



## OPEN ACCESS

### EDITED BY

Tapasi Bhattacharjee,  
Techno India Group, India

### REVIEWED BY

Mohmad Iqbal,  
Delhi Technological University, India  
Dong Li,  
Chinese Academy of Sciences (CAS),  
China

### \*CORRESPONDENCE

Lana A. Abullah  
✉ lana.abdulrazaq@gmail.com

RECEIVED 09 December 2025

REVISED 23 January 2026

ACCEPTED 23 February 2026

PUBLISHED 17 March 2026

### CITATION

Abullah LA and Rahman CM (2026)  
Advancing bearing fault detection  
through a modified metaheuristic  
optimization approach.  
*Front. Appl. Math. Stat.* 12:1763637.  
doi: 10.3389/fams.2026.1763637

### COPYRIGHT

© 2026 Abullah and Rahman. This is an open-access article distributed under the terms of the [Creative Commons Attribution License \(CC BY\)](https://creativecommons.org/licenses/by/4.0/). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

# Advancing bearing fault detection through a modified metaheuristic optimization approach

Lana A. Abullah<sup>1\*</sup> and Chnoor M Rahman<sup>2,3,4</sup>

<sup>1</sup>Department of Software Engineering, College of Engineering and Computational Science, Charmo University, Sulaymaniyah, Iraq, <sup>2</sup>Department of Computer Science, College of Science, Charmo University, Sulaymaniyah, Iraq, <sup>3</sup>Department of Computer Science, College of Science, Sulaymaniyah University, Sulaymaniyah, Iraq, <sup>4</sup>Computing and Informatic Department, American University of Iraq, Sulaymaniyah, Iraq

**Introduction:** Detecting bearing faults plays a vital role in industrial maintenance since discovering problems early can help avoid unexpected breakdowns and expensive production losses. Yet, spotting these faults in their initial stages is still difficult because vibration signals are often complex and change over time.

**Methods:** In this study, optimized Mel Frequency Cepstral Coefficients (MFCC) feature extraction approach enhanced through a modified FOX optimization algorithm. The enhancement focuses on fine-tuning MFCC hyperparameters to maximize the discriminative power of extracted features for fault detection tasks. The proposed Enhanced FOX (EFOX) algorithm integrates different random distribution method and improved exploration–exploitation balance, enabling more effective parameter optimization compared to conventional methods.

**Results:** Experimental evaluations were conducted using benchmark datasets, and the optimized MFCC features were compared against those obtained via standard MFCC settings and other metaheuristic optimization techniques. Results demonstrate that our approach consistently outperforms competing methods in terms of classification accuracy and the robustness of the proposed model was assessed by testing it on two distinct bearing's datasets with different noise ratios including  $-3$  dB and  $-6$  dB.

**Discussion:** The analysis highlights the impact of each of hyperparameter's of MFCC to bearing fault detection.

### KEYWORDS

bearing, fault detection, metaheuristic algorithm, MFCC, optimized feature

## 1 Introduction

Electronic machines are essential in modern industries and transport systems, yet they are becoming more complex and expensive. They play a key role in applications such as aircraft engines, wind farms, metallurgy, and petrochemical plants. However, harsh operating conditions like high humidity, overload, and extreme temperatures often cause malfunctions that can halt entire systems, leading to costly downtime, higher maintenance demands, and safety risks. This highlights the need for fault-tolerant control systems. Rotating machines and induction motors are especially critical, with components such as stators, rotors, shafts, and bearings. Bearings, in particular, are a major focus in condition monitoring since they account for nearly half of induction motor failures reported in industrial studies (1).

The literature shows that bearing condition monitoring can be performed using vibration, acoustic, and temperature signals, either individually or in combination (2). For example, some

studies rely solely on vibration signals (3), Hybrid Signal Decomposition is applied to extract features that are subsequently input into non-time-series models for bearing fault prediction. In contrast, other researchers have adopted multimodal approaches by combining vibration and acoustic signals (4); in such frameworks, STFT-based time-frequency representations are processed using convolutional neural networks and classified with multiclass support vector machines to enhance fault diagnosis performance (5).

Early approaches to bearing fault detection relied mainly on signal processing techniques such as statistical analysis, Fourier Transform (FT), and Wavelet Transform to extract features from vibration signals. These methods required identify patterns or irregularities linked to faults, but their effectiveness was limited by the complex dynamics of bearings and their weak ability to capture non-linear behavior. The introduction of machine learning (ML) marked a major shift, enabling automated feature extraction and pattern recognition. Classical machine learning Algorithms like K Nearest Neighbor (kNN), Support Vector Machines (SVM), Decision Trees, and random forest, followed later by deep learning models like CNN and LSTM, greatly improved detection accuracy by capturing complex non-linear relationships (6). In ML-based fault detection, the feature extraction remains critical, as the model's performance depends heavily on the quality of the features used for training the ML. wide range of feature extraction methods have been proposed for vibration signal analysis, such as Intrinsic Mode Function (IMF) energies, Hilbert–Huang spectrum descriptors, Wavelet decomposition (7), spectrogram-based features, multi-level decomposition energies, and cepstral coefficients (8). Despite their effectiveness, the task of fine-tuning their hyperparameters remains an open and continuous challenge. Many researchers have employed deterministic optimization algorithms, such as gradient descent, to optimize hyperparameters in fault detection models. However, a major limitation of deterministic approaches is their tendency to become trapped in local minima, which prevents the model from achieving the optimal feature representation for accurate fault diagnosis (9). To overcome this problem, recent studies have explored the use of metaheuristic optimization algorithms for hyperparameter tuning. For instance, Abdul et al. utilized the Grey Wolf Optimization (GWO) algorithm to optimize GTCC features for gear fault detection, demonstrating improved classification performance compared to conventional deterministic methods (10).

This work offers two key contributions. First, an improved version of the FOX algorithm is introduced by strengthening its exploration and exploitation capabilities using different randomization strategies, allowing the algorithm to search more effectively and avoid premature convergence. Second, the enhanced FOX (EFOX) algorithm is employed to optimize the hyperparameters of MFCC features, which significantly improves the accuracy and robustness of bearing fault detection.

The rest of the paper is structured as follows: Section 2 provides the background of the study, Section 3 explains the methodology of the proposed model, Section 4 presents and discusses the evaluation of the EFOX algorithm and the results of bearing fault based on optimized MFCC feature, and Section 5 concludes the paper.

## 2 Basics and background

The background section of this study is divided into two main parts. The first part introduces the original FOX optimization

algorithm and describes the specific strategies applied in this work to enhance its performance. The second part focuses on fault detection models including the Echo State Network (ESN) and provides details about the bearing datasets utilized in the experiments, namely the HUST and CWRU datasets, which are employed to evaluate the effectiveness of the proposed model.

### 2.1 Original fox optimization algorithm

The FOX algorithm was inspired by the hunting behavior of red foxes diving into the snow to catch a prey. It was first introduced as an optimization technique by Hardi et al. (21). The FOX algorithm copies the hunting strategies of a pole fox diving into snow to catch its prey. When snow obscures visibility, the red fox begins its hunt by searching randomly to locate prey. It identifies its target by detecting ultrasound emissions from the prey. Using the time difference of the sound waves, the fox computes the distance to reach the prey. Based on this distance, it estimates the optimal jump needed to catch its target. Finally, the fox optimizes its movements by adjusting them randomly while prioritizing minimal time and the most advantageous position (11). In the algorithm, the red fox's random walk represents exploratory behavior, while its response to sound corresponds to the exploitation segment. The distance that sound travels is computed using the speed of sound in air (343 m/s), and additional adjustments were made for accuracy. A magnetic alignment effect inspires the red fox's jumping strategy. It shows a preference for jumping northeast. Research indicates this orientation increases the chance of success to 82%, compared to 18% in other directions (12). The algorithm balances exploration and exploitation, optimizing search efficiency and prey capture likelihood (12). Figures 1, 2 illustrate the red fox's hunting behavior. between the fox and the prey. Consequently, the hunting behavior of the red fox encompasses both the exploration and exploitation phases jumping strategy, as its focus is on randomly exploring the search zone to locate the prey. To ensure that the fox's random movements guide it toward the most favorable location, the search process is refined using two variables: the minimum time variable. ( $T_{\min}$ ) and  $a$ . The values of  $T_{\min}$  and  $a$  are calculated in Equations 1, 2:

$$T_{\min} = \min(t) \quad (1)$$

$$a = 2 * \left( it - \left( \frac{1}{\max_{it}} \right) \right) \quad (2)$$

Where,  $\max_{it}$  is the maximum number of iterations.

Determining both  $T_{\min}$  and  $a$  is essential for steering the search process toward a near-optimal solution. The random exploration of prey by the fox is facilitated through the use of a random vector with a size of (1, dimension). However, to enhance the efficiency of the algorithm, the variables  $T_{\min}$ ,  $A$  and  $r$  are integrated to maintain a balance between the exploration and exploitation stages. The best solution found so far,  $BestX_{it}$  plays a crucial role in influencing the exploration phase. The strategy for exploration, which aims to identify a new position  $New_{Pos}$  within the search space, is demonstrated in Equation 3:

$$New_{Pos} = BestX_{it} * rand(1, dimension) * T_{\min} * a \quad (3)$$

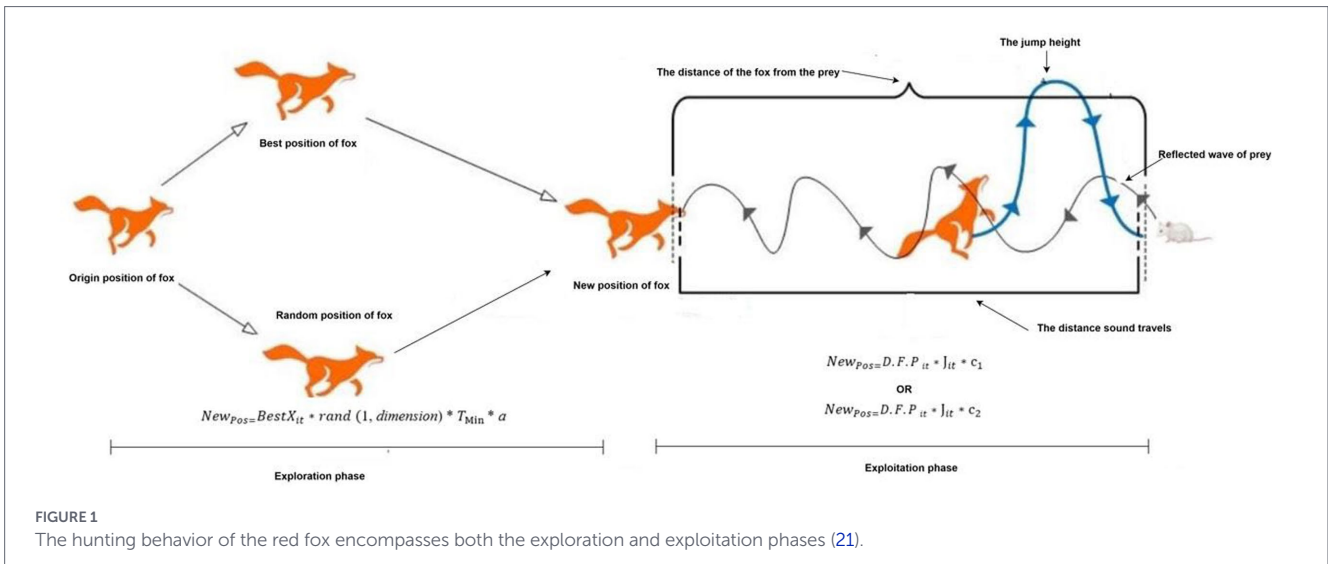


FIGURE 1 The hunting behavior of the red fox encompasses both the exploration and exploitation phases (21).

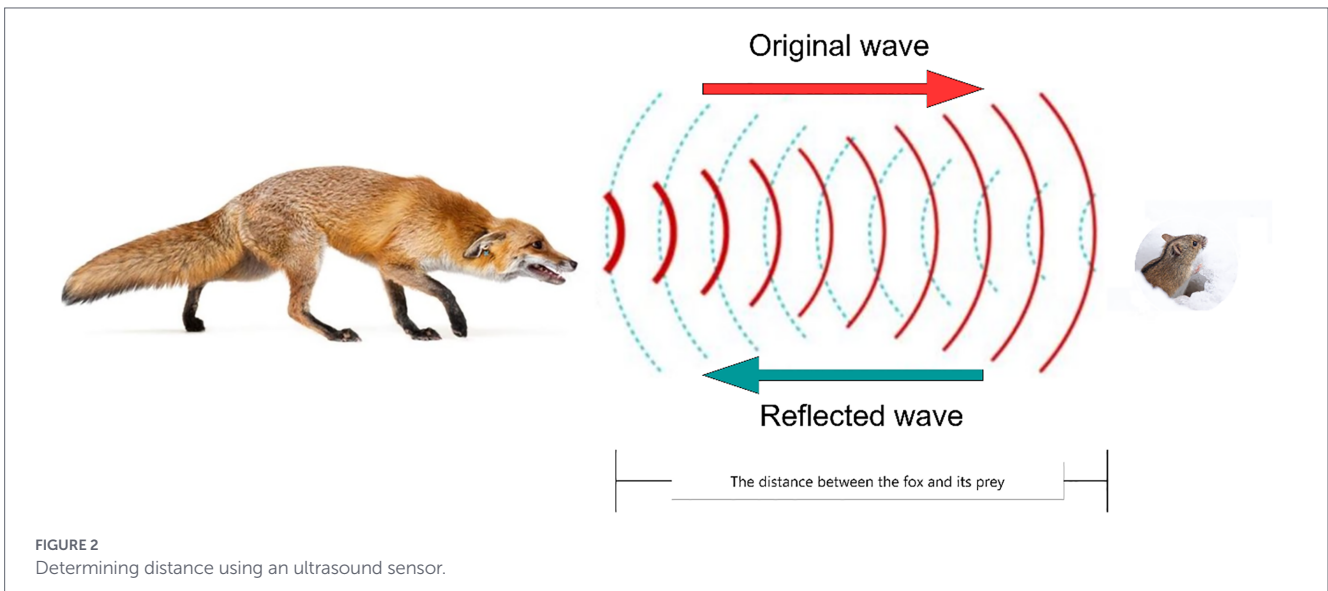


FIGURE 2 Determining distance using an ultrasound sensor.

## 2.2 Adopted techniques

In this research, some techniques are proposed to improve the exploration, exploitation and local optimal of the FOX optimization algorithm. The techniques are randomization and survivor techniques. Both techniques are discussed below.

### 2.2.1 Randomization techniques

The role of the randomization techniques is crucial in the performance of all metaheuristic algorithms because they affect both the exploration and exploitation phases (13). To overcome the lack of exploration of the FOX algorithm, the advantage of the usage of randomization methods is taken in this paper. The randomization leads the FOX algorithm to explore more areas of the search space. The section below explains two random methods that have been used in this enhancement.

### 2.2.2 Levy distribution

The Levy distribution, named after Paul Levy, is characterized as a fat-tailed distribution, meaning its tails have a higher probability of

extreme values compared to distributions like the normal or Cauchy distributions. It is a stable distribution with parameters  $\alpha = 1/2$  alpha and  $\beta = 1$  beta (14). The probability density function (PDF) of the Levy distribution is expressed in Equation 4:

$$f(x) = (\sigma / 2\pi)^{1/2} * 1 / (x - \mu)^{3/2} * \exp(\sigma / 2(x - \mu)) \quad (4)$$

Where

$\mu$ : the location parameter (shifts the distribution along the x-axis).

$\sigma$ : the scale parameter (controls the spread of the distribution).

This PDF is leptokurtic, which means it has heavy tails. Unlike the Gaussian PDF, the Levy distribution's fat tails provide a higher probability for extreme events, making it advantageous for modeling rare occurrences. Notably, both the mean and variance of the Levy distribution are infinite. The characteristic function of the Levy distribution is defined in Equation 5:

$$\Phi_X(w) = \exp\{i\mu w - |\sigma w|^{1/2} (1 - \text{sgn}(w))\} \quad (5)$$

Where:

$\mu$ : the location parameter.

$\sigma$ : the scale parameter.

$i$ : the imaginary unit ( $i^2 = -1$ ).

$w$ : the frequency variable.

$\text{Sgn}(w)$ : the sign function, which is defined as:

For very large values of  $x$ , the exponential term in the characteristic function approaches consequently, the PDF approximates based on Equation 6:

$$(x) \sim \frac{1}{x^{3/2}} = \frac{1}{x^{1/2+1}} \tag{6}$$

This demonstrates that the PDF decreases at a rate of  $1/x^{3/2}$ , Resulting in its fat-tailed behavior (15). Figure 3 shows the Levy distribution and the Gaussian distribution with fatter tails.

### 2.2.3 *t*-distribution

The *t*-distribution, often referred to as Student’s *t*-distribution, is a theoretical probability distribution used when estimating a population mean in cases where the population standard deviation is unknown. It has a symmetric, bell-shaped curve similar to the normal distribution, but it tends to be flatter and wider. As the degrees of freedom (*df*) increase, the *t*-distribution gradually converges toward the standard normal distribution, which has a mean of 0 and a standard deviation of 1.

Essentially, the *t*-distribution illustrates the full range of possible *t*-values that can be obtained from all random samples of a specific size or degree of freedom, and its shape closely resembles that of the normal curve. Figure 4 shows the distribution of randomly generated numbers.

If a variable  $x$  follows a normal distribution with a mean  $\mu$ , and we take a sample of size  $n$  with a sample mean  $\bar{x}$  and a sample standard deviation  $s$ , then the *t*-value derived from this sample follows the

Student’s *t*-distribution with degrees of freedom =  $n - 1$  (16). The formula for *t*-distribution is given in Equation 7:

$$t = \frac{\bar{x} - \mu}{\frac{s}{\sqrt{N}}} \tag{7}$$

Where,  $\bar{x}$  is the mean of the first sample.

$\mu$  is the mean of the second sample.

$\frac{s}{\sqrt{N}}$  =the estimate of the standard error of the difference between

the means.

### 2.2.4 Elitism selection

The concept of elitism is closely tied to the acceptance of newly generated solutions. Various algorithms ensure elitism through different mechanisms. Some maintain a secondary population, or archive, to store the discovered non-dominated solutions. Additionally, algorithms incorporate strategies to involve elite individuals in offspring production. According to the authors, elitism ensures that elite individuals are not excluded from the population’s archive or gene pool in favor of inferior individuals (17). In (18), the authors proposed always including the best individual from the current population in the next generation to prevent its loss due to sampling or operator disruption. This strategy can be extended to retain the top individuals. De Jong’s experiments showed that elitism improves genetic algorithm performance on unimodal functions, but it may lead to premature convergence for multimodal functions.

### 2.2.5 Sinc function

The Sinc function, represented as  $\text{Sinc}(x)$ , is a non-periodic waveform characterized by its interpolating graph. It is an even function

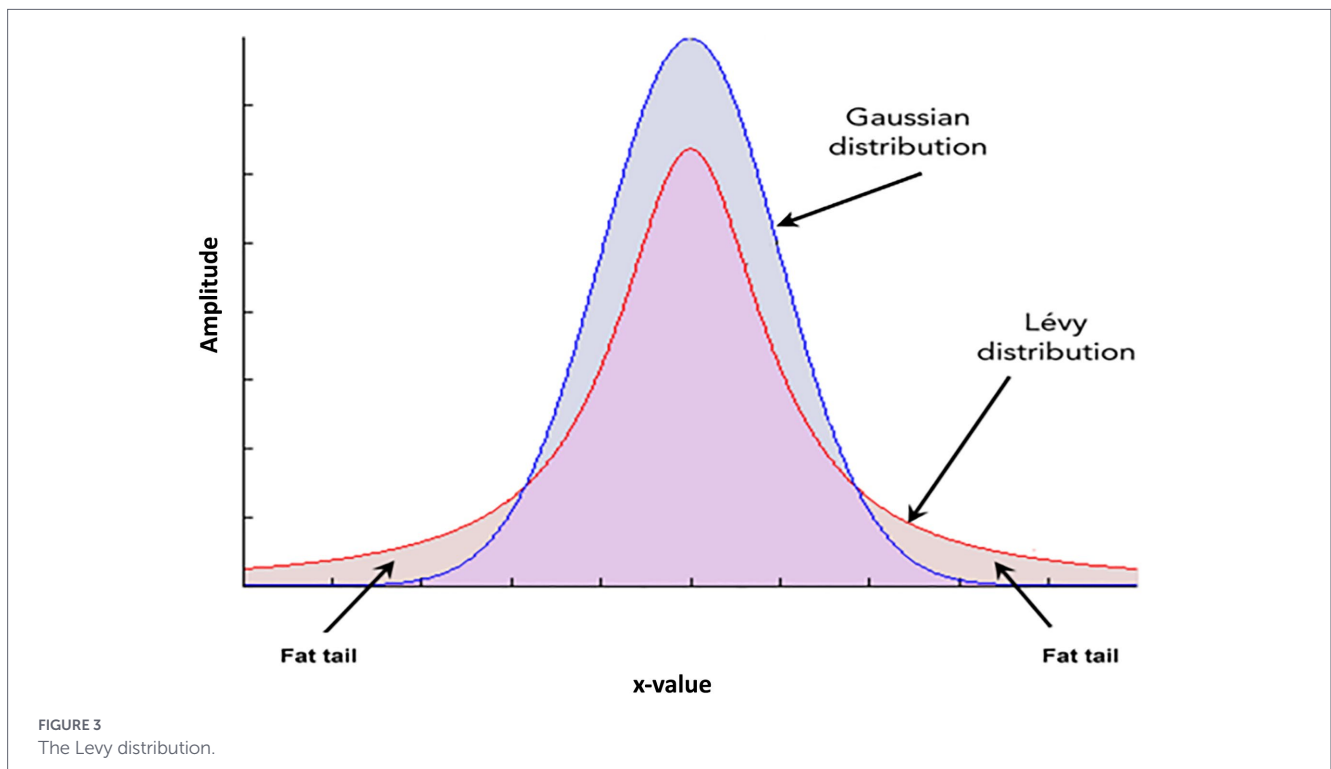


FIGURE 3 The Levy distribution.

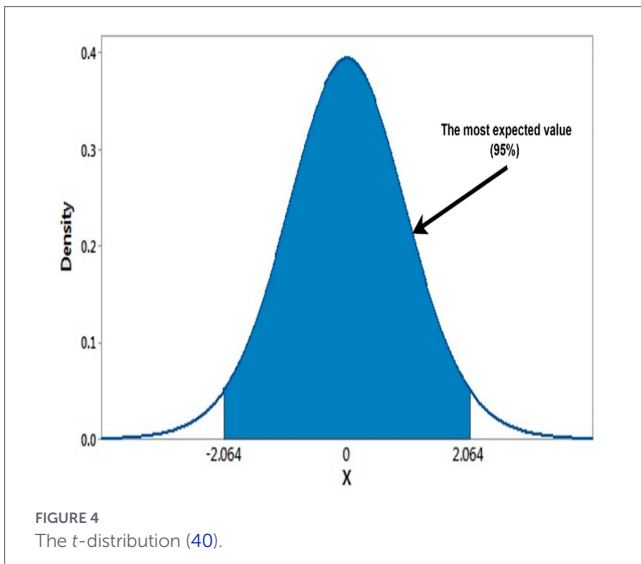


FIGURE 4 The t-distribution (40).

with a total area of unity and is commonly referred to as the sampling function. The Sinc function is extensively utilized in signal processing and plays a significant role in the theory of Fourier Transforms (19).

In mathematics, physics, and engineering, the Sinc function, represented as  $\text{Sinc}(x)$ . It exists in two forms: normalized and unnormalized. The historical unnormalized Sinc function in mathematics is defined for  $x \neq 0$  as follows:

$$\text{Sinc } x = \sin x / x \tag{8}$$

In digital signal processing and information theory, the normalized Sinc function is typically defined for  $x \neq 0$  in Equation 9:

$$\text{Sinc } x = \frac{\sin(\pi x)}{\pi x} \tag{9}$$

In both cases, the value at  $x = 0$  is determined as the limit of the function:

$$\text{Sinc } 0 := \lim_{x \rightarrow 0} \frac{\sin(ax)}{ax} = 1 \tag{10}$$

For any real  $a \neq 0$ .

## 2.3 Fault detection model

### 2.3.1 Datasets

Data is the core of ML and DL for developing effective fault detection algorithms. As natural bearing degradation is a cumulative process with time, artificially introduced faults or life tests with accelerations are commonly used to create datasets that closely mimic actual failure scenarios. While still a time-consuming activity, a number of organizations have publicly released bearing fault datasets that have become a rich source for algorithm development, widely accepted benchmarks, and a basis for comparing different fault diagnosis methods. The following three datasets have been chosen to feed and evaluate the proposed model.

Regarding the first dataset, The Case Western Reserve University (CWRU) bearing dataset is widely used as a benchmark dataset for

fault diagnosis. The CWRU provides vibration data for both healthy and damaged bearings using a 2 hp motor (see Figure 5). Defect sizes range from 0.007 to 0.040 inches when defects are introduced separately on the inner raceway, rolling elements (balls), and outer raceway, which are artificially induced using Electro-Discharge Machining (EDM). The dataset contains vibration data collected across different configurations, including 0–3 hp and 1720–1797 rpm. The Data was gathered from two accelerometers mounted at the drive end and the fan end of the motor housing with sampling rates of 12 kHz and 48 kHz [57].

The second dataset, The Huazhong University of Science and Technology (HUST) bearing dataset is newer compared to CWRU and Paderborn for the fault diagnosis of ball bearings, which was designed to address limitations of existing datasets by incorporating diverse bearing types and defect states. The HUST consists of 99 raw vibration signals for six fault types (inner crack, outer crack, ball crack, and their combinations) on five bearing types (6,204, 6,205, 6,206, 6,207, and 6,208) with three operating loads (0 W, 200 W, and 400 W). Wire-cutting methods artificially induced the faults, creating 0.2 mm micro-cracks to mimic initial stage defects, with the signals recorded at a sampling frequency of 51.2 kHz for 10 s on a 750 W induction motor test bench. High-resolution vibration signals are collected using torque transducers and accelerometers, which provide accurate measurements. HUST is richer compared to traditional datasets and is, therefore, applicable for assessing performance of machine learning models under normal and faulty operating states [45, 59]. HUST dataset test-bench is shown in Figure 6.

### 2.3.2 Mel-frequency cepstral coefficients (MFCC)

The MFCC are widely used features for representing the spectral characteristics of vibration or acoustic signals in fault detection tasks. The process begins by applying the Short-Time Fourier Transform (STFT) to a framed signal  $x(t)$ :

$$X(k) = \sum_{n=0}^{N-1} x(n)w(n)e^{-2j\pi kn/N} \tag{11}$$

The magnitude spectrum ( $|X(k)|$ ) is then filtered through a Mel filter bank, where the Mel scale is defined Equation 12:

$$f_{\text{mel}} = 2595 \log_{10} \left( 1 + \frac{f}{700} \right) \tag{12}$$

Finally, the Discrete Cosine Transform (DCT) is applied to the logarithmic Mel-filtered energies to obtain the MFCCs:

$$C_m = \sum_{k=1}^K \log(E_k) \cos \left[ m \left( k - \frac{1}{2} \right) \frac{\pi}{K} \right] \tag{13}$$

These coefficients effectively capture the nonlinear spectral features of the signal, making them robust descriptors for fault detection and condition monitoring.

The Mel-Frequency Cepstral Coefficients (MFCC) have demonstrated strong representational capabilities not only in speech

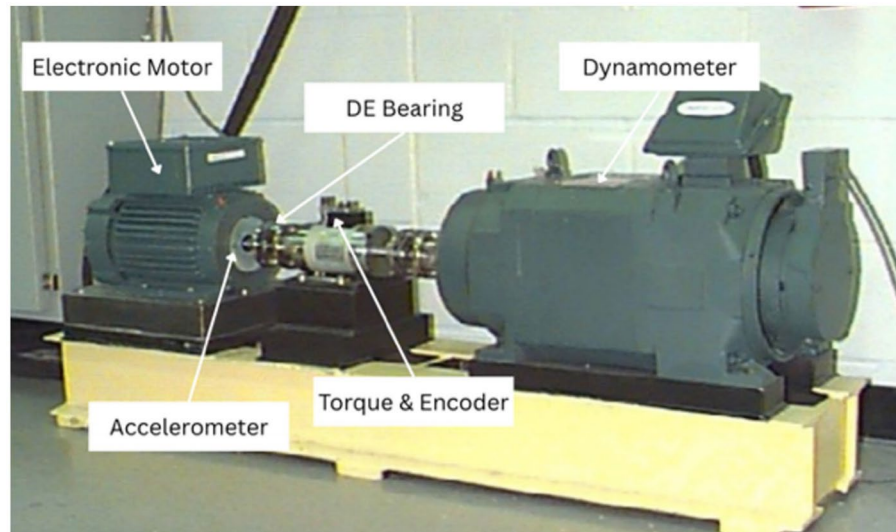


FIGURE 5  
CWRU dataset test-bench (2).

analysis but also in fault detection of rotating machinery (20). To enhance their performance in mechanical fault diagnosis, several hyperparameters of the MFCC can be fine-tuned to better capture the relevant signal characteristics. In this study, four key hyperparameters are considered and optimized to improve feature quality and classification accuracy. These hyperparameters are summarized in Table 1.

Hyperparameter	Effect
Number of MFCC coefficients ( $n_{mfcc}$ )	Controls level of spectral detail captured
FFT window size ( $n_{fft}$ )	Controls frequency resolution tradeoff
Step size between FFT frames ( $hop\_length$ )	Controls temporal resolution and feature size
Number of time frames kept ( $n_{time\_step}$ )	Controls temporal coverage and feature length

### 2.3.3 Echo State Network (ESN)

The Echo State Network (ESN) is an efficient recurrent neural network model within the reservoir computing framework. Its architecture consists of an input layer, a randomly initialized dynamic reservoir, and a trainable output layer. Unlike conventional RNNs or LSTMs, the recurrent reservoir connections remain fixed after initialization, and only the output weights are trained. This design greatly reduces computational cost, mitigates gradient-related problems, and enables fast learning. The reservoir acts as a nonlinear dynamical system, projecting inputs into a high-dimensional space where temporal dependencies are effectively captured. Owing to its simplicity and efficiency, ESNs are widely applied in time-series prediction, control, and signal processing. Figure 7 illustrates the ESN architecture.

## 3 Methodology

### 3.1 Phase one: the proposed EFOX algorithm

This section provides a comprehensive explanation of the proposed EFOX method developed to address continuous optimization challenges.

#### 3.1.1 Motivations

The FOX optimization algorithm was selected for enhancement because of its simple and straightforward structure. Despite its effectiveness, there remains room to further improve the balance between exploration and exploitation, which motivates enhancing the algorithm's overall search performance (21). Previous work (22) has shown that better balancing exploration and exploitation in FOX can speed up convergence and reduce the likelihood of getting stuck in local optima. Building on these findings, this study introduces an enhanced FOX algorithm (EFOX) that integrates randomization and selection strategies to better handle continuous optimization problems. This enhancement is especially suitable for MFCC hyperparameter tuning in bearing fault detection, which is inherently nonlinear and multimodal. In such problems, strong exploration helps avoid premature convergence, while improved exploitation supports stable convergence and more discriminative fault-sensitive feature representations. The complete flow chart of the proposed EFOX is given in Figure 8.

#### 3.1.2 The EFOX algorithm

Like most metaheuristic algorithms, the proposed approach begins by randomly generating an initial population. This population, represented by matrix  $X$ , corresponds to the positions of all foxes in the search space. During each iteration, the fitness of every search agent is assessed using standard benchmark functions. To identify the BestFitness, as well as the first and second best positions, each agent's fitness value (each row in  $X$ ) is compared with those of the others. This

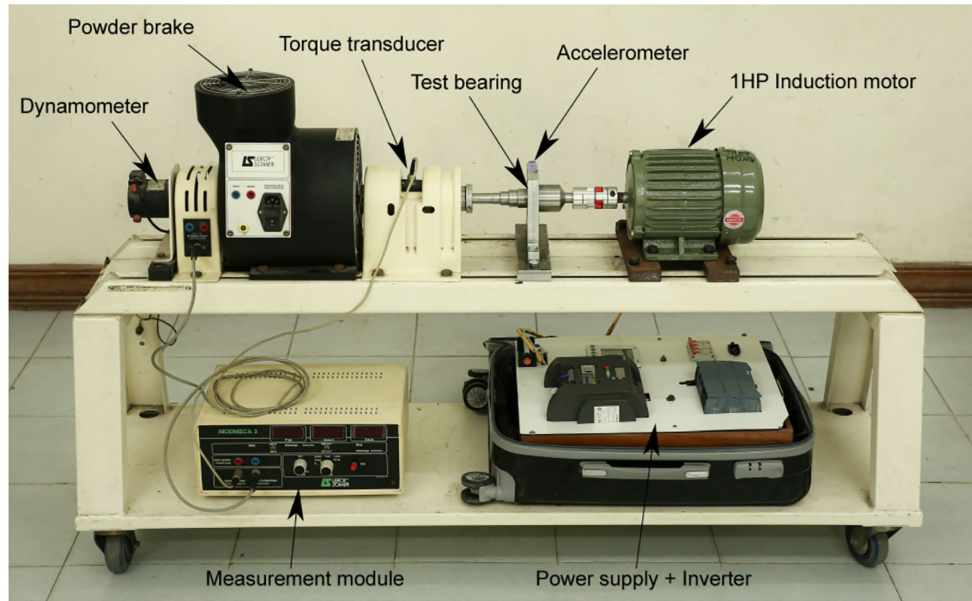


FIGURE 6 HUST dataset test-bench (2).

TABLE 1 Comparing EFOX against FOX, GWO, CHOA, and GOOES using classical benchmark functions.

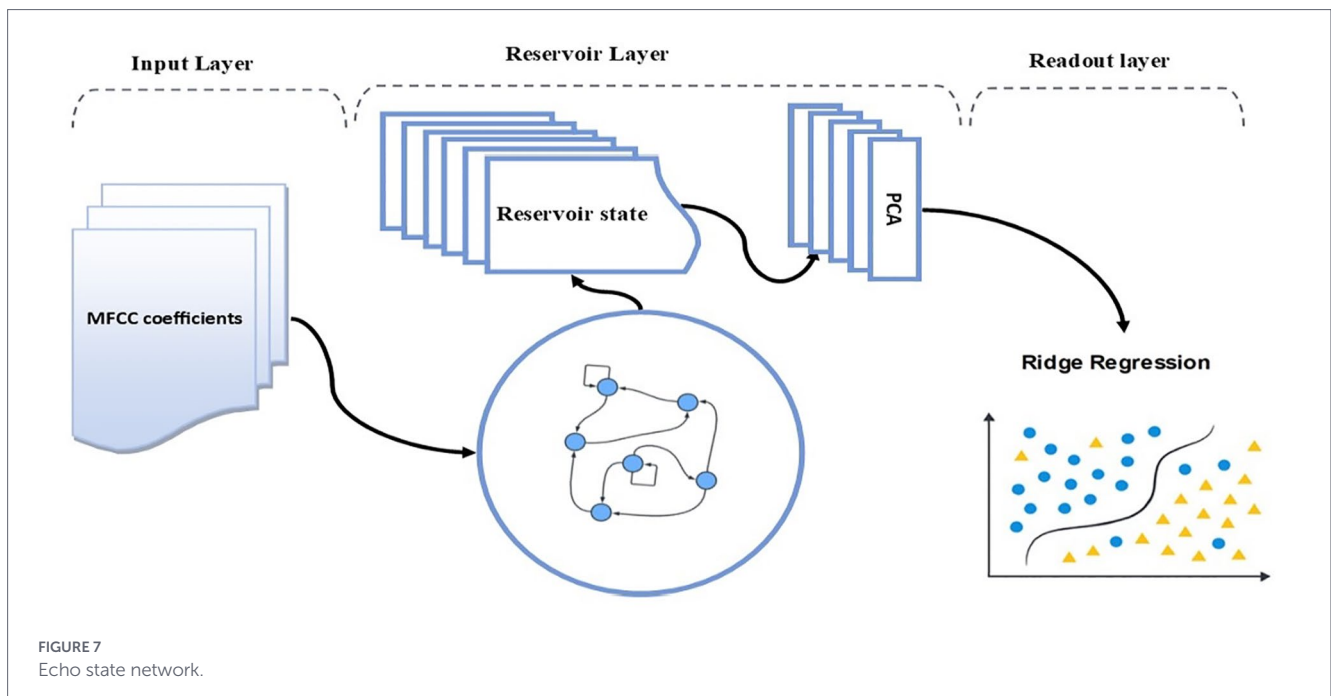
F	EFOX		FOX		GWO		CHOA		GOOSE	
	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std
TF1	0	0	0	0	1.96E-27	3.2915E-27	6.08E-06	1.689E-05	5.47E-04	0.00026896
TF2	0	0	0	0	8.15E-17	5.6905E-17	5.72E-05	0.00015554	5.77E-02	0.01805323
TF3	0	0	0	0	7.49E-06	1.3169E-05	1.20E+02	245.547689	4.80E-03	0.00356042
TF4	0	0	0	0	8.98E-07	9.6014E-07	3.15E-01	0.32473928	1.55E-02	0.00570136
TF5	27.6565552	0.34259044	0.34259044	0.0587183	2.73E+01	0.84251058	2.89E+01	0.20917015	1.87E+02	446.509789
TF6	0.00247268	0.000788316	0.000788316	0.00418182	7.97E-01	0.40198055	3.61E+00	0.47871959	5.04E-04	0.00027793
TF7	7.6244E-05	8.96888E-05	8.96888E-05	0.00014082	2.11E-03	0.00117152	2.48E-03	0.0025304	1.64E-02	0.00777164
TF8	-5547.10391	130.224082	130.224082	545.497008	-5.78E+03	949.535053	-5.74E+03	68.3310988	-2.64E+03	325.229344
TF9	0	0	0	0	1.43E+00	2.37464532	1.31E+01	11.0833708	2.07E+01	11.6011569
TF10	8.88E-16	1.00E-31	1.00293E-31	1.0029E-31	1.10E-13	1.878E-14	2.00E+01	0.00109651	8.48E+00	9.10052745
TF11	0	0	0	0	1.80E-03	0.00493553	2.43E-02	0.03185958	3.21E+01	10.6928125
TF12	0.00015957	6.82022E-05	6.82022E-05	7.8436E-05	4.22E-02	0.02310749	5.35E-01	0.19924978	5.60E+00	3.46322555
TF13	0.01004866	0.009041319	0.009041319	1.42789638	5.77E-01	0.19723146	2.77E+00	0.10815446	6.57E-04	0.00210973
TF14	9.309628	4.813753882	4.813753882	3.88725079	4.40E+00	3.99047805	1.13E+00	0.50022271	1.38E+01	6.7570208
TF15	0.00032938	5.38329E-05	5.38329E-05	0.00017436	5.08E-03	0.00857727	1.32E-03	7.4925E-05	6.09E-03	0.00875736
TF16	-1.03162845	2.44483E-09	2.44483E-09	0.24903515	-1.03E+00	3.3019E-08	-1.03E+00	1.5628E-05	-1.03E+00	3.222E-06
TF17	0.39788736	3.00374E-09	3.00374E-09	9.3772E-10	3.98E-01	0.0001185	3.99E-01	0.0016997	3.98E-01	2.0261E-06
TF18	3.00000008	1.34164E-07	1.34164E-07	25.0521527	3.00E+00	4.4453E-05	3.00E+00	0.00021215	3.00E+00	0.00030501
TF19	-3.8627758	1.18608E-05	1.18608E-05	1.7125E-06	-3.86E+00	0.00221601	-3.86E+00	0.00258606	-3.86E+00	4.7814E-05

comparison is carried out through a condition that evaluates whether the fitness of the current agent (fitness<sub>{i + 1}</sub>) is better than that of the previous one (fitness<sub>i</sub>) across iterations.

After 1st iteration, we have the Elitism position, which refers to a mechanism that ensures the preservation of the best agents from the

current generation into the next generation. This guarantees that high-quality solutions are not lost during the evolutionary process due to random operations (23–26).

To maintain a proper balance between the exploration and exploitation stages, the algorithm introduces a random factor that helps



distribute both processes evenly throughout the iterations. This random variable, denoted as  $r$ , gives each phase an equal 50% chance of being selected during optimization. When  $r$  is greater than or equal to 0.5, the algorithm switches to the exploitation phase.

In the exploitation phase, a new condition is established based on the probability of successfully hunting the prey. If  $p$  exceeds 0.5, the fox must determine a new position and it will jump towards the north-east direction. The random variable  $p$  ranges between [0, 1].

To determine a new position, the sound distance travels  $SDT_{it}$ , the Fox-Prey Relative Distance  $FPD_{it}$  and jumping value  $JV_{it}$  has to be computed Equation 14.

$$SDT_{it} = SOS * STT_{it} \tag{14}$$

Where,  $STT_{it}$  is the sound travel time and a random number ranging from 0 to 1.

SOS is the speed of sound in the medium SOS is equal to 343 in the air.

It is the number of iterations, ranging from 1 to 500.

However, another equation is introduced to determine the SOS, which relies on the best and second-best positions identified so far. This calculation involves dividing the sound travel time ( $STT_{it}$ ) between the fox and its prey to guide the search process more effectively. The SOS is calculated as follows:

$$SOS = (1^{st} \text{ best position} + 2^{nd} \text{ best position}) / STT_{it} \tag{15}$$

The Equation 13 is used to calculate the Fox-Prey Relative Distance  $FPD_{it}$  using the Sinc function, which is applied to a scaled value of the sound distance traveled,  $SDT_{it}$  divided by  $\frac{1}{2}$ . The output of the multiplication between  $(SDT_{it} * 0.5)$  is substituted in the Sinc function. The sinc function produces an output based on the value of  $(SDT_{it} * 0.5)$ . The Sinc function's role is to adjust the distance between the fox and the prey, ensuring that the shortest sound travel distance is normalized to one, while the longest sound travel distance is

reduced to a value significantly less than one, as shown in the Equations 8, 10.

$$FPD_{it} = \text{Sinc}(SDT_{it} * 0.5) \tag{16}$$

The jumping value  $JV_{it}$  need to determine the height of jump the fox to catch the prey in Equation 17.

$$JV_{it} = 0.5 * g * \text{AvgT}^2 \tag{17}$$

Where  $g$  is a gravitation acceleration ( $g = 9.81 \text{ m/s}^2$ ).

$\text{AvgT}^2$  is the average time it takes for sound to travel, squared to account for the upward and downward motion during the jump. The time transition  $\text{TranT}$  value is calculated in Equation 18 by dividing the sum of the  $STT_{it}$  to the number of problems dimensions.

$$\text{TranT} = \sum STT_{it} / \text{dim} \tag{18}$$

Then, the average time  $\text{AvgT}$  is found in Equation 19 by Levy distribution of the time transition  $\text{TranT}$  divided by 2. Levy distributions are adopted in Equation 4. It is often used for improving exploratory behavior because it allows large jumps that help escape local optima.

$$\text{AvgT} = (\text{levy} * \text{TranT}) / 2 \tag{19}$$

Now, the new position  $P_{\text{new}}$  of the fox is calculated in Equation 20 by multiplying the Fox-Prey Relative Distance  $FPD_{it}$ , the jumping value  $JV_{it}$  and the random number  $c_1$  that range of  $c_1$  is [0, 0.18].

$$P_{\text{new}} = FPD_{it} * JV_{it} * c_1 \tag{20}$$

If  $P$  exceeds 0.5, it is false, and another condition is detected. If  $P > 0.4$  and  $P = 0.5$ , the fox must determine another new position.

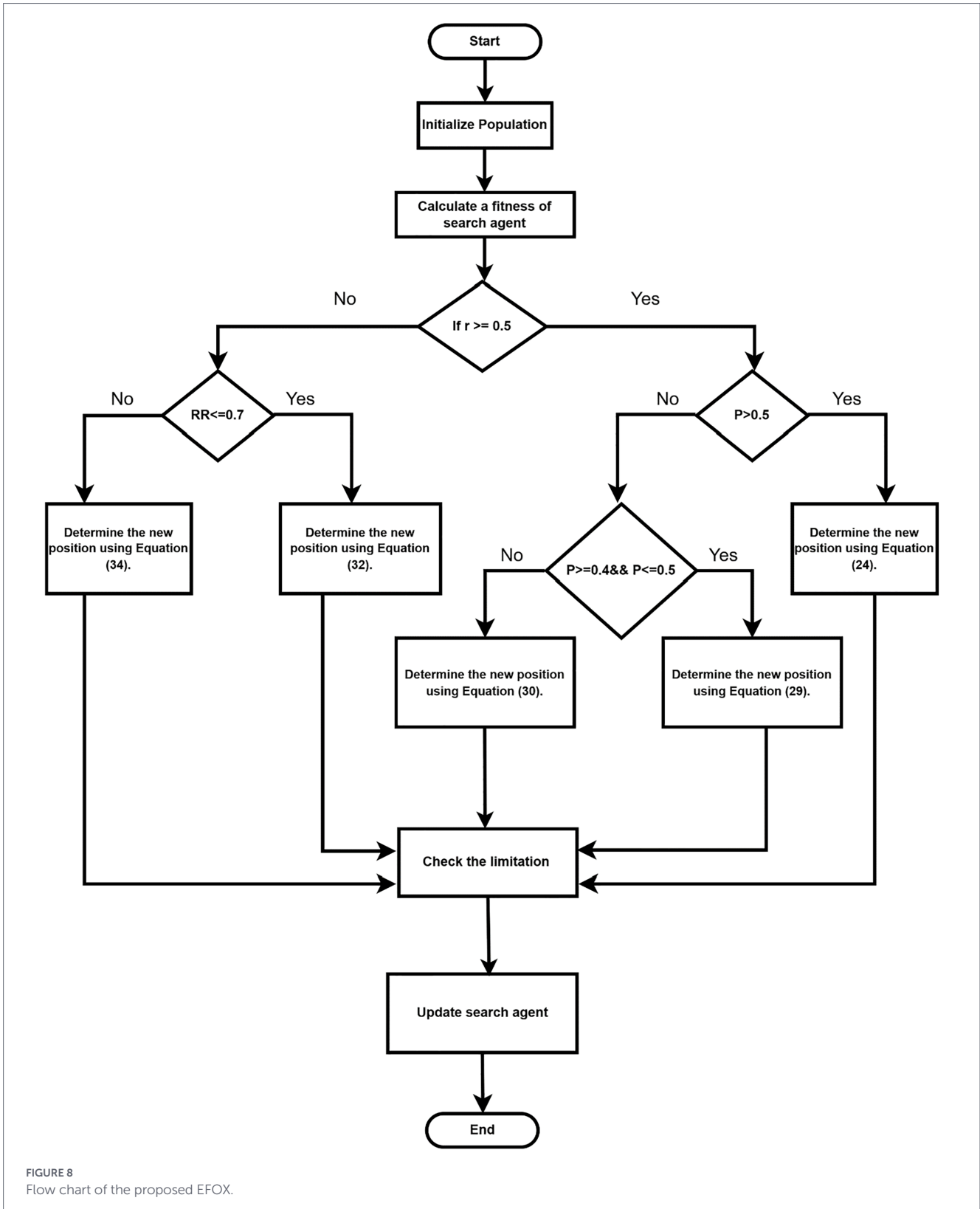


FIGURE 8  
Flow chart of the proposed EFOX.

To find SOS by Equation 21, which is based on the 2nd best position and Elitism position that was selected the previous iteration. Keeping the best position of the previous iteration is also important as it ensures that high-quality solutions are preserved throughout the evolutionary process, preventing them from being lost due to random operations.

$$SOS = (2^{nd} \text{ best position} + ELI) / STT_{it} \quad (21)$$

The sound distance travels  $SDT_{it}$  also calculate by Equation 14, and the time transition  $TranT$  calculated by Equation 18. The Fox-Prey

Relative Distance  $FPD_{it}$ , in this stage is found in Equation 22 by the sound distance travels  $SDT_{it}$  multiplied by 0.5 or  $\frac{1}{2}$ .

$$FPD_{it} = SDT_{it} * 0.5 \tag{22}$$

To adjust the values of  $AvgT$  and  $JV_{it}$  to exploit most of the areas that are  $\pm 2$  far from the fox, the  $t$  distribution is utilized. Hence, the average time  $AvgT$  and the jumping value  $JV_{it}$  are multiplied by the  $t$  distribution. Both of the processes are formulated in Equations 23, 24.

$$AvgT = (TranT \div 2) * tdis \tag{23}$$

$$JV_{it} = 0.5 * g * AvgT^2 * tdis \tag{24}$$

So, the new position  $P_{new}$  of the fox is calculated in Equation 25 by multiplying the Fox-Prey Relative Distance  $FPD_{it}$ , the jumping value  $JV_{it}$  and the random number  $c_2$  that range of  $c_1$  is [0.19, 1].

$$P_{new} = FPD_{it} * JV_{it} * c_2 \tag{25}$$

When the condition of ( $P > = 0.4$  &&  $P < = 0.5$ ) is false the equation of the new position changes and Using  $rand(1, dimension)$  to ensure that the fox moves stochastically to explore the prey.

$$P_{new} = 1^{st} \text{ best position} + randn(1, dim) * (a * ELI) \tag{26}$$

Where  $a$  is a variable to control the search, the minimum time variable and the variable  $a$  are utilized to control and regulate the search process.

Equation 24 shows the calculation of the variable  $a$ .

$$a = 0.5 * \left( 1 - \left( \frac{1}{\max_{it}} \right) \right) \tag{27}$$

Where,  $\max_{it}$  is the maximum iterations.

In the exploration phase, the fox randomly walks to the search area, guided by the best position discovered. Unlike other phases, the fox does not use a jumping technique here, as the focus is on random movement to locate prey. To ensure the fox moves randomly yet progressively toward the best position. If ( $r > = 0.5$ ) is false, the exploration phase is to be activated in this stage a condition is established, and the new random variable  $r$  ranges between [0, 1]. Also, the  $rr$  is the random number in the exploration phase. If  $rr$  is smaller and equal to 0.7, the equation to find the new position as follows Equation 28 defines the new position:

$$P_{new} = 1^{st} \text{ best position} + randn(1, dim) * (a * MinT) \tag{28}$$

Where  $MinT$  is the minimum time variable, the Equation 29 is used to find  $MinT$ . The  $MinT$  is scaled using the golden ratio (1.618) which helps maintain proportions that converge efficiently.

$$MinT = TranT * 1.618 \tag{29}$$

Otherwise, if the  $rr$  is greater than 0.7, the equation of the new position is modified as formulated in Equation 30.

$$P_{new} = 1^{st} \text{ best position} + randn(1, dim) * (a * 2^{nd} \text{ best position}) \tag{30}$$

The EFOX algorithm is an enhanced version of the FOX algorithm, incorporating features such as elitism, second-best position updates, and dynamic adjustments. Each of these features plays a crucial role in improving the performance of the FOX algorithm. For instance, the elitism feature ensures that the best solutions are retained and not lost in subsequent iterations. The second-best position is incorporated to refine and enhance the exploitation phase. Random variables  $r$ ,  $p$ , and  $a$  are used to balance the exploration and exploitation phases effectively. Additionally, randomization like Levy distribution and  $t$ -distribution, which are used in proposed model, help the metaheuristic strike a better balance between exploring new areas and refining good solutions. It allows the algorithm to make small, careful adjustments as well as occasional larger jumps, which helps avoid getting stuck too early and leads to better overall optimization performance. Figure 9 demonstrates the hunting.

### 3.1.3 Procedure of EFOX

Following are the detailed steps of how the proposed algorithm works:

Step 1 Initialization: the EFOX algorithm begins by initializing the search agents' positions ( $X$ ) randomly within the specified lower and upper bounds. It sets initial values for critical variables such as the 1st best position, best score, 2nd best position and the Elitism solution. Additional parameters like  $c1$ ,  $c2$ , and  $a$  are also defined to guide the movement and balance exploration and exploitation phases during the optimization process.

Step 2 Fitness Evaluation: the algorithm evaluates the fitness of each search agent using the objective function. The fitness of each agent is compared to identify and update the best score and its corresponding position. The fitness values of all agents are stored and sorted to determine the secondary best solution ensuring that the two top-performing agents guide the search process.

Step 3 Exploration and Exploitation Control: the two phases are controlled dynamically using the variable  $a$ , which decreases over iterations to shift focus from exploration to exploitation as the algorithm progresses. Random variables  $r$  and  $p$  are used to determine the movement type. Based on these variables, the algorithm selects different update strategies, positions may be updated using the 1st best position, 2nd best position and the Elitism solution with distance and movement calculations guided by the sinc function and Lévy flights.

Step 4 Jumping Mechanism: The jumping mechanism enhances the algorithm's exploration capabilities by calculating jump distances using the Lévy distribution and  $t$  distribution. This allows the search agents to make both small and large steps, enabling efficient exploration of the search space while avoiding local optima. The movement is carefully designed to balance exploration of new areas and refinement near promising solutions.

Step 5 Elitism and 2nd Updates: The algorithm incorporates elitism by retaining the best solution and updates it at the end of each iteration to ensure no high-quality solutions are lost. Additionally, 2nd

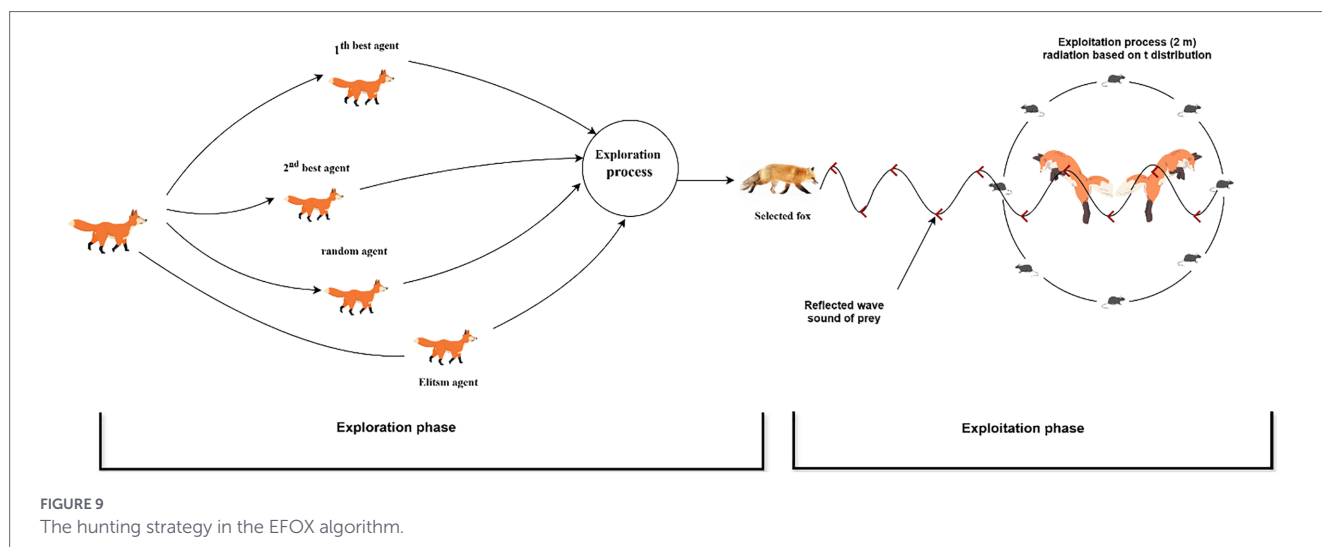


FIGURE 9  
The hunting strategy in the EFOX algorithm.

best position plays a significant role in guiding the search, especially during exploitation phases. The minimum time variable (MinT) is dynamically adjusted to refine movement, further improving the algorithm's precision.

**Step 6 Boundary Handling:** Boundary handling ensures that the search agents remain within the specified bounds of the search space. If any agent's position exceeds lower and upper bounds, it is reset to lie within the allowed range. This prevents the search process from deviating into invalid regions of the solution space.

**Step 7 Termination criterion:** The process repeats until the maximum number of iterations is reached. Once the termination condition is met, the algorithm returns the best score, the corresponding best position, and the updated elitism position as the final output. The mentioned approach ensures a balance between exploration and exploitation, making the EFOX algorithm highly effective for complex optimization problems.

### 3.2 Phase two: adopting a metaheuristic algorithm

Within the proposed framework, MFCC features are extracted from the raw vibration signals of two benchmark datasets, namely the CWRU dataset with a sampling frequency of 12 kHz and the HUST dataset with a sampling frequency of 51.2 kHz., where the hyperparameters of MFCC ( $n\_mfcc$ ,  $n\_fft$ ,  $hop\_length$ , and  $n\_time\_steps$ ) are initially randomized during the first iteration of the FOX algorithm. The extracted MFCC features are then fed into the Echo State Network (ESN), which performs fault classification based on the given feature representation. After each iteration, the error rate from the ESN is evaluated. If the performance does not meet the desired criterion, the EFOX optimizer updates the hyperparameters, and a new iteration begins. This process continues until the error rate reaches zero or the maximum number of iterations is achieved. Through this iterative optimization, the optimal MFCC hyperparameters are obtained, ensuring improved feature quality and classification accuracy. In the initial iteration, the EFOX algorithm initializes the hyperparameters. The corresponding error rate is then obtained from the machine learning model. Based on this feedback, EFOX applies its optimization strategy to minimize the error rate. This iterative process continues until either the maximum number of 500 iterations is reached, or a zero-error rate is achieved. Regarding the hyperparameters of the

optimization algorithm, such as the population size (30 agents) and the number of iterations (500), they were selected based on commonly used values in well-established previous studies. This choice helps ensure fair comparison with existing methods while keeping the computational cost reasonable the overall workflow of the proposed framework is illustrated in Figure 10.

## 4 Results and discussion

### 4.1 MFOX evaluation

To ensure the proposed algorithm performs effectively, it must undergo thorough testing through various methods. First, the algorithm should be evaluated against a diverse set of standard benchmark functions to assess its performance under varied conditions. Next, it should be applied to practical optimization problems relevant to its intended application, validating its effectiveness in real-world scenarios. Additionally, the results of the proposed EFOX algorithm should be compared with those of other algorithms. Statistical comparisons are essential to determine the significance of these results, ensuring that observed differences are meaningful and not due to random chance. This approach increases the reliability and validity of the conclusions. This section presents the performance evaluation of the proposed EFOX algorithm using these methods.

#### 4.1.1 Classical benchmark functions

A collection of well-known benchmark functions is used to assess the performance of the EFOX algorithm. These functions are divided into three categories unimodal, multimodal, and composite each possessing unique characteristics that test different aspects of the algorithm's capability (27).

To begin with, unimodal test functions are employed to evaluate how effectively the algorithm converges and exploits the search space. Since these functions contain only one optimal point, they serve as a reliable measure of the algorithm's efficiency in reaching the best possible solution (28). Secondly, multimodal test functions, as the name implies, contain several optima one global optimum and multiple

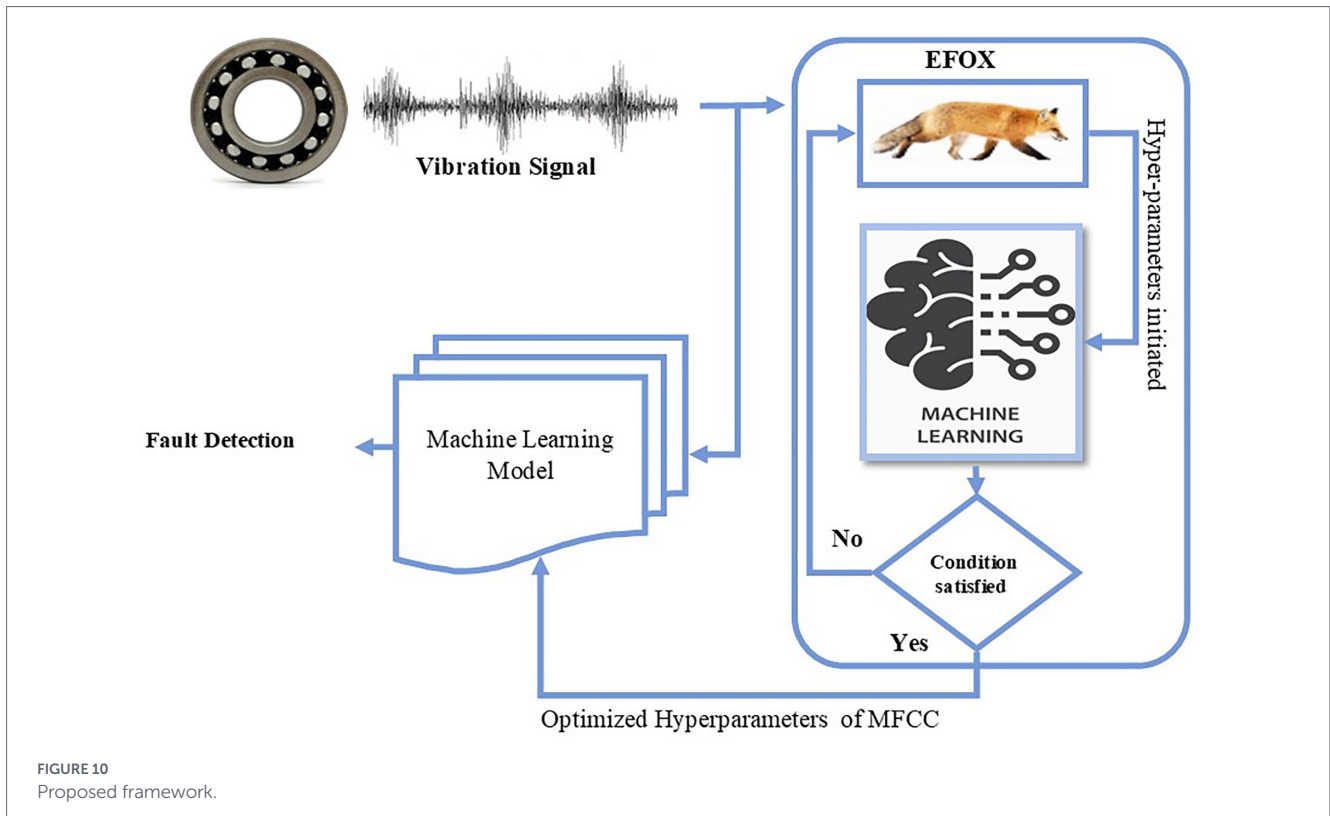


FIGURE 10 Proposed framework.

TABLE 2 Performance of ESN based on optimized MFCC.

Methods	CWRU			HUST		
	Original	-3 dB noise	-6 dB noise	Original	-3 dB noise	-6 dB noise
Optimized MFCC by FOX	100	96.5	95.1	100	95.3	94.2
Optimized MFCC by EFOX	100	100	99.9	100	99.8	99.59

local optima. To successfully reach the global optimum, the algorithm must skillfully explore the search space and avoid getting stuck in local solutions. This category is especially valuable for evaluating the algorithm’s exploration strength and its effectiveness in escaping local optima (29).

Finally, composite test functions represent intricate combinations of the previous categories, often incorporating biased, rotated, or shifted variations. These functions closely resemble the complexities of real-world optimization problems by introducing many local optima and diverse landscape patterns across different regions. They are designed to assess how well an algorithm maintains a balance between exploration and exploitation in challenging environments. Using such a diverse collection of benchmark functions provides a thorough assessment of the algorithm’s performance under various conditions (30). The test functions are executed 30 times using 30 search agents over 500 iterations. The average (Avg.) and standard deviation (Std) of the results are then computed. The best results for each test function are highlighted in bold and listed in Table 2. The results of the Classical benchmark functions for FOX algorithm, Chimp Optimization

Algorithm (ChOA) and the GOOSE algorithm are taken from (21, 31–33) for the first four unimodal test functions (TF1–TF4), the EFOX algorithm outperforms all the participated algorithms. However, the FOX algorithm performs better in the (TF5). The GOOSE algorithm achieved a superior result in only function (TF6). Moreover, for (TF7) the EFOX is the top. This proves that the algorithm’s exploitation and convergence speed are better than those in the comparison. GWO, however, provided a better result in (TF8). Nevertheless, the EFOX algorithm outperformed the other algorithms in (TF9 and TF11). The FOX algorithm, however, provided a better result in (TF10) and the EFOX algorithm achieved the second position for TF10, which means it outperformed the GWO, CHOA, and GOOSE algorithms. Furthermore, the produced result by the EFOX was better in (TF12) compared to the participated algorithms. This indicates that the EFOX possesses enhanced exploration capabilities by allocating almost half of the iterations to the exploration phase and the remaining half to the exploitation phase. EFOX also outperformed the other algorithms in (TF16 and TF19). However, regarding the remainder composite test functions, the EFOX algorithm achieved the second

rank among the presented algorithms. Overall, the results show that the EFOX and FOX produced similar results in six test functions (TF1, TF2, TF3, TF4, TF9, TF11), whereas the EFOX outperformed the FOX in three test functions (TF7, TF16, TF19). Additionally, the proposed algorithm ranks the second best in ten test functions compared to the selected comparator algorithms. These results demonstrate the proposed algorithm's capability to evade local optima, thoroughly explore the search space, and maintain a balance between exploration and exploitation.

### 4.1.2 CEC2019 benchmark test functions

In many real-world applications, achieving high accuracy is often more important than computational speed. Practitioners typically adjust algorithm parameters and run the optimization several times to obtain the most reliable solution for their particular problem, even if it takes longer. This aspect of numerical optimization is assessed using the CEC-C06 benchmark functions, commonly referred to as "The 100-Digit Challenge." These test functions measure an algorithm's performance by examining its function values at specific points, or "horizontal slices," along the convergence curve. They are widely used in an annual optimization competition to assess algorithms' capabilities in solving large-scale optimization problems (34). The first three benchmark functions (CEC01–CEC03) differ in their dimensional settings, as shown in Table 3. Meanwhile, functions CEC04 to CEC10 are formulated as 10-dimensional minimization problems within the range of [−100, 100], incorporating both shifting and rotation to increase the complexity of the optimization task. All CEC functions are scalable, and their global optima have been unified at point 1 (35). The outcomes of the CEC-C06 2019 benchmark tests for the EFOX algorithm are presented in Table 3. In Table 4, the best-performing results for each test function are highlighted in bold. Each function was executed 30 times using a population of 30 search agents across 500 iterations, after which the average performance and standard deviation were calculated. The results of the CEC2019 benchmark functions for the FOX, GWO, ChOA, and the GOOSE algorithms are taken from (21, 31–33). Table 4 shows the proposed EFOX has the first rank in seven test functions (CEC01, CEC02, CEC03, CEC06, CEC07, CEC09 and CEC10) compared to the other algorithms. However, GWO showed its superiority in CEC05 and CEC08. The results of the CEC2019 benchmark functions demonstrated that EFOX outperforms other algorithms in exploration efficiency and effectively avoids local optima. The results of the CEC2019 benchmarks are evidence that the proposed algorithm has a great ability to optimize large-scale optimization problems.

### 4.1.3 Statistical analysis

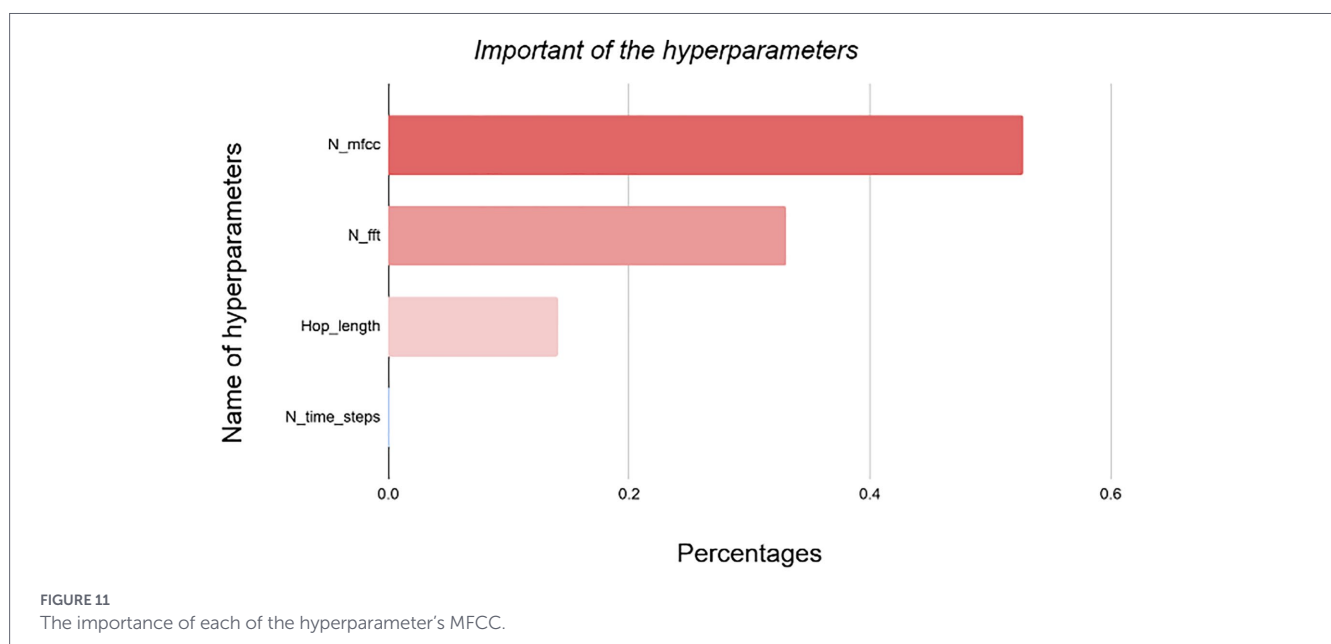
Statistical analysis is necessary to evaluate the algorithm's performance and determine its significance. To achieve this, the Wilcoxon rank-sum test is employed to calculate the *p*-value of the results (36). The *p* values reported in Table 2 for CEC2019 benchmark test functions prove that for most of the test functions the EFOX showed significantly better results compared to the FOX. As shown in Table 4, all results, except for CEC01 in FOX and CEC04 in GWO, were less than 0.05, highlighting the significance of the proposed algorithm's outcomes statistically.

TABLE 3 Comparing EFOX with FOX, GWO, CHOA, and GOOSE using CEC2019 benchmark functions.

F	EFOX			FOX			GWO			CHOA			GOOSE		
	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std	Avg.	Std	
CEC01	22038.71684	22531.70352	22561.57057	22996.30258	348,394,299	464,690,785	2,561,813,500	4,599,549,205	2.46077E+12	2.55486E+12					
CEC02	18.34377977	0.00037067	18.34348239	0.000272066	18.34394246	0.000288712	18.41507441	0.055832809	14731.61613	5174.261823					
CEC03	13.70240425	2.15335E−08	13.70240425	1.86367E−08	13.70240473	1.58553E−06	13.70242525	8.17822E−06	13.70253844	0.00052556					
CEC04	88.80753166	41.65098223	2019.562978	1735.507184	91.15020536	170.0443199	4835.769224	2476.48145	75.12198911	56.2698951					
CEC05	3.216313064	0.884599834	6.438179083	1.415452438	2.388861243	0.260732788	4.038578702	0.701567364	4.876941591	1.885557934					
CEC06	4.514850347	1.097928458	4.763926171	1.60385144	12.05454456	0.765329189	12.1729208	0.727051086	6.537763178	1.382478					
CEC07	326.9558009	220.471581	368.2207438	218.1869917	510.683356	299.1312799	975.4981999	187.0163171	383.458533	192.2233609					
CEC08	5.497164582	0.551931032	5.848268998	0.450511243	4.987291159	1.027349643	6.833879889	0.134251587	5.700032411	0.420108856					
CEC09	3.519874017	0.10124324	3.647504778	0.307348022	5.613164822	0.789622587	403.9208431	160.8329999	3.367272845	0.016567065					
CEC10	20.98610694	0.003488376	20.9927026	0.007872845	21.47163706	0.115197535	21.48486749	0.084910816	21.01230173	0.044238169					

TABLE 4 Statistical analysis using the Wilcoxon rank-sum test on CEC2019 benchmark functions.

F	EFOX vs. FOX	EFOX vs. GWO	EFOX vs. CHOA	EFOX vs. GOOSE
CEC01	0.929426683	0.000128	0.00344	2.05E-06
CEC02	0.000790107	0.062865	3E-09	2.2E-22
CEC03	0.448004205	0.10172	2.4E-20	0.167275
CEC04	9.61848E-08	0.941826	4.98E-15	0.288731
CEC05	3.78772E-15	7.65E-06	0.000188	5.26E-05
CEC06	0.485550454	1.11E-37	1.93E-38	4.76E-08
CEC07	0.469142258	0.008879	8.78E-18	0.294425
CEC08	0.009090404	0.019888	1.14E-18	0.114601
CEC09	0.034893209	8.22E-21	9.63E-20	3.47E-11
CEC10	9.45707E-05	6.54E-31	1.17E-38	0.002021



### 4.2 Bearing fault detection model

Both the enhanced FOX (EFOX) and the original FOX algorithms were utilized to fine-tune the MFCC hyperparameters, aiming to improve fault detection accuracy. Table 2 compares their performance on two benchmark bearing datasets CWRU and HUST under both clean and noisy (-3 dB and -6 dB) conditions. While both methods achieved perfect accuracy (100%) on the noise-free data, this does not fully demonstrate the impact of hyperparameter tuning, as bearing fault features are often clearly distinguishable in such ideal settings. To better assess robustness, controlled noise was added to make the classification task more challenging. Under these conditions, the EFOX-optimized MFCC outperformed the standard FOX approach, reaching 99.9 and 99.59% accuracy on the CWRU and HUST datasets, respectively, compared to 95.1 and 94.2% achieved by FOX. These results highlight the effectiveness of the EFOX algorithm in enhancing noise resistance and improving the stability of fault detection (see Table 2).

Comparing the two datasets (see Table 2), the proposed methods perform slightly better on the CWRU dataset than on the HUST

dataset, particularly under noisy conditions, showing that they are more robust and consistent when evaluated on CWRU data. In terms of hyperparameter importance, the Figure 11 illustrates how each MFCC parameter contributes to bearing fault detection performance. Among them, the number of MFCC coefficients (*n\_mfcc*) has the greatest impact accounting for more than half of the total significance which suggests that the dimensionality of the cepstral representation plays a key role in capturing meaningful spectral details related to fault conditions. The FFT window size (*n\_fft*) has a moderate effect, as it helps balance frequency resolution and temporal accuracy. Meanwhile, both the hop length (*hop\_length*) and *N\_time\_step* have relatively minor influence, with *N\_time\_step* contributing less than 0.1% to overall performance. These observations highlight the importance of precisely tuning *n\_mfcc* to strengthen the accuracy and reliability of bearing fault detection systems.

Regarding dataset CWRU, Table 5 presents a comparison of the proposed model with existing techniques in terms of classification accuracy and the number of fault labels considered. The PGCNN method (37) achieved an accuracy of 99.93% on 10 fault classes, while

TABLE 5 Accuracy comparison of fault-diagnosis techniques on CWRU.

Technique	Accuracy (%)	No. labels
PGCNN (37)	99.93	10
GCT-GNN (38)	98.83	10
Proposed model	99.90	16

TABLE 6 Classification results on HUST dataset.

Technique	Accuracy (%)
Multitask CNN-LSTM (39)	97.63
Proposed model	98.2

GCT-GNN (38) obtained 98.83% under the same conditions. The proposed model, although evaluated on a larger and more complex dataset containing 16 labels, achieved a comparable accuracy of 99.90%, demonstrating its strong generalization capability and robustness in bearing fault detection.

Regarding the HUST dataset, Table 6 compares the performance of the proposed model with the Multitask CNN-LSTM approach (39). While the Multitask CNN-LSTM achieved an accuracy of 97.63%, the proposed model attained a slightly higher accuracy of 98.2%, indicating improved feature representation and classification capability. This improvement highlights the effectiveness of the enhanced FOX-based MFCC optimization in enhancing fault detection performance.

## 5 Conclusion

This study introduced an optimized Mel-Frequency Cepstral Coefficients (MFCC) feature extraction framework enhanced with the Improved FOX (EFOX) optimization algorithm to boost bearing fault detection performance. Through EFOX-based tuning of MFCC hyperparameters, the extracted features became more discriminative, resulting in higher classification accuracy and greater robustness than traditional methods. Among the optimized parameters, the number of MFCC coefficients ( $n_{mfcc}$ ) had the strongest impact accounting for over 50% of overall importance—emphasizing the crucial role of cepstral dimensionality in representing fault-related spectral characteristics. The FFT window size ( $n_{fft}$ ) had a moderate effect, while  $hop\_length$  and  $N\_time\_step$  contributed less, with  $N\_time\_step$  influencing performance by less than 0.1%. Overall, these results demonstrate that fine-tuning MFCC parameters, particularly  $n_{mfcc}$ , using the EFOX algorithm can significantly improve the accuracy and reliability of Echo State Network-based bearing fault diagnosis systems.

## References

- Abdul ZK, Al Talabani AK. Highly accurate gear fault diagnosis based on support vector machine. *J Vib Eng Technol.* (2022);0123456789. doi: 10.1007/s42417-022-00768-6
- Masood A., Zrar O., Abdul K., Maghdid S. (2025) Bearing Fault Detection Using an Enhanced Multi-Reservoir Echo State Network.
- Madan MIAK. Bearing fault diagnosis in CNC machine using hybrid signal decomposition and gentle AdaBoost learning. *J Vib Eng Technol.* (2024) 12:1621–34. doi: 10.1007/s42417-023-00930-8
- Madan MIAK. CNC machine—bearing fault detection based on convolutional neural network using vibration and acoustic signal. *J Vib Eng Technol.* (2022) 10:1613–21. doi: 10.1007/s42417-022-00468-1
- Iqbal M, Naseem AKM. Vibration and acoustic signal-based bearing fault diagnosis in CNC machine using an improved deep learning. *Iran J Comput Sci.* (2024) 7:723–33. doi: 10.1007/s42044-024-00205-9
- Zhang S., Member S., Zhang S., Member S. (2020) Deep Learning Algorithms for Bearing Fault Diagnostics—A Comprehensive Review. Springer Nature / Singapore.

## Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found in the article/supplementary material.

## Author contributions

LA: Writing – original draft. CR: Writing – review & editing.

## Funding

The author(s) declared that financial support was not received for this work and/or its publication.

## Conflict of interest

The author(s) declared that this work was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## Generative AI statement

The author(s) declared that Generative AI was not used in the creation of this manuscript.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

## Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

7. Zhang X., Zhu J., Wu Y., Zhen D. (2020). Feature Extraction for Bearing Fault Detection Using Wavelet Packet Energy and Fast Kurtogram Analysis. MDPI / Basel, Switzerland.
8. Abdul AKA-T, Zrar KH. Mel frequency cepstral coefficient and its applications: a review. *Ieee Access*. (2022) 10:122136–58. doi: 10.1109/ACCESS.2022.3223444
9. Marseglia GR, Scott JK, Magni L, Braatz RD, Raimondo DM. A hybrid stochastic-deterministic approach for active fault diagnosis using scenario optimization. *IFAC Proc Vol*. (2014) 47:1102–7. doi: 10.3182/20140824-6-za-1003.02590
10. Abdul ZK, Al-Talabani AK, Gwad WH, Alkayal E, Maghdid HS, Asaad SM. Optimizing gammatone cepstral coefficients for gear fault detection. *IEEE Access*. (2025) 13:101711–20. doi: 10.1109/ACCESS.2025.3577667
11. Frafjord K. Summer food habits of Arctic foxes in the alpine region of southern Scandinavia, with a note on sympatric red foxes. *Ann Zool Fenn*. (1995) 32:111–6. doi: 10.5962/p.356168
12. Červený J, Begall S, Koubek P, Nováková P, Burda H. Directional preference may enhance hunting accuracy in foraging foxes. *Biol Lett*. (2011) 7:355–7. doi: 10.1098/rsbl.2010.1145
13. Choi KP, Kam EHH, Tong XT, Wong WK. Appropriate noise addition to metaheuristic algorithms can enhance their performance. *Sci Rep*. (2023) 13:1–12. doi: 10.1038/s41598-023-29618-5
14. Chawla M, Duhan M. Levy flights in metaheuristics optimization algorithms—a review. *Appl Artif Intell*. (2018) 32:802–21. doi: 10.1080/08839514.2018.1508807
15. Ibe OC. "Levy processes". In: Markov Process. Stochastic Modeling (2013). p. 329–47.
16. Peel D, McLachlan GJ. Robust mixture modelling using the t distribution. *Stat Comput*. MDPI / Basel, Switzerland (2000) 10:339–48. doi: 10.1023/A:1008981510081
17. Grosan C., Oltean M., Oltean M., "Acta Universitatis Apulensis the role of elitism in multiobjective optimization," pp. 83–90 (2003).
18. DeJong K. A. Analysis of the Behavior of a Class of Genetic Adaptive Systems. University of Michigan / Michigan (1975)
19. Qi F, Taylor P. Series expansions for powers of Sinc function and closed-form expressions for specific partial bell polynomials. *Appl Anal Discrete Math*. (2024) 18:92–115. doi: 10.2298/AADM230902020Q
20. Abdul ZK, Al-Talabani AK, Ramadan DO. A hybrid temporal feature for gear fault diagnosis using the long short term memory. *IEEE Sensors J*. (2020) 20:14444–52. doi: 10.1109/jsen.2020.3007262
21. Mohammed H, Tarik R. "FOX: a FOX-inspired optimization algorithm: FOX: a fox-inspired optimization algorithm." Applied Intelligence Springer Nature / Singapore. (2023) 1030–50.
22. ALRahhal H, Jamous R. AFOX: a new adaptive nature-inspired optimization algorithm. *Artif Intell Rev*. (2023) 56:15523–66. doi: 10.1007/s10462-023-10542-z
23. Ab Wahab MN, Nefti-Meziani S, Atyabi A. A comprehensive review of swarm optimization algorithms. *PLoS One*. (2015) 10:1–36. doi: 10.1371/journal.pone.0122827
24. Kumar R. Effect of polygamy with selection in genetic algorithms. *International Journal of Soft Computing and Engineering (IJSCE)* (2012) 1:194–9. doi: 10.1007/978-3-642-27443-5\_5
25. Katoch S, Chauhan SS, Kumar V. A review on genetic algorithm: past, present, and future. *Multimed Tools Appl*. (2021) 80:8091–126. doi: 10.1007/s11042-020-10139-6
26. Bhargava S. A note on evolutionary algorithms and its applications. *Adults Learn Math Int J*. (2013) 8:31–45.
27. Yao X, Liu Y, Lin G. Evolutionary programming made faster. *IEEE Trans Evol Comput*. (1999) 3:82–102. doi: 10.1109/4235.771163
28. Marcin Molga C. S., Test functions for optimization algorithm. pp. 205–209, (2005) (no. c).
29. Okamoto T, Aiyoshi E. Global optimization using a synchronization of multiple search points autonomously driven by a chaotic dynamic model. *J Glob Optim*. (2008) 41:219–44. doi: 10.1007/s10898-007-9222-5
30. Chen Q, Liu B, Zhang Q, Liang J. J., Suganthan P. N., Qu B., Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical optimization. In: *2015 IEEE Congr. Evol. Comput*. (2015). pp. 84–88.
31. Liu W, Sun J, Liu G, Fu S, Liu M, Zhu Y, et al. Improved GWO and its application in parameter optimization of Elman neural network. *PLoS One*. (2023) 18:e0288071. doi: 10.1371/journal.pone.0288071
32. Khishe M, Mosavi MR. Chimp optimization algorithm. *Expert Syst Appl*. (2020) 149:113338. doi: 10.1016/j.eswa.2020.113338
33. Hamad RK, Rashid TA. Goose algorithm: a powerful optimization tool for real-world engineering challenges and beyond. *Evol Syst*. (2024) 15:1249–74. doi: 10.1007/s12530-023-09553-6
34. Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. *Adv Eng Softw*. (2014) 69:46–61. doi: 10.1016/j.advengsoft.2013.12.007
35. Rahman CM, Rashid TA. A new evolutionary algorithm: learner performance based behavior algorithm. *Egypt Inform J*. (2021) 22:213–23. doi: 10.1016/j.eij.2020.08.003
36. Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol Comput*. (2011) 1:3–18. doi: 10.1016/j.swevo.2011.02.002
37. Ruan D, Wang J, Yan J, Gühmann C. CNN parameter design based on fault signal analysis and its application in bearing fault diagnosis. *Adv Eng Inform*. (2023) 55:101877. doi: 10.1016/j.aei.2023.101877
38. Zhang Z, Wu L. Graph neural network-based bearing fault diagnosis using granger causality test. *Expert Syst Appl*. (2024) 242:122827. doi: 10.1016/j.eswa.2023.122827
39. Nguyen TH, Hung VV, Thinh DD, Tran TT, Hong HS. Generalized simulation-based domain adaptation approach for intelligent bearing fault diagnosis. *Arab J Sci Eng*. (2024) 49:16941–57. doi: 10.1007/s13369-024-09282-1
40. Hazra A, Gogtay N. Biostatistics series module 3: comparing groups: numerical variables. *Indian J Dermatol*. (2016) 61:251–60. doi: 10.4103/0019-5154.182416