy (%)**
Grid Search (GS)	×	×	*Not applicable*	×	×	×
Random Search (RS)	3.13	↓ 11.82 %	30	100%	87%	87.10%
Standard BO (*EI*)	7.50	↓ 63.33 %	30	100%	91%	99.15%
Standard BO (*EIps*)	6.49	↓ 57.63 %	30	100%	94%	99.20%
Proposed BO (*EIps*+)	2.76	*Not applicable*	30	100%	91%	99.60%

5) Conclusion

This paper introduces an advanced diagnostic approach for rolling element bearing faults using a BO-based method designed for adaptive feature extraction of bearing faults. Unlike other DL solutions, the proposed BO-DAE and BO-Bi-LSTM emphasize optimizing parameters and the backbone network structure. Through a probabilistic optimization algorithm, this approach refines network structure and hyperparameters that strengthen generalization and feature extraction capabilities. The incorporation of parallel BO with *EIps*+ proved essential in speeding up hyperparameter tuning and optimizing computational resources while preserving high model performance. The rigorous evaluation highlights the effectiveness of our approach, with metrics such as a macro precision of 99.50%, recall of 99.60%, F1-Score of 99.57%, and Cohen's Kappa metric (Cκ) of 99.53%. These metrics highlight the high accuracy and reliability of our model in accurately classifying bearing faults. Furthermore, comparative analysis against shallow and deep learning models reveals the superiority of our approach in terms of efficiency and accuracy. Our method outperforms these models across various performance metrics, reinforcing its effectiveness in dealing with various engineering challenges in diagnostic scenarios. Overall, the results demonstrate that our proposed framework, integrating PCA, BO-DAE, and BO-enhanced Bi-LSTM, effectively captures the complex spatial and temporal dependencies in vibrational signals with better generalization and mitigating overfitting issues. This comprehensive approach facilitates precise fault classification, offering a promising solution for predictive maintenance in mechanical systems. These results underscore the potential of our method to significantly reduce downtime, minimize financial losses, and enhance operational safety in industrial settings.

In future work, we plan to explore autonomous data labelling techniques based on active learning to further enhance the robustness of the framework against mislabelled data. Active learning involves the model finding uncertain or ambiguous samples and asking a human for the correct label.

6) References

[1] J. Henry and J. Pomeroy, "The world in 2030," 2018, vol. 75.

[2] A. Biswas, S. Ray, D. Dey, and S. Munshi, "Detection of simultaneous bearing faults fusing cross correlation with multikernel SVM," *IEEE Sensors Journal,* vol. 23, no. 13, pp. 14418-14427, 2023.
[3] S. Wang and Z. Feng, "Multi-sensor fusion rolling bearing intelligent fault diagnosis based on VMD and ultra-lightweight GoogLeNet in industrial environments," *Digital Signal Processing,* vol. 145, p. 104306, 2024.
[4] L. Ciabattoni, F. Ferracuti, A. Freddi, and A. J. I. T. o. I. E. Monteriu, "Statistical spectral analysis for fault diagnosis of rotating machines," vol. 65, no. 5, pp. 4301-4310, 2017.
[5] X. Yu, F. Dong, E. Ding, S. Wu, and C. Fan, "Rolling bearing fault diagnosis using modified LFDA and EMD with sensitive feature selection," *IEEE Access,* vol. 6, pp. 3715-3730, 2017.
[6] A. Rai and S. H. Upadhyay, "A review on signal processing techniques utilized in the fault diagnosis of rolling element bearings," *Tribology International,* vol. 96, pp. 289-306, 2016.
[7] C. Hu, J. Wu, C. Sun, X. Chen, A. K. Nandi, and R. Yan, "Unified Flowing Normality Learning for Rotating Machinery Anomaly Detection in Continuous Time-Varying Conditions," *IEEE Transactions on Cybernetics,* 2024.
[8] A. Biswas, S. Ray, D. Dey, and S. Munshi, "Detection of simultaneous bearing faults fusing cross correlation with multikernel SVM," *IEEE Sensors Journal,* vol. 23, no. 13, pp. 14418-14427, 2023.
[9] Z. Shen, Z. He, X. Chen, C. Sun, and Z. Liu, "A monotonic degradation assessment index of rolling bearings using fuzzy support vector data description and running time," *Sensors,* vol. 12, no. 8, pp. 10109-10135, 2012.
[10] Y. Liao, L. Zhang, W. J. J. o. I. Li, and F. Systems, "Regrouping particle swarm optimization based variable neural network for gearbox fault diagnosis," *Journal of Intelligent Fuzzy Systems,* vol. 34, no. 6, pp. 3671-3680, 2018.
[11] S. Dong, X. Xu, R. J. J. o. t. B. S. o. M. S. Chen, and Engineering, "Application of fuzzy C-means method and classification model of optimized K-nearest neighbor for fault diagnosis of bearing," *Journal of the Brazilian Society of Mechanical Sciences Engineering,* vol. 38, pp. 2255-2263, 2016.
[12] S. S. Roy, S. Dey, and S. Chatterjee, "Autocorrelation aided random forest classifier-based bearing fault detection framework," *IEEE Sensors Journal,* vol. 20, no. 18, pp. 10792-10800, 2020.
[13] C. Abdelkrim, M. S. Meridjet, N. Boutasseta, and L. Boulanouar, "Detection and classification of bearing faults in industrial geared motors using temporal features and adaptive neuro-fuzzy inference system," *Heliyon,* vol. 5, no. 8, 2019.
[14] H. Shao, H. Jiang, Y. Lin, and X. Li, "A novel method for intelligent fault diagnosis of rolling bearings using ensemble deep auto-encoders," *Mechanical Systems Signal Processing,* vol. 102, pp. 278-297, 2018.
[15] R. Rajabioun, M. Afshar, Ö. Atan, M. Mete, and B. Akin, "Classification of Distributed Bearing Faults using a Novel Sensory Board and Deep Learning Networks with Hybrid Inputs," *IEEE Transactions on Energy Conversion,* 2023.
[16] G. Jiang *et al.*, "Rolling bearing fault diagnosis based on convolutional capsule network," *Journal of Dynamics, Monitoring Diagnostics,* pp. 275-289, 2023.
[17] J. Zhang, S. Yi, G. Liang, G. Hongli, H. Xin, and S. Hongliang, "A new bearing fault diagnosis method based on modified convolutional neural networks," *Chinese Journal of Aeronautics,* vol. 33, no. 2, pp. 439-447, 2020.
[18] J. Tang, J. Wu, B. Hu, and J. Qing, "Towards a Fault Diagnosis Method for Rolling Bearings with Time-Frequency Region-Based Convolutional Neural Network," *Machines,* vol. 10, no. 12, p. 1145, 2022.
[19] H. Pan, W. Jiao, T. Yan, A. U. Rehman, A. Wan, and S. Yang, "Combining kernel principal component analysis and spatial group-wise enhance convolutional neural network for fault recognition of rolling element bearings," *Measurement Science Technology,* vol. 34, no. 12, p. 125003, 2023.
[20] Z. Zhu, G. Peng, Y. Chen, and H. Gao, "A convolutional neural network based on a capsule network with strong generalization for bearing fault diagnosis," *Neurocomputing,* vol. 323, pp. 62-75, 2019.
[21] M. Xia, T. Li, L. Xu, L. Liu, and C. W. De Silva, "Fault diagnosis for rotating machinery using multiple sensors and convolutional neural networks," *IEEE/ASME transactions on mechatronics,* vol. 23, no. 1, pp. 101-110, 2017.
[22] Y. Yao *et al.*, "End-to-end convolutional neural network model for gear fault diagnosis based on sound signals," *Applied Sciences,* vol. 8, no. 9, p. 1584, 2018.
[23] S. Gao, S. Shi, and Y. Zhang, "Rolling bearing compound fault diagnosis based on parameter optimization MCKD and convolutional neural network," *IEEE Transactions on Instrumentation Measurement,* vol. 71, pp. 1-8, 2022.
[24] Y. An, K. Zhang, Q. Liu, Y. Chai, and X. Huang, "Rolling bearing fault diagnosis method base on periodic sparse attention and LSTM," *IEEE Sensors Journal,* vol. 22, no. 12, pp. 12044-12053, 2022.
[25] Y. Han, N. Ding, Z. Geng, Z. Wang, and C. Chu, "An optimized long short-term memory network based fault diagnosis model for chemical processes," *Journal of Process Control,* vol. 92, pp. 161-168, 2020.
[26] L. Yu, J. Qu, F. Gao, and Y. Tian, "A novel hierarchical algorithm for bearing fault diagnosis based on stacked LSTM," *Shock Vibration,* vol. 2019, no. 1, p. 2756284, 2019.

[27] F. Zou, H. Zhang, S. Sang, X. Li, W. He, and X. Liu, "Bearing fault diagnosis based on combined multi-scale weighted entropy morphological filtering and bi-LSTM," *Applied Intelligence,* pp. 1-18, 2021.
[28] A. Zhang *et al.*, "Transfer learning with deep recurrent neural networks for remaining useful life estimation," *Applied Sciences,* vol. 8, no. 12, p. 2416, 2018.
[29] W. Mao, W. Feng, Y. Liu, D. Zhang, and X. Liang, "A new deep auto-encoder method with fusing discriminant information for bearing fault diagnosis," *Mechanical Systems Signal Processing,* vol. 150, p. 107233, 2021.
[30] J. Shi *et al.*, "Planetary gearbox fault diagnosis using bidirectional-convolutional LSTM networks," *Mechanical Systems Signal Processing,* vol. 162, p. 107996, 2022.
[31] Z. Zhuang, H. Lv, J. Xu, Z. Huang, and W. Qin, "A deep learning method for bearing fault diagnosis through stacked residual dilated convolutions," *Applied Sciences,* vol. 9, no. 9, p. 1823, 2019.
[32] K. You, G. Qiu, and Y. Gu, "Rolling bearing fault diagnosis using hybrid neural network with principal component analysis," *Sensors,* vol. 22, no. 22, p. 8906, 2022.
[33] L. Yuan, D. Lian, X. Kang, Y. Chen, and K. Zhai, "Rolling bearing fault diagnosis based on convolutional neural network and support vector machine," *IEEE Access,* vol. 8, pp. 137395-137406, 2020.
[34] S.-s. Zhong, S. Fu, and L. Lin, "A novel gas turbine fault diagnosis method based on transfer learning with CNN," *Measurement,* vol. 137, pp. 435-453, 2019.
[35] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data," *Mechanical systems signal processing,* vol. 72, pp. 303-315, 2016.
[36] X. Kong, G. Mao, Q. Wang, H. Ma, and W. Yang, "A multi-ensemble method based on deep auto-encoders for fault diagnosis of rolling bearings," *Measurement,* vol. 151, p. 107132, 2020.
[37] Y. Guo, Y. Zhou, and Z. Zhang, "Fault diagnosis of multi-channel data by the CNN with the multilinear principal component analysis," *Measurement,* vol. 171, p. 108513, 2021.
[38] M. Qiao, S. Yan, X. Tang, and C. Xu, "Deep convolutional and LSTM recurrent neural networks for rolling bearing fault diagnosis under strong noises and variable loads," *IEEE Access,* vol. 8, pp. 66257-66269, 2020.
[39] M. Hakim, A. A. B. Omran, A. N. Ahmed, M. Al-Waily, and A. Abdellatif, "A systematic review of rolling bearing fault diagnoses based on deep learning and transfer learning: Taxonomy, overview, application, open challenges, weaknesses and recommendations," *Ain Shams Engineering Journal,* vol. 14, no. 4, p. 101945, 2023.
[40] X. Chang, S.-p. Yang, S. Li, and X. Gu, "Rolling Element Bearing Fault Diagnosis Based on Multi-objective Optimized Deep Auto-encoder," *Measurement Science Technology,* 2024.
[41] M. Z. Yousaf, H. Liu, A. Raza, A. J. C. J. o. P. Mustafa, and E. Systems, "Deep learning-based robust dc fault protection scheme for meshed HVdc grids," *CSEE Journal of Power Energy Systems,* 2022.
[42] X. Yan and M. Jia, "A novel optimized SVM classification algorithm with multi-domain feature and its application to fault diagnosis of rolling bearing," *Neurocomputing,* vol. 313, pp. 47-64, 2018.
[43] B. Wang, W. Qiu, X. Hu, and W. Wang, "A rolling bearing fault diagnosis technique based on recurrence quantification analysis and Bayesian optimization SVM," *Applied Soft Computing,* vol. 156, p. 111506, 2024.
[44] X. Zhang, Y. Liang, and J. Zhou, "A novel bearing fault diagnosis model integrated permutation entropy, ensemble empirical mode decomposition and optimized SVM," *Measurement,* vol. 69, pp. 164-179, 2015.
[45] M. Jiaocheng, S. Jinan, Z. Xin, and Z. Peng, "Bayes-DCGRU with bayesian optimization for rolling bearing fault diagnosis," *Applied Intelligence,* vol. 52, no. 10, pp. 11172-11183, 2022.
[46] H. Liu, C. Chen, Y. Li, Z. Duan, and Y. Li, *Smart metro station systems: data science and engineering*. Elsevier, 2022.
[47] H. Pan, W. Jiao, T. Yan, A. U. Rehman, A. Wan, and S. Yang, "Combining kernel principal component analysis and spatial group-wise enhance convolutional neural network for fault recognition of rolling element bearings," *Measurement Science Technology,* vol. 34, no. 12, p. 125003, 2023.
[48] F. Laakom, J. Raitoharju, A. Iosifidis, and M. Gabbouj, "Reducing redundancy in the bottleneck representation of autoencoders," *Pattern Recognition Letters,* 2024.
[49] A. Abu-Jasser and M. Ashour, "A backpropagation feedforward NN for fault detection and classifying of overhead bipolar HVDC TL using DC measurements," *Journal of Engineering Research Technology,* vol. 2, no. 3, 2015.
[50] Y. Qiu, W. Zhou, N. Yu, P. J. I. T. o. N. S. Du, and R. Engineering, "Denoising sparse autoencoder-based ictal EEG classification," *IEEE Transactions on Neural Systems Rehabilitation Engineering,* vol. 26, no. 9, pp. 1717-1726, 2018.
[51] M. Z. Yousaf, S. Khalid, M. F. Tahir, A. Tzes, and A. Raza, "A novel dc fault protection scheme based on intelligent network for meshed dc grids," *International Journal of Electrical Power Energy Systems,* vol. 154, p. 109423, 2023.

[52] J. Chen, D. Zhou, C. Lyu, and C. Lu, "Feature reconstruction based on t-SNE: an approach for fault diagnosis of rotating machinery," *Journal of Vibroengineering,* vol. 19, no. 7, pp. 5047-5060, 2017.
[53] R. Jaiswal and D. Romero, "Implicit wiener filtering for speech enhancement in non-stationary noise," in *2021 11th International Conference on Information Science and Technology (ICIST)*, 2021: IEEE, pp. 39-47.
[54] M. Z. Yousaf, M. F. Tahir, A. Raza, M. A. Khan, and F. Badshah, "Intelligent Sensors for dc Fault Location Scheme Based on Optimized Intelligent Architecture for HVdc Systems," *Sensors,* vol. 22, no. 24, p. 9936, 2022.
[55] M. F. Tahir, M. Z. Yousaf, A. Tzes, M. S. El Moursi, and T. H. El-Fouly, "Enhanced solar photovoltaic power prediction using diverse machine learning algorithms with hyperparameter optimization," *Renewable and Sustainable Energy Reviews,* vol. 200, p. 114581, 2024.
[56] M. Hamadache, J. H. Jung, J. Park, and B. D. Youn, "A comprehensive review of artificial intelligence-based approaches for rolling element bearing PHM: Shallow and deep learning," *JMST Advances,* vol. 1, pp. 125-151, 2019.
[57] P. H. Jain and S. P. Bhosle, "A review on vibration signal analysis techniques used for detection of rolling element bearing defects," *SSRG Int. J. Mech. Eng,* vol. 8, pp. 14-29, 2021.
[58] M. R. Islam, M. M. Islam, and J.-M. Kim, "Feature selection techniques for increasing reliability of fault diagnosis of bearings," in *2016 9th International Conference on Electrical and Computer Engineering (ICECE)*, 2016: IEEE, pp. 396-399.
[59] L. Li *et al.*, "A system for massively parallel hyperparameter tuning. arXiv 2018," *arXiv preprint arXiv:1810.05934*.
[60] S. Wang, F. Zhao, C. Cheng, H. Chen, and Y. Jiang, "Threshold-optimized and features-fused semi-supervised domain adaptation method for rotating machinery fault diagnosis," *Neurocomputing,* vol. 613, p. 128734, 2025.
[61] S. Wang, G. Lian, C. Cheng, and H. Chen, "A novel method of rolling bearings fault diagnosis based on singular spectrum decomposition and optimized stochastic configuration network," *Neurocomputing,* vol. 574, p. 127278, 2024.
[62] S. Wang, Y. Ju, C. Fu, P. Xie, and C. Cheng, "A Reversible Residual Network-Aided Canonical Correlation Analysis to Fault Detection and Diagnosis in Electrical Drive Systems," *IEEE Transactions on Instrumentation and Measurement,* 2024.
[63] M. Liao, C. Liu, C. Wang, and J. Yang, "Research on a rolling bearing fault detection method with wavelet convolution deep transfer learning," *IEEE Access,* vol. 9, pp. 45175-45188, 2021.
[64] X. Liang, J. Yao, W. Zhang, and Y. Wang, "A novel fault diagnosis of a rolling bearing method based on variational mode decomposition and an artificial neural network," *Applied Sciences,* vol. 13, no. 6, p. 3413, 2023.