Received February 12, 2020, accepted March 7, 2020, date of publication March 11, 2020, date of current version April 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2979999

Data Association-Based Fault Diagnosis of IMUs: Optimized DBN Design and Wheeled Robot Evaluation

LINLIN XIA, SHUFENG ZHANG, RAN SHEN, AND JIASHUO CUI
School of Automation Engineering, Northeast Electric Power University, Jilin 132012, China

Corresponding author: Linlin Xia (xiall521@neepu.edu.cn)

This work was supported in part by the Natural Research Fund of Science and Technology Department, Jilin Province, under Grant 20170101125JC, in part by the Science and Technology Program of Department of Education, Jilin Province, under Grant JJKH20200117KJ, and in part by the Research Fund for Distinguished Young Scholars of Jilin City under Grant 20190104128.

ABSTRACT Deep belief network (DBN) is now being recognized as a powerful and eminently practical tool for large scale data processing. The main characteristics of DBN are the feature extension from low-level content to high-level data association and the representation of joint distribution between original data and matched labels. For a wheeled robot with no other available location reference supports, the internally integrated inertial measurement units (IMUs) essentially requires the robot to be able to implement efficient fault diagnosis to locate and identify the faults, especially for the accumulated error caused by large drifts of gyroscopes. An optimized DBN based fault diagnosis design is proposed to deal with such faults with complexity and diversity. The highlights of the proposed DBN model lies in its combination of weight value optimization via an inexact LSA-GA (abbreviates 'inexact linear searching algorithm- genetic algorithm') and dynamic adjustment for hidden-layer neurons of constituent RBMs (abbreviates 'restricted Boltzmann machines'). The problems associated with DBN anatomy, bat algorithm (BA) description and fault diagnosis modeling are discussed in detail. The real robot platform experiments and dataset tests are conducted. The results indicate that, the optimized DBN design leads to a better fault classification with excellent generalization ability on given datasets, and the adjustable 'DBN structure' contributes to the data association extraction between multiples of fault categories. The proposed scheme may therefore be considered to provide preferred reference models for a class of data based fault diagnosis problems.

INDEX TERMS Optimized DBN, fault diagnosis, IMUs, weight value optimization, data association extraction.

I. INTRODUCTION

With the development and maturation of intelligent robot technology, robot and robotics appear to be not constrained to manufacturing domain, whereas, they show superior applicability to a wider range of areas involving resource exploration, disaster relief, medical services, military, aerospace, etc., [1]. For most GPS-denied working environments, if other external sensors (or references) are not available, the navigation information derived from IMUs (abbreviates 'inertial measurement units') is generally recognized as the main source from which the robot derives its location reference and further estimates its own location. Gyroscopes and accelerometers are the major components of IMUs. The functions of gyroscopes are to implement posture detection & navigation of robot base and motion control of robot manipulators. And the purposes of accelerometers are to provide information for moving acceleration detection & navigation of robot base as well as the feedback control of robot joints. Apparently, in cases where gyroscopes or accelerometers fail to properly function because of some certain faults, it will deteriorate the IMUs and consequently decrease the stability of robot due to the title of the base. In fact, the polluted output of IMUs (affected by external interference signals) and accumulated error of inertial sensors (more precisely, gyroscopes) are significant contributors that disable an autonomous robot to locate. The resulting faults therefore appear to be complex with varied types of risks, often accompanied by various

The associate editor coordinating the review of this manuscript and approving it for publication was Yangmin Li.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

malfunction behaviors afterwards. Note that, the complex and diverse faults of inertial sensors will not be valid for direct fault analyses, bringing great difficulties to desired fault diagnoses.

The expert system-based, filtering-based [2], [3] and χ^2 detection-based [4]–[6] algorithms form the conventional solutions to fault diagnoses of robot IMUs. They all show eminent performances in individual inertial sensor fault diagnosis tasks, but for purpose of combined sensor treatments (like fault diagnoses for integrated navigation systems), the above typical strategies that fail to exploit more deep knowledge between I-O data ultimately show their limitations. Lately, there has been more research in the area of data-driven based robot fault diagnosis. Meanwhile, the introduction of neural networks or deep neural networks makes it possible for the representation of associated nonlinear data via efficiently extracting the matched data features.

The principal component analysis (PCA), neural network, and support vector machine (SVM) represent the typical data-driven tools for sensor fault diagnoses. PCA methods [7]– [11] can be used to reduce data dimension and extract feature vectors. The PCA-fused algorithms therefore can provide solutions to feature extraction and fault identification for sensors. However, PCA itself is restricted to linear data processing, since a majority of IMU vibration signals to be corrected is essentially nonlinear, in cases where key data partially lost, the real time capacity and accuracy of PCA-based fault diagnoses would hardly be guaranteed. By contrast, SVM conforms to nonlinear, small sample data based classification tasks, being regarded as a statistically definable learning method [12]–[15]. References [16]–[19] perform SVM based robot sensor fault diagnoses by means of quadratic programming (for solving support vector). Due to the fact that the quadratic solution process involves the calculation of multi-order small matrix (the order of small matrix is equivalent to the number of samples), so that a large amount of small matrix storage (a large number of samples) implies much amount of machine memory and operation time have to be consumed. This is unacceptable for real-time industrial applications. Also, the network structure of SVM does not allow the training of large number of sample data. The answer to this challenging issue would be neural network [20]–[25].

The typical neural networks, like BP (abbreviates 'back prorogation'), RBF (abbreviates 'radial basis function'), adaptive probabilistic neural network, etc., enhance their own generalization abilities and classification performances by fusing some other computational means (including genetic algorithm, fuzzy logic control, SVM, etc.). They essentially reflect a low level I-O data association. Some research [26]–[30] focuses on exploring a deep association between fault data. Actually, the introduction of deep learning in sensor fault diagnosis tasks enlarges the applied ranges of the concerned fault diagnosis models. Reference [31] successfully realizes the DBN (abbreviates 'deep belief network') based fault diagnoses of navigation systems in high-speed rails. It is important to appreciate that, the binary detection data provided by high-speed rails is consistent with the data representation of input layer in DBN. In principle, the significant advantage of DBN over typical neural networks consists in two aspects. One is its applicability to complex nonlinear approximation problems with no concern of over-fitting and falling into local optimum situations. The other one is its applicability to deep level extraction of data association between data and fault categories, since the mapping is so essential.

Inspired by this, a dynamic DBN fault diagnosis model is proposed, which is expected to elevate the accuracy of the fault identification for inertial sensors of wheeled robots. Considering the engineering background, two training processes (Wake and Sleep) of DBN model have been stressed. The inexact linear genetic search idea (for weight optimization in Sleep stage) and bat algorithm (BA) fused DBN structure adjustment are both highlighted. As main contributions of this paper, the optimized DBN model is numerically simulated and further experimentally analyzed on the constructed fault datasets.

The outline of the remainder of the paper is as follows. In Section II, the typical DBN anatomy that consists of numbers of RBM models and the functions of wake stage and sleep stage in training process is given as the basis for further work. Section III is devoted to the optimized DBN design involving a weight value optimization strategy and a dynamic number adjustment treatment of RBM (abbreviates'restricted Boltzmann machine') hidden-layer neuron. The corresponding wheeled robot fault diagnosis modeling is stressed in Section IV, which emphasizes the collections of navigation parameters and the establishments of matched fault datasets. Section V carries out the experiments on the given datasets to fulfill the performance analysis of optimized DBN for fault diagnosis. Section VI presents the main conclusions of this investigation.

II. THE DBN ANATOMY

As discussed before, the complicated work volume of wheeled robot and accumulated error of interior inertial sensors may cause great IMU fault complexity. Also, the low correlative degree of raw data derived from IMUs, including attitude, velocity and position can hardly form the direct source for fault location or fault type determination. Taking the preprocessed IMU data as input of DBN training model, therefore, contributes to a deep association extraction of inertial information as well as an efficient mapping between data-type input and fault label-type output.

The diagrammatic representation of designed DBN anatomy is given in Fig.1. The DBN which is composed of 3 RBMs is actually a probability generation model. This model establishes a joint distribution between observation data and classification labels. The typical DBN adopts layer-by-layer training mode, i.e., the neuron outputs of current layers serve as the neuron inputs of next layers. Generally, the training process of DBN can be divided into two stages.

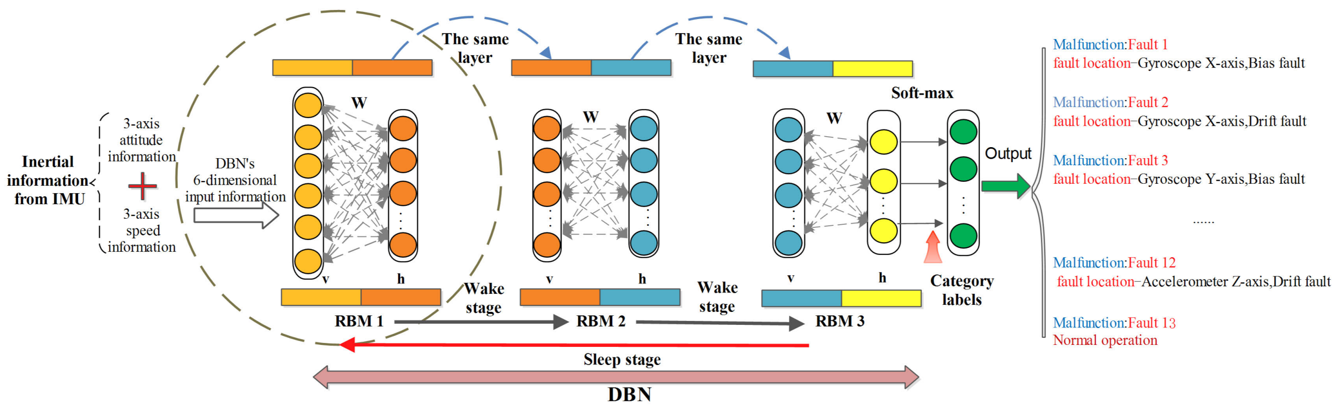

Inertial information from IMU
3-axis attitude information
3-axis speed information
DBN's 6-dimensional input information
The same layer
The same layer
W
v
h
Wake stage
RBM 1
RBM 2
RBM 3
Soft-max
Output
Category labels
Sleep stage
DBN
Malfunction:Fault 1
fault location–Gyroscope X-axis,Bias fault
Malfunction:Fault 2
fault location–Gyroscope X-axis,Drift fault
Malfunction:Fault 3
fault location–Gyroscope Y-axis,Bias fault
Malfunction:Fault 12
fault location–Accelerometer Z-axis,Drift fault
Malfunction:Fault 13
Normal operation

FIGURE 1. Structure diagram of DBN anatomy.

They are unsupervised Wake stage for pre-training and supervised Sleep stage for fine-tuning of weights involved.

A. THE RBM MODEL

3 RBMs (stacked in series) form the basic components of a DBN. For each RBM, it is structured with a hidden layer and a visible layer. As in Fig.1, the reciprocal dotted arrows indicate how information flows between visible - hidden layer of each RBM. Note that, the mechanism of information flow from hidden layer to visible layer is termed reconstruction. Given neuron states of visible layer, the activated neuron states of hidden layer are independent with each other. Analogously, given neuron states of hidden layer, the activated neuron states of visible layer are independent. Note also that, the variables describing neuron states in either hidden layer or visible layer appear to be binary, viz., the variables of both layers would conform to $\{0, 1\}$ [32].

Denote the neuron state of visible layer by v_i ($i = 1, 2, \ldots m$,m denotes neuron number of visible layer) and denote the neuron state of hidden layer by h_j($j = 1, 2, \ldots n$, n denotes neuron number of hidden layer). Considering the energy value $E(\cdot)$ of individual RBM model, we have

$$E(v, h|\theta) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} v_i W_{ij} h_j \tag{1}$$

where $\theta = \{W, a, b\}$ denotes the weight of RBM model, a_i and b_j respectively denote bias of visible layer and bias of hidden layer, W_{ij} denotes the weight value between the ith visible layer and the jth hidden layer.

Given $E(v, h|\theta)$, we further obtain the joint probability distribution between visible layer and hidden layer, viz.

$$P(v, h|\theta) = \frac{e^{-E(v,h|\theta)}}{Z(\theta)}, Z(\theta) = \sum_{v,h} e^{-E(v,h|\theta)} \tag{2}$$

where $Z(\theta)$ denotes the normalization factor.

We can solve for the activated neuron states of hidden layer (indicated by probability $P(\cdot)$) in terms of the known neuron states of visible layer, and vice versa. Then

$$P(h_j|v, \theta) = \sigma(b_j + \sum_{i=1}^{m} v_i W_{ij}) \tag{3}$$

$$P(v_i|h, \theta) = \sigma(a_i + \sum_{j=1}^{n} h_j W_{ij}) \tag{4}$$

where $\sigma = \frac{1}{1+exp(-1)}$ denotes the sigmoid-type activation function.

The updating of weight $\theta = \{W, a, b\}$ subjects to the following criterions

$$\begin{aligned}
\Delta W_{ij}^k &= \varepsilon(< v_i h_j >_o - < v_i h_j >_c),\\
W_{ij} &= W_{ij} + \Delta W_{ij}^k + \kappa \Delta W_{ij}^{k-1}\\
\Delta a_i^k &= \varepsilon(< v_i >_o - < v_i >_c),\\
a_i &= a_i + \Delta a_i^k + \kappa \Delta a_i^{k-1}\\
\Delta b_j^k &= \varepsilon(< h_j >_o - < h_j >_c),\\
b_j &= b_j + \Delta b_j^k + \kappa \Delta b_j^{k-1}
\end{aligned} \tag{5}$$

where ε denotes learning rate, $< \ . \ >_o$ (data-dependent term) denotes an expectation of sample data, $< \ . \ >_c$ (data-independent term) denotes an expectation with respect to the distribution by one-step reconstruction. κ denotes a momentum, and $\kappa \Delta W_{ij}^{k-1}$ is referred to as a 'momentum term'.

B. THE TRAINING PROCESS OF DBN

The unsupervised Wake stage and supervised Sleep stage constitute the whole training process of a DBN [33]. Fig.2 presents the graphic interpretation of this process. As illustrated, the top layer (fused by soft-max) outputs the final fault classification result, and expression '39000 × 6' indicates 'sample number×IMU variable dimension' (please refer to Part B in Section 4 for details).

1) WAKE STAGE

The tasks of Wake stage are performed by unsupervised training as impinge upon each RBM model from bottom

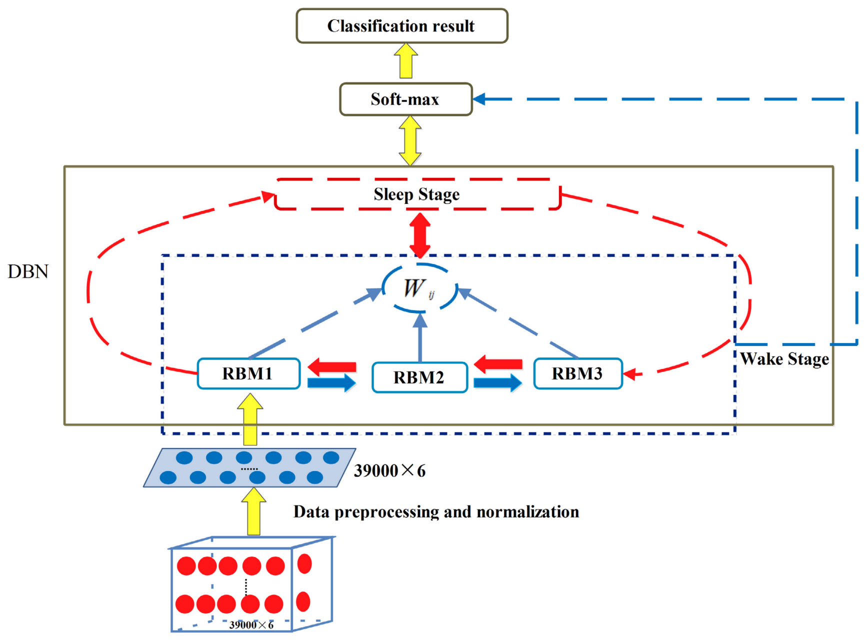


FIGURE 2. **Training process of DBN.**

to top. The weight $\theta = \{W, a, b\}$ of each RBM (3 in total) is consequently achieved.

2) SLEEP STAGE

The substance of Sleep stage is to fulfill the fine-tuning of weight value W_{ij}. We may proceed by observing that the weight $\theta = \{W, a, b\}$ is solvable after one-step Wake stage. Now, the classifier 'Soft-max' can collect the outputs of the last RBM (function as the inputs of Soft-max) and further output the fault label-type results [34]. Note that, the raw outputs of RAM are binary-type, Soft-max favorably fulfills a 'binary data to classification label' mapping according to the known fault labels (provided by IMU fault samples). The predicated fault classification results by Soft-max are then compared with the known fault categories, with being fed back to the lower 3 RBMs via back propagation strategies, the fine-tuning of W_{ij} for each RBM is accomplished, i.e., the fault classification results are consequently optimized.

III. OPTIMIZED DBN DESIGN

There are several difficulties that confront the designers of such a DBN model for sensor fault diagnosis. One source of the difficulty is that assigning initial weight values W_{ij} for each RBM is generally restricted to a randomly given range. Moreover, the determination of neuron number for each DBN hidden layer is highly dependent on repeated tests. On the ground of these challenging issues (more precisely, challenges associated with neuron number to be determined and weight value to be optimized), it has been very difficult to improve the training efficiency or classification accuracy of engineering application-oriented DBNs. We therefore concentrate on an optimized DBN design, which fuses an inexact linear searching algorithm- genetic algorithm (inexact LSA-GA) based weight value optimization and a BA based dynamic number adjustment of hidden-layer neuron.

A. INEXACT LSA-GA BASED WEIGHT VALUE OPTIMIZATION

The use of term 'inexact LSA-GA' could suggest a fusion of inexact linear searching algorithm and genetic algorithm ideas. In principle, inexact LSA-GA aims to ① efficiently optimize the searching step length α_k (searching interval of W_{ij} for 3 RBMs) via initially finding a searching direction p_k and ② accurately update the weight value W_{ij} via determining probabilities for 3 genetic operations (including reproduction, crossover and mutation) of typical GA. As illustrated, the essential part of inexact LSA-GA consists in its determination of a sufficiently degraded function value along the searching direction by an inexact linear searching idea. It is desired to derive a big enough searching step length so that inexact LSA can continuously reduce the step length in an iterative manner. Owing to the powerful evolutionary mechanisms, the introduction of GA is to refine the end searching process of W_{ij}. This could be illustrated by two searching areas in Fig. 3. Define the searching area of inexact LSA by (e, d), and define the searching area of GA by (e', d'), thus, it would need to be $d' - e' = k * (d - e)$ if it were desired to employ GA for better weight value optimization efficiency. It is also experimentally illustrated that conversion factor $k = 0.3$ corresponds to optimal results.

To demonstrate this method, it will be used to refer to the concept of loss function, since inexact LSA-GA aims to optimize W_{ij} by means of the evaluation of a loss

TABLE 1. Major steps of inexact LSA-GA based weight value optimization.

Major Steps:
(1) Define the searching area (e, d) of inexact LSA, and then iteratively narrow (e, d) to (e', d').
(2) Invoke GA if $d' - e' = k * (d - e)$ is satisfied. The population exist in (e', d') is iteratively updated via the following step 1) and 2) until the solution precision E is satisfied:
1)Determine the fitness function and coding pattern
Directly designate $M = \frac{1}{e}$ as fitness function, so that a smaller $L(\theta)$ denotes a higher value of matched individual fitness. Concerning the large number of individuals, the real encoding method is adopted.
2) Invoke selection, crossover, and mutation operators to generate offspring individuals, i.e., the next generation population.
(3) The derived individual (binary string, chromosome) that satisfies the desired solution precision is designated as the final result of GA, being referred to as the optimal solution of W_{ij}.

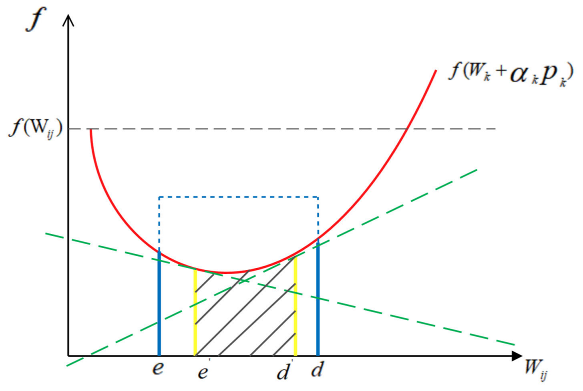


FIGURE 3. Searching idea of inexact LSA-GA.

function $L(\theta)$, it follows that

$$L(\theta) = \sum_{i=1}^{n} logP(v_i|\theta) = \sum_{i=1}^{n} [logf(v_i|\theta) - logZ(\theta)] \quad (6)$$

The searching direction p_k should be fully taken into account. It would need to meet Eq. (7) and gradient descent simultaneously, i.e., $p_k = -\Delta f(W_{ij})$, thus

$$\nabla f(W)^T p_k < 0 \quad (7)$$

where $f(\cdot) = \sum_{v,h} e^{-E(v,h|\cdot)}$ being an objective function. It should be stressed that, the input of v_i in $f(\cdot)$ could be any sample in fault dataset, whilst, the input of v_i in $Z(\theta)$ must be the label sample. The variable W_k (as in Fig.3) represents the current optimal solution of W_{ij}.

For the inexact LSA-GA that is represented by the above Fig.3, the determination of step leng α_k is identically significant since it essentially determines the optimizing efficiency of W_{ij}. To achieve this, α_k that is chosen for adapting to weight value optimization would subject to strong Wolf-Powell criterion as Eq.(8) indicates, also, it must meet Eq.(9) as follows

$$f(W_k + \alpha_k p_k) \leq f(W_k) + c_1 \alpha_k p_k \nabla f(W_k)^T,$$
$$||p_k^T \nabla f(W_k + \alpha_k p_k)^T|| \leq -c_2 p_k \nabla f(W_k)^T \quad (8)$$
$$f(W_k + \alpha_k p_k) < f(W_k), \alpha_k > 0 \quad (9)$$

where c_1 and c_2 are two constants with respect to strong Wolfe-Powell criterion. They satisfy $0 < c_1 < c_2 < 1$.

Denote the new optimal solution of W_{ij} by W_{k+1}(at sample time $k + 1$), then a simple iteration type of relationship exists between W_{k+1} and W_k, viz.

$$W_{k+1} = W_k + \alpha_k p_k \quad (10)$$

The fusion of typical GA aims at the goals of so-called 'refine searching', so that we can accelerate the weight value optimizing procedure. It is desired to fulfill the inexact LSA to GA conversion at the end of W_{ij} iteration process. The solution precision of GA is designated as $E = 0.001$, meanwhile, for better local searching performances, the area (e', d') is divided into M parts, i.e., $M = \frac{(e',d')}{E}$. We further present the major steps of this inexact LSA-GA based weight value optimization process, as given in Table 1.

The comparative analysis (on basis of same optimizing task and same time taken) between inexact LSA and inexact LSA-GA is illustrated by the different optimizing paths in Fig. 4. As we shall see, their main difference consists in the end optimizing mechanism- the inexact LSA aims to approach the target (red point) in a kinked chain manner by consistently optimizing step length α_k throughout the whole time period, whilst, the inexact LSA-GA maximizes its own superiors by refining the end optimizing, which leads to approach the target with more intensively kinked chains (more precisely, it occurs at last phase of optimizing process). The refined end optimizing result, however, appears to be much closer to the target point in same conditions of time taken for W_{ij} optimization.

B. BA BASED DYNAMIC NUMBER ADJUSTMENT OF HIDDEN-LAYER NEURON

The dynamic neuron-number adjustment will be described with chief reference to the adjustment that is taken on the hidden layer of each RBM. Since the direct approach involves the iterations of momentum κ and learning rate ε, we therefore define a constructed vector (for problem describing) to be $x = (N_1, N_2, N_3, \kappa, \varepsilon) \in R^5$, where $N_i(i = 1, 2, 3)$ denote the hidden-layer neuron numbers to be determined. The optimizing of x may suggest a parallel evaluation of N_i, so that DBN could explicitly invoke some specific evolutionary learning tools.

We focus on the BA based optimum calculating. Analogous to standard GA, under BA framework, it is desired to define a large-enough bat population to describe the possible

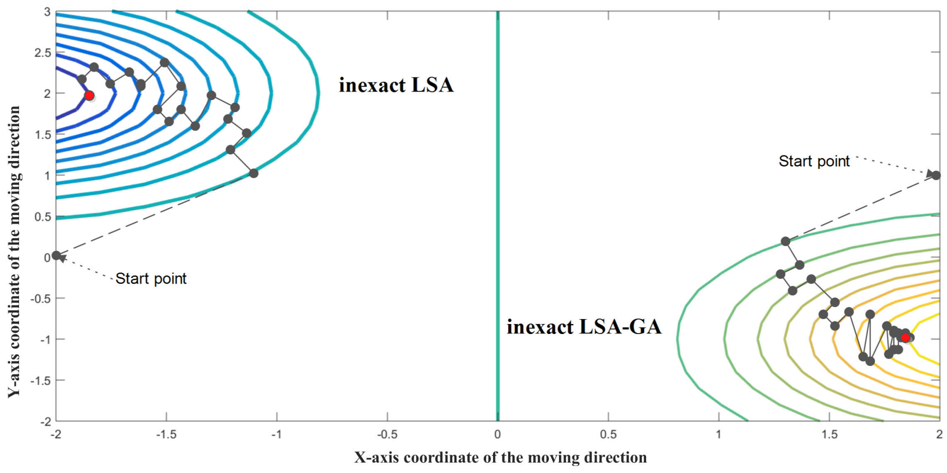


FIGURE 4. The optimizing path comparison between inexact LSA and inexact LSA-GA.

solutions of x, note here, x also represents the positions of individual bats [35]. Quite simple, a set of estimates (including κ, ε and N_i) of DBN after one-step training apparently supplies BA framework with initial values of x. The typical BA can be theoretically described by

$$\begin{aligned} f_i &= f_{min} + (f_{max} - f_{min})\beta, \\ v_i^t &= v_i^{t-1} + (x_i^{t-1} - x_*)f_i, \\ x_i^t &= x_i^{t-1} + v_i^t \end{aligned} \tag{11}$$

where f_i is the acoustic frequency emitted by the *ith* bat in the population,f_{max} and f_{min} respectively denote the maximum and minimum acoustic frequencies. At the initialization stage of BA, an acoustic frequency that yields to uniform distribution in $[f_{max}, f_{min}]$ domain is arbitrarily assigned to a bat. β is a random number uniformly distributed between [0, 1]. v_i^t and x_i^t respectively denote the speed and position of the *ith* bat at *tth* generation. $x*$ is the current optimal position, which in nature denotes the desired solution x when it is globally optimal.

The generation of global optimum $x*$ should primarily rely on the developed optimizing strategies. Define a threshold value p, we set: if $p > r_l$(pulse emissivity of the *lth* bat), a local searching strategy (by Eq. (12)) is directly used to iteratively update the current position of concerned bat. Otherwise, it would need to adjust the loudness A_l & pulse emissivity r_l by Eq. (13) at first and further employ Eq. (12) to conduct a global searching task with an updated A^t, viz.

$$x_{new} = x_{old} + \eta A_t \tag{12}$$

$$\begin{aligned} A_l^{t+1} &= \lambda A_l^t, \\ r_l^{t+1} &= r_l^0[1 - exp(-\gamma t)] \end{aligned} \tag{13}$$

where x_{old} and x_{new} respectively denote the current position and updated position of concerned bat. Random number η obeys Gaussian normal distribution $N(0, 1)$.A^t denotes the average loudness of bat population at *tth* generation. A_l^t and A_l^{t+1} respectively denote the loudness of sound waves emitted by the *lth* bat at *tth* and $(t+1)th$ generation. r_l^{t+1} denotes the pulse emission frequency of the *lth* bat at $(t+1)th$ generation, and r_l^0 denotes the initial pulse transmission frequency. λ and γ are constants, being both less than 1.

Referring to Eq. (13), as t increases, A_l^t would linearly decrease and r_l would exponentially increase (never exceeds r_l^0), so that for any $0 < \eta < 1$,$r_l > 0$, when $t \to \infty$, then $A_l^t \to 0$ and $r_l \to r_l^0$. The loudness A_l^t has a natural physical interpretation. It usually decreases once a bat has found its prey, i.e., its optimized position. $A_l^t \to 0$ therefore may suggest that global optimum $x*$ is consequently achieved.

Fig.5 describes the detailed algorithm block diagram for wheeled robot fault diagnosis. The central blocks shows how information flows forward from RBM1 to classifier 'soft-max', where 6-dimensional vector (navigation parameters of 3-axis attitude and 3-axis velocity) functions as the input vector of RBM1.

The initialization of algorithm is based on initially making a set of assignments of, W_{ij}, a_i, b_j and x, which involves the neuron state determination for visible layer and hidden layer of RBMs, viz., we can solve for bias a_i and b_j according to joint probability distribution expressions (as the sub-block at the upper left indicates). The sub-block 'BA' also explicitly presents the definition of vector x. As illustrated, each bat in population would completely execute Wake stage and Sleep stage for neuron number determination of each RBM hidden layer, which reflects the updating of acoustic frequency, speed and position of *ith* bat (as 'BA' sub-block indicates). In a sense, it is fair to say BA dominates the main algorithm framework of optimized DBN, i.e., multiplied 3-neuron-number combination results would imply different types of DBN structure. We emphasize that, optimizing W_{ij} via inexact LSA-GA drives the variations of κ and ε (as 'inexact LSA-GA' sub-block indicates), which then increasingly corrects the possible solutions of x (positions of individual bats). In addition, for a better network convergence

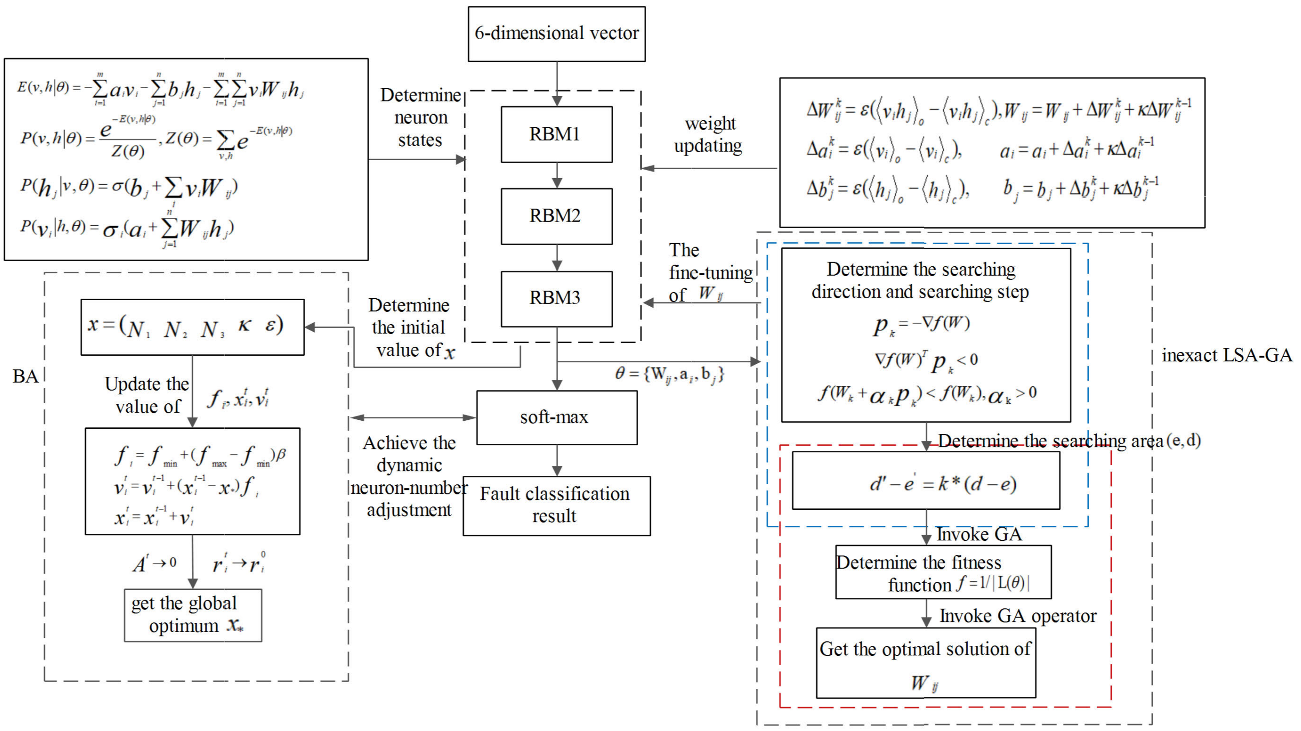

6-dimensional vector
Determine neuron states
RBM1
RBM2
RBM3
weight updating
The fine-tuning of W_{ij}
Determine the initial value of x
soft-max
Fault classification result
Achieve the dynamic neuron-number adjustment
$\theta = \{W_{ij}, a_i, b_j\}$
BA
$x = (N_1\ N_2\ N_3\ \kappa\ \varepsilon)$
Update the value of f_i, x_i^t, v_i^t
$A^t \to 0$
$r_i^t \to r_i^0$
get the global optimum x_*
inexact LSA-GA
Determine the searching direction and searching step
Determine the searching area (e,d)
Invoke GA
Determine the fitness function $f = 1/|L(\theta)|$
Invoke GA operator
Get the optimal solution of W_{ij}

FIGURE 5. The algorithm block diagram of optimized DBN.

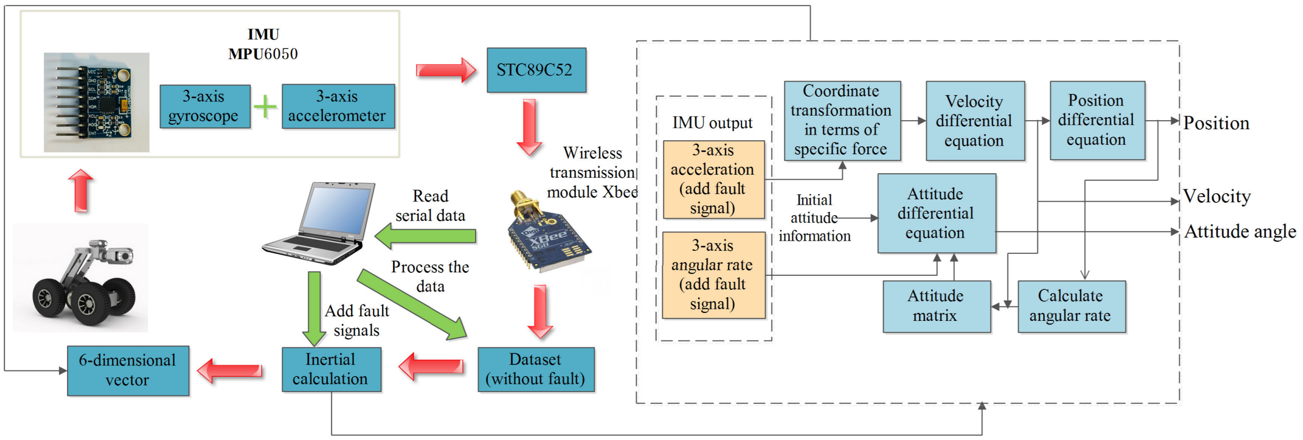

IMU MPU6050
3-axis gyroscope
3-axis accelerometer
STC89C52
Wireless transmission module Xbee
Read serial data
Process the data
Add fault signals
6-dimensional vector
Inertial calculation
Dataset (without fault)
IMU output
3-axis acceleration (add fault signal)
3-axis angular rate (add fault signal)
Initial attitude information
Coordinate transformation in terms of specific force
Velocity differential equation
Position differential equation
Attitude differential equation
Attitude matrix
Calculate angular rate
Position
Velocity
Attitude angle

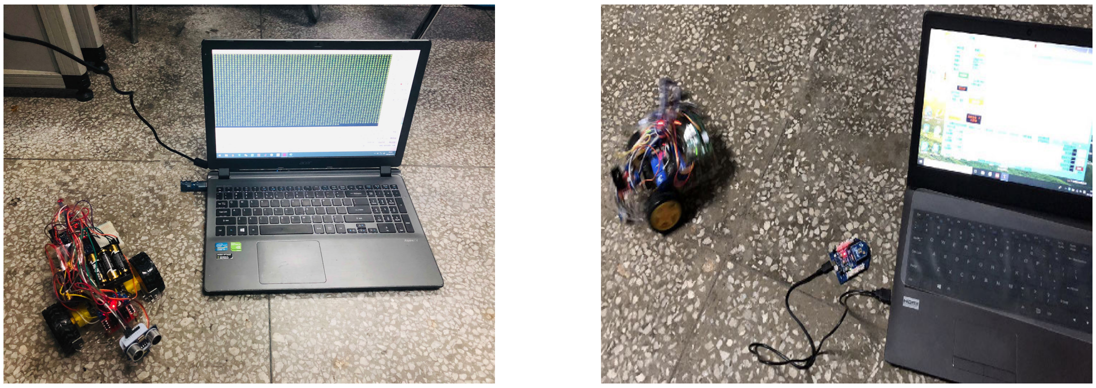
FIGURE 6. The schematic diagram of wheeled robot fault diagnosis modeling.

purpose, the typical weight adjustment strategy with inertial terms $\kappa\Delta W_{ij}^{k-1}, \kappa\Delta a_i^{k-1}$ and $\kappa\Delta b_j^{k-1}$ is allowed to iteratively update $\theta = \{W, a, b\}$, see details in the sub-block at the upper right.

IV. WHEELED ROBOT FAULT DIAGNOSIS MODELING

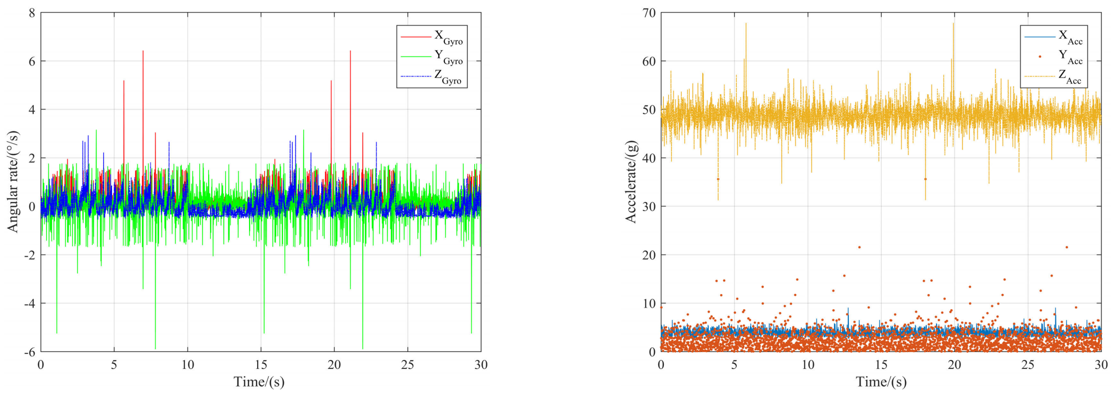
The IMU fault diagnosis modeling consists of a laboratory experiment based navigation information collecting and a numerical simulation based fault dataset creating. Concerning the physical sizes of sensors embedded in robot base, a low-cost integrated chip MPU6050 which functions as the element for physical motion measurement of robot is invited. MPU6050 that enables extremely small mechanical structures to be fabricated into bulk materials is composed of a 3-axis MEMS (abbreviates'micro-electromechanical systems') accelerometer, a 3-axis MEMS gyroscope, a 16-bit ADC converter, a 16-bit digital temperature sensor and a digital motion processor.

It is desired that, under laboratory experiment, MPU6050 provides the navigation computer with 3-axis angular rate and 3-axis acceleration. The collected navigation parameters are then used for the simulated fault dataset construction. The block diagram that favorably fuses sensor functions and algorithms is presented in Fig.6. It mainly involve



(a) Initial phase of indoor experiment (b) The robot is in uniform circular motion

FIGURE 7. Experimental data acquisition process.



(a) Gyroscope output (b) Accelerator output

FIGURE 8. Under uniform linear motion state (with no fault).

the experimental data's acquision, the data pre-processing, the addition of simulated faults and the inertial calculation for attitude and velocity estimates.

A. NAVIGATION INFORMATION COLLECTING VIA MPU6050

The laboratory experiments are conducted under robot kinestates in terms of uniform linear motion and uniform circular motion. A single chip microcomputer STC89C52 is firstly invited, which functions as a link between a MPU6050 module and a wireless transmission module Xbee (whose maximum communication rate is 250 kbps). In sequence, they are both mounted in a slot (as a stationary base on robot) as a united whole. Now, the preparations for navigation parameter collecting at fixed sampling points are implemented. Note that, another matched Xbee module should be directly connected with the navigation computer via a USB interface (See Fig.7 (b)), i.e., all the experimental data transmission is accomplished in a wireless manner (MPU6050 adopts 400 kHz I^2C communication interface).

The indoors flat marble floor is designated as the work envelope of laboratory experiments. Fig.7 records the experimental data acquisition process. As we shall see in Fig.8 and Fig.9, the collected navigation parameters are illustrated in terms of 3-axis gyroscope output and 3-axis accelerator output. We emphasize that, the collected parameters (3-axis angular rate and 3-axis acceleration) would not be directly used as the inputs of optimized DBN (even though they are substantially smoothed). We shall immediately address the reasons in the following Part B. Before that, let us start from the uncertainty factor analyses.

The known uncertainty factors and sources of error may include ① static drift noise of gyroscopes and accelerators. Especially for MEMS gyroscopes of MPU6050, their drifts account for a large proportion of such error; ② gyroscope random error. Due to the drifts of MEMS gyroscopes and the limited measurement environments, gyroscope random error generates during the operation of MPU6050; ③ incorrect accelerator output. Due to the fact that the marble floor may be not flat enough, when the wheeled robot makes contact with the ground, it perhaps generates some forms of acceleration fluctuation and derives larger accelerator output than the true value; ④ occasional data transmission delays. It is mainly due to the cause that the wireless

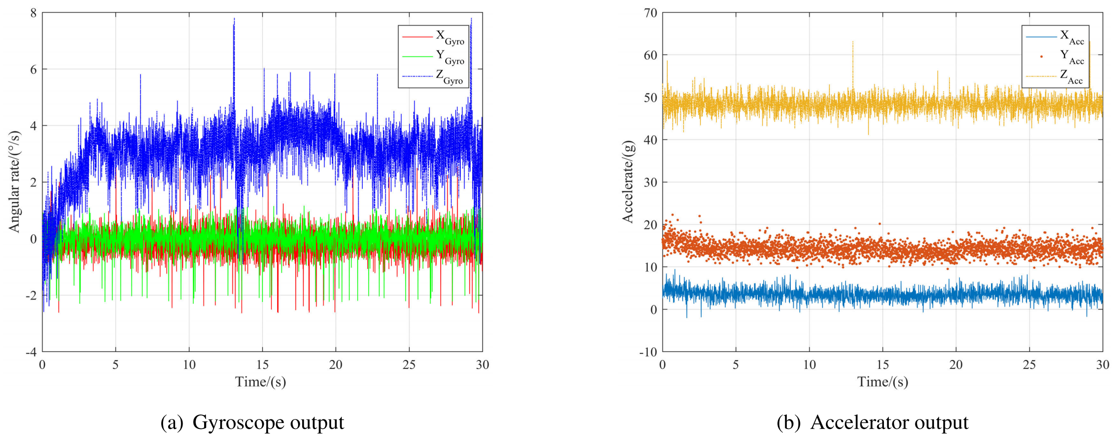


FIGURE 9. Under uniform circular motion state (with no fault).

transmission of Xbee module is influenced by electromagnetic interferences.

The solutions to the elimination of influences produced by above uncertainty factors are to conduct multiple sets of data pre-processing. To begin with the experiment, MPU6050 was mounted on a stationary base on a two-axis horizontal turntable. As the initial alignments of gyroscopes and accelerometers were accomplished, we collected 2000 sets of data, calculated the average value and subtracted from the raw data. The results were considered to be the actual outputs of gyroscopes and accelerometers and were subsequently recorded. In sequence, the density-based local outlier data mining algorithm-Local Outlier Factor (LOF) was invited to eliminate the outliers. Specifically, for sampling data integrity, every single outlier was replaced by the calculated average value of 50 sets of data that were adjacent to the outlier from front and rear. Now, the data (3-axis angular rate and 3-axis acceleration) may then be normalized for the following simulated fault dataset creating, so as to further eliminate the possible influences produced by data forms. It was linearly formulated in terms of min-max normalization,

$$y = \frac{x - MinValue}{MaxValue - MinValue} \tag{14}$$

where x and y respectively denote the original data and normalized data. $MinValue$ and $MaxValue$ respectively represent the minimum and maximum in the original datasets.

B. SIMULATED FAULT DATASET CREATING

The creating of fault datasets, as in Fig.6, is to implement the simulated fault signal addition, which is mainly about the signals that are manually added to the derived 3-axis angular rate and 3-axis acceleration. Note that, these parameters cannot directly reflect the behaviors of IMUs, since we merely aim at the fault detection and fault diagnosis of IMUs (rather than integrated systems, like GPS/IMUs, visual odometer/IMUs, etc.), we therefore choose the equivalent attitude angel (with fault) estimates and velocity (with fault) estimates as the input variables of optimized DBN, as illustrated by the top block (6-dimensional vector) in Fig.5.

Let us begin the simulated fault addition process by determining the components of signals that are derived from inertial sensors. Note that, as in Fig.6, the essential part of inertial calculation consist in performing the data pre-processing beforehand, and the purpose of inertial calculation is to fulfill the conversion of 3-axis angular rate (with simulated fault) & 3-axis angular acceleration (with simulated fault) to equivalent attitude angel and velocity form.

For the signals derived from the inertial sensors, they can be mathematically described as

$$y_R(\mu) = y(\mu) + f(\mu) + v(\mu) \tag{15}$$

where μ denotes run time of inertial sensors.$y_R(\mu)$ denotes the actual output, $y(\mu)$ denotes the theoretical (or real) output, $f(\mu)$ denotes the added fault, $v(\mu)$ denotes the noise. Generally, the noise may come from the internal sensors or the external work volumes. Moreover, it may be random and unpredictable at every single moment.

The fault type identification of the inertial sensors could be complex due to the diversity of faults, but for the purpose of fault analysis, three distinct idealized types are possible. These are abrupt fault (element parameters suddenly exhibit large deviation that cannot be monitored or predicted beforehand), soft fault (individual parameter values of elements slowly change with the variation of time and environment) and clearance fault (time-lapse fault caused by aging, insufficient tolerance or poor contact). According to the above fault attributes (hard fault clearance is of no concern), the constructed $f(\mu)$ may be

(1) Deviation fault

Deviation fault could imply a sudden fault. Concerning the system excitation, the severe step-type function is a certainly reminiscent. For the amplitude of step function, we denote by constant c and define

$$f(\mu) = c \tag{16}$$

(2) Drift fault

Drift occurs normally in MEMS IMUs, as one of typical soft faults, which may be a function of $\Delta\mu$ (the time kept since fault occurs) and be written in general linear form, viz.

$$f(t) = \zeta * \Delta\mu \tag{17}$$

where ζ is a random number.

The simulated fault magnitude would need to be taken into account so as to precisely identify the fault intensity and fault occurring time. For the faults represented above, through multiplied laboratory tests, we set $c = 4\ deg$ and $\zeta = 11$, which identically enables the faulty states to be identified and to be available for the further optimized DBN fault diagnosis.

Taking the estimated attitude (Euler angle: pitch, roll and yaw) and 3-axis velocity under uniform circular motion state as example, we present the comparative analysis results of drift fault (located at X-axis of MEMS gyroscope) occurrence, deviation fault (located at X-axis of MEMS accelerator) occurrence and IMU fault-free circumstances, as respectively shown in Fig.10 and Fig.11.

Fig.10 (a) and Fig.11 (a) indicate that in cases where the axial gyroscope drift faults occur, the attitude estimates of wheeled robot appear to be strongly fluctuant or even worse (non-convergent), whilst, the 3-axis velocity estimates maintain almost unchanged curve tendencies with respect to variation of generated 3-axis angular rate. This is mainly due to the fact that the gyroscopes are not used in the propagation of velocity vector, and only used as a direct attitude reference. Fig.10 (b) and Fig.11 (b) represent that, the axial accelerator deviation fault may lead to a change in velocity vector, but its one principal attribute is its relevance of estimated attitude vector. As with typical IMU navigation strategies, accelerators may generally be used for attitude updating. Clearly, the constructed datasets that have characteristics of data coupling fairly meets the fault diagnosis requirements for certain data association concerned circumstances.

Table 2 gives the detailed dataset descriptions in terms of fault category, fault location and fault type. Dataset A and dataset B respectively correspond to uniform linear motion and uniform circular motion. For each row of vector-type data, it involves the estimated orthogonal attitude angles (pitch, roll and yaw, all in °) and orthogonal 3-axis velocity (in m/s). Concerning the efficient representation, storage and retrieval of sufficient sets of data for the following optimized DBN classification tests, for each fault category (13 in total, as in Table 2), it would contain 3000 sets of data (each one is in vector space R^6), so that the amount of data in dataset A (the same to dataset B) would be equal to $3000 \times 13 = 39000$ in total. To take a specific example, as in Table 2, example items in Category 1of such datasets A and B might be representations for such fact as "X-axis of certain gyroscope malfunctions", "The bias fault is considered to be happening with $c = 4\ deg$", "The estimated navigation parameters in first and last (3000*th*) sampling instants in dataset A are [1.0872 − 0.2686 0.1813 0.0187 0.0086 0.1550] and [1.0573 −0.2687 0.1832 0.0147 0.0045 0.1548] respectively", "[1.0872 − 0.6686 1.1813 0.0106 0.0586 0.0195] and [1.0632 −0.6531 1.1766 0.0184 0.0512 0.0179] for those in dataset B", etc.

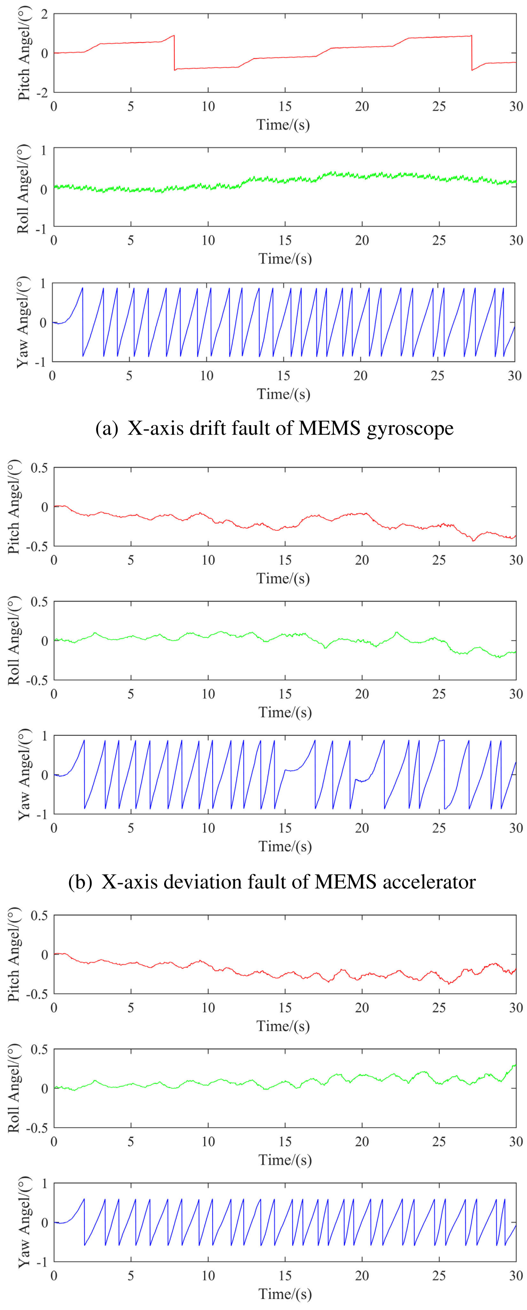


FIGURE 10. **Attitude estimates under uniform circular motion.**

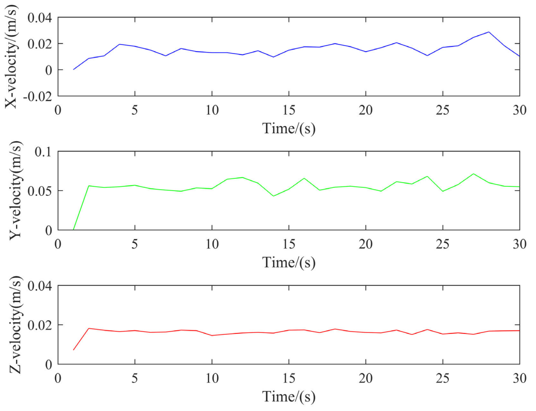
In the following section for optimized DBN performance analysis, we set 70% of the total data volume in both dataset A and dataset B functions as training sets, 20% functions as test sets and the remaining 10% is for validation sets.

V. PERFORMANCE ANALYSIS OF OPTIMIZED DBN FOR FAULT DIAGNOSIS

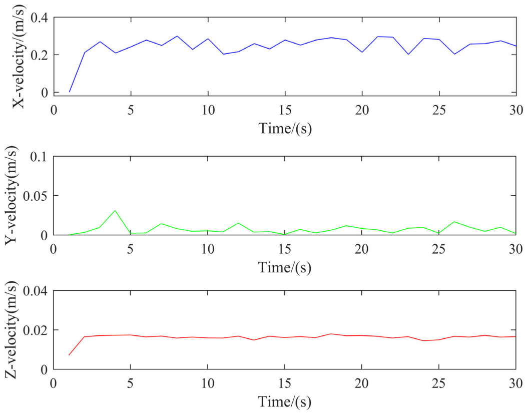
As in Table 2, the data features in dataset A and dataset B appear to be different even though same fault category and

TABLE 2. **Dataset descriptions.**

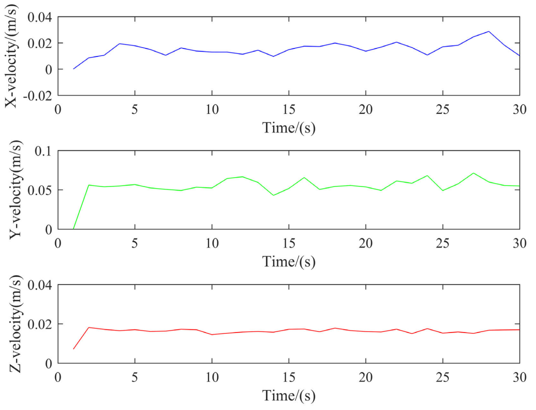
Fault category	Fault location	Fault type	Data structure		Fault category	Fault location	Fault type	Data structure	
			Dataset A	Dataset B				Dataset A	Dataset B
1	Gyroscope X-axis	Deviation fault	$\begin{pmatrix} 1.0872 & -0.2686 & 0.1813 & 0.0187 & 0.0086 & 0.1550 \\ 1.0755 & -0.2609 & 0.1824 & 0.0159 & 0.0046 & 0.1531 \\ 1.0659 & -0.2891 & 0.1767 & 0.0178 & 0.0064 & 0.1546 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1.0659 & -0.2775 & 0.1782 & 0.0112 & 0.0036 & 0.1731 \\ 1.0573 & -0.2687 & 0.1832 & 0.0147 & 0.0045 & 0.1548 \end{pmatrix}$	$\begin{pmatrix} 1.0872 & -0.6686 & 0.0106 & 0.0106 & 0.0586 & 0.0195 \\ 1.0755 & -0.6609 & 1.1824 & 0.0159 & 0.0546 & 0.0189 \\ 1.0700 & -0.6587 & 1.1796 & 0.0187 & 0.0551 & 0.0182 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1.0659 & -0.6575 & 1.1782 & 0.0112 & 0.0536 & 0.0183 \\ 1.0632 & -0.6531 & 1.1766 & 0.0184 & 0.0512 & 0.0179 \end{pmatrix}$	7	Accelerometer X-axis	Deviation fault	$\begin{pmatrix} 0.4720 & -0.2851 & 0.1795 & -1.2953 & 0.0014 & 0.5157 \\ 0.5559 & -0.2819 & 0.1775 & -1.2468 & 0.0059 & 0.5638 \\ 0.4981 & -0.2764 & 0.1789 & -1.2781 & 0.0000 & 0.5517 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.5788 & -0.2300 & 0.1724 & -1.2102 & 0.0019 & 0.5679 \\ 0.4329 & -0.2616 & 0.1768 & -1.2632 & 0.0045 & 0.5504 \end{pmatrix}$	$\begin{pmatrix} -0.3752 & -0.0648 & 0.7501 & 0.2311 & 0.0576 & 0.0191 \\ -0.3937 & -0.0941 & -0.7501 & 0.2097 & 0.0614 & 0.0183 \\ -0.4011 & -0.1042 & 0.7501 & 0.2419 & 0.0701 & 0.0187 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.4413 & -0.1088 & -0.7501 & 0.2921 & 0.0486 & 0.0193 \\ -0.4906 & -0.1307 & 0.7501 & 0.2416 & 0.0341 & 0.0174 \end{pmatrix}$
2		Drift fault	$\begin{pmatrix} 2.0350 & -0.4703 & 0.1524 & 0.0187 & 0.0086 & 0.1550 \\ 2.6236 & -0.4778 & 0.1324 & 0.0159 & 0.0046 & 0.1531 \\ 2.4671 & -0.4421 & 0.1472 & 0.0178 & 0.0064 & 0.1546 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 2.8136 & -0.4365 & 0.1078 & 0.0112 & 0.0036 & 0.1736 \\ 2.7816 & -0.4615 & 0.1217 & 0.0147 & 0.0045 & 0.1548 \end{pmatrix}$	$\begin{pmatrix} 0.8350 & 0.3703 & 0.9461 & 0.0106 & 0.0586 & 0.0195 \\ 0.8436 & 0.3778 & -0.9461 & 0.0159 & 0.0546 & 0.0189 \\ 0.9490 & 0.4987 & 0.9461 & 0.0187 & 0.0551 & 0.0182 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.9490 & 0.5973 & 0.9461 & 0.0112 & 0.0536 & 0.0183 \\ -0.8436 & 0.5635 & -0.9461 & 0.0184 & 0.0512 & 0.0179 \end{pmatrix}$	8		Drift fault	$\begin{pmatrix} 0.9573 & 0.3645 & 0.3418 & 0.2001 & 1.2214 & 0.4417 \\ 0.9561 & 0.3848 & 0.3897 & 0.2034 & 1.1189 & 0.4671 \\ 0.9681 & 0.3612 & 0.3964 & 0.2451 & 1.0031 & 0.4597 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.9777 & 0.3796 & 0.3674 & 0.2310 & 1.3064 & 0.4421 \\ 0.9857 & 0.3419 & 0.3761 & 0.2191 & 1.2019 & 0.4298 \end{pmatrix}$	$\begin{pmatrix} -0.4925 & -0.5415 & 1.5702 & 0.3745 & 0.0524 & 0.1479 \\ -0.3021 & -0.5219 & 1.6026 & 0.4511 & 0.0488 & 0.1689 \\ -0.4413 & -0.5386 & 1.6781 & 0.5617 & 0.0452 & 0.1543 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.3385 & -0.5987 & 1.5495 & 0.5424 & 0.0618 & 0.1643 \\ -0.4817 & -0.5495 & 1.5341 & 0.5878 & 0.0672 & 0.1543 \end{pmatrix}$
3	Gyroscope Y-axis	Deviation fault	$\begin{pmatrix} 0.4892 & 2.0306 & 0.3822 & 0.0187 & 0.0086 & 0.1550 \\ 0.5727 & 2.0298 & 0.3832 & 0.0159 & 0.0046 & 0.1531 \\ 0.4978 & 2.0264 & 0.3612 & 0.0178 & 0.0064 & 0.1546 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.5933 & 2.0320 & 0.3788 & 0.0112 & 0.0036 & 0.1731 \\ 0.4871 & 2.1633 & 0.3712 & 0.0147 & 0.0045 & 0.1548 \end{pmatrix}$	$\begin{pmatrix} 0.4892 & 2.0306 & 1.1822 & 0.0106 & 0.0586 & 0.0195 \\ 0.5727 & 2.0298 & 1.1832 & 0.0159 & 0.0546 & 0.0189 \\ 0.5142 & 2.0176 & 1.1827 & 0.0187 & 0.0551 & 0.0182 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.5933 & 2.0320 & 1.1788 & 0.0112 & 0.0536 & 0.0183 \\ 0.4761 & 2.0541 & 1.1769 & 0.0184 & 0.0512 & 0.0179 \end{pmatrix}$	9	Accelerometer Y-axis	Deviation fault	$\begin{pmatrix} 0.8740 & 1.0749 & 0.2569 & 1.1155 & 0.8051 & 0.1003 \\ 0.8805 & 1.0211 & 0.1980 & 1.0978 & 0.5031 & 0.1069 \\ 0.9017 & 1.0761 & 0.2123 & 1.1902 & 0.7183 & 0.1150 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.7913 & 1.0167 & 0.1866 & 1.1104 & 0.8208 & 0.1169 \\ 0.8194 & 1.0924 & 0.2103 & 1.0498 & 0.6723 & 0.1201 \end{pmatrix}$	$\begin{pmatrix} -0.6883 & -1.1228 & 1.0601 & 0.1869 & 0.1781 & 0.1638 \\ -0.6752 & -1.0648 & -1.0012 & 0.1684 & 0.1698 & 0.1613 \\ -0.6547 & -1.0122 & 1.0601 & 0.1752 & 0.1732 & 0.1611 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.6937 & -1.0741 & 1.0601 & 0.1659 & 0.1892 & 0.1636 \\ -0.6781 & -1.1044 & -1.0012 & 0.1833 & 0.1784 & 0.1677 \end{pmatrix}$
4		Drift fault	$\begin{pmatrix} -0.7406 & 3.0128 & -0.2467 & 0.0187 & 0.0086 & 0.1550 \\ -0.7376 & 3.1491 & -0.2456 & 0.0159 & 0.0046 & 0.1531 \\ -0.7489 & 3.2406 & -0.2448 & 0.0178 & 0.0064 & 0.1546 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.7795 & 3.4900 & -0.2419 & 0.0112 & 0.0036 & 0.1736 \\ -0.7587 & 3.2619 & -0.2334 & 0.0147 & 0.0045 & 0.1548 \end{pmatrix}$	$\begin{pmatrix} -0.9406 & 3.0128 & -1.2467 & 0.0106 & 0.0586 & 0.0195 \\ -0.9376 & 3.1491 & -1.2456 & 0.0159 & 0.0546 & 0.0189 \\ -0.9189 & 3.0765 & -1.2431 & 0.0187 & 0.0551 & 0.0182 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.9095 & 3.4900 & -1.2419 & 0.0112 & 0.0536 & 0.0183 \\ -0.9129 & 3.3216 & -1.2378 & 0.0184 & 0.0512 & 0.0179 \end{pmatrix}$	10		Drift fault	$\begin{pmatrix} 0.6720 & 0.9168 & 0.5471 & 0.9112 & 0.4899 & 0.4366 \\ 0.6097 & 0.9487 & 0.3896 & 0.9412 & 0.5011 & 0.4417 \\ 0.6781 & 0.9423 & 0.4409 & 0.9365 & 0.5612 & 0.4869 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.6686 & 0.9277 & 0.4687 & 0.9017 & 0.5972 & 0.4981 \\ 0.6812 & 0.9617 & 0.4231 & 0.9644 & 0.5688 & 0.4098 \end{pmatrix}$	$\begin{pmatrix} -0.8936 & -0.9881 & 0.8315 & 0.0120 & 2.6686 & 0.1632 \\ -0.8410 & -0.6562 & -0.8318 & 0.0193 & 2.8944 & 0.1675 \\ -0.8647 & -0.7936 & 0.8315 & 0.0000 & 2.5673 & 0.1685 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.8409 & -0.9649 & 0.8315 & 0.0155 & 2.9471 & 0.1627 \\ -0.8861 & -0.7406 & -0.8518 & 0.0124 & 2.7846 & 0.1645 \end{pmatrix}$
5	Gyroscope Z-axis	Deviation fault	$\begin{pmatrix} 0.5024 & -0.2677 & -2.1758 & 0.0187 & 0.0086 & 0.1550 \\ 0.5856 & -0.2566 & -2.1982 & 0.0159 & 0.0046 & 0.1531 \\ 0.5912 & -0.2762 & -2.2348 & 0.0178 & 0.0064 & 0.1546 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.6032 & -0.2027 & -2.2185 & 0.0112 & 0.0036 & 0.1731 \\ 0.6078 & -0.2013 & -2.1764 & 0.0147 & 0.0045 & 0.1548 \end{pmatrix}$	$\begin{pmatrix} 0.5024 & -1.2677 & -2.1758 & 0.0106 & 0.0586 & 0.0195 \\ 0.5856 & -1.2566 & -2.1982 & 0.0159 & 0.0546 & 0.0189 \\ 0.5921 & -1.2431 & -2.2033 & 0.0187 & 0.0551 & 0.0182 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.6032 & -1.2027 & -2.2185 & 0.0112 & 0.0536 & 0.0183 \\ 0.6103 & -1.2003 & -2.2241 & 0.0184 & 0.0512 & 0.0179 \end{pmatrix}$	11	Accelerometer Z-axis	Deviation fault	$\begin{pmatrix} -0.6120 & 2.0936 & 0.2001 & 0.2201 & 1.6617 & 0.8098 \\ -0.6891 & 2.1011 & 0.1916 & 0.2486 & 1.6549 & 0.7271 \\ -0.6453 & 2.1089 & 0.2472 & 0.2716 & 1.5966 & 0.8974 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.6662 & 2.2478 & 0.1724 & 0.2267 & 1.6871 & 0.9114 \\ -0.6238 & 2.1964 & 0.2314 & 0.2710 & 1.6437 & 0.8271 \end{pmatrix}$	$\begin{pmatrix} -1.3551 & -0.4766 & 1.2797 & 0.0147 & 0.0559 & 0.6285 \\ -1.3629 & -0.4777 & -1.3083 & 0.0128 & 0.0526 & 0.6018 \\ -1.3412 & -0.4416 & 1.2797 & 0.0196 & 0.0541 & 0.6471 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -1.3812 & -0.4690 & 1.2797 & 0.0205 & 0.0608 & 0.6984 \\ -1.3744 & -0.4535 & -1.3083 & 0.0113 & 0.0561 & 0.6432 \end{pmatrix}$
6		Drift fault	$\begin{pmatrix} 1.0230 & -1.0785 & -2.6545 & 0.0187 & 0.0086 & 0.1550 \\ 1.2574 & -1.4369 & 2.6837 & 0.0159 & 0.0046 & 0.1531 \\ 1.1973 & -1.2611 & -2.6545 & 0.0178 & 0.0064 & 0.1546 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1.4016 & -1.3654 & -2.6545 & 0.0112 & 0.0036 & 0.1731 \\ 1.0729 & -1.2978 & 2.6837 & 0.0147 & 0.0045 & 0.1548 \end{pmatrix}$	$\begin{pmatrix} -1.2301 & -1.0785 & -2.6545 & 0.0106 & 0.0586 & 0.0195 \\ -1.2574 & -1.4369 & -2.6837 & 0.0159 & 0.0546 & 0.0189 \\ -1.2134 & -1.4956 & -2.6461 & 0.0187 & 0.0551 & 0.0182 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -1.2016 & -1.8654 & -2.8547 & 0.0112 & 0.0536 & 0.0183 \\ -1.2109 & -1.8731 & -2.7408 & 0.0184 & 0.0512 & 0.0179 \end{pmatrix}$	12		Drift fault	$\begin{pmatrix} 1.3849 & 0.2976 & 0.7432 & 0.3013 & 0.9001 & 0.6630 \\ 1.4791 & 0.3100 & -0.7782 & 0.3341 & 0.8927 & 0.6476 \\ 1.3146 & 0.2998 & 0.7432 & 0.3169 & 0.8841 & 0.6791 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1.2469 & 0.2517 & -0.7782 & 0.3572 & 0.9036 & 0.6740 \\ 1.4121 & 0.3167 & 0.7432 & 0.3481 & 0.8914 & 0.6895 \end{pmatrix}$	$\begin{pmatrix} -0.5409 & -0.6964 & 0.9096 & 0.1557 & 0.0465 & 0.8098 \\ -0.5633 & -0.6009 & -0.9018 & 0.1151 & 0.0539 & 0.8187 \\ -0.5435 & -0.6433 & 0.9069 & 0.1946 & 0.0442 & 0.8914 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.5827 & -0.6814 & 0.9069 & 0.1135 & 0.0546 & 0.8998 \\ -0.5453 & -0.6641 & -0.9018 & 0.1142 & 0.0611 & 0.8769 \end{pmatrix}$
13	Without fault		$\begin{pmatrix} 0.4720 & -0.2851 & 0.1795 & 0.0187 & 0.0086 & 0.1550 \\ 0.5559 & -0.2819 & 0.1775 & 0.0159 & 0.0046 & 0.1531 \\ 0.4981 & -0.2764 & 0.1789 & 0.0178 & 0.0064 & 0.1546 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.5788 & -0.2300 & 0.1724 & 0.0112 & 0.0036 & 0.1531 \\ 0.4329 & -0.2616 & 0.1768 & 0.0147 & 0.0045 & 0.1548 \end{pmatrix}$	$\begin{pmatrix} -0.4720 & 0.3851 & 0.7951 & 0.0106 & 0.0586 & 0.0195 \\ -0.5559 & 0.3819 & -0.7951 & 0.0159 & 0.0546 & 0.0189 \\ -0.4651 & 0.2891 & 0.7951 & 0.0187 & 0.0551 & 0.0182 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.5788 & 0.2300 & 0.7951 & 0.0112 & 0.0536 & 0.0183 \\ -0.4658 & 0.3451 & -0.7951 & 0.0184 & 0.0512 & 0.0179 \end{pmatrix}$					



(a) X-axis drift fault of MEMS gyroscope



(b) X-axis deviation fault of MEMS accelerator



(c) IMU fault-free state

FIGURE 11. Velocity estimates under uniform circular motion.

fault location of individual sensors are considered, thus, we create 26 labels that correspond to 26 possible faults, where 13 for dataset A, 13 for dataset B.

A. SIMULATION ANALYSIS OF DYNAMIC ADJUSTMENT PROCESS OF DBN MODEL

As stated above, the adjustable DBN model refers to the dynamic number adjustment of its hidden-layer neuron. To begin with the simulation test, we set the initial neuron number of 3 RBM hidden layer to be '11-7-21', and set the initial values of ε and κ to be both 0.01. It should be emphasized that, in our design, the training and optimization of DBN is accomplished via sufficient and heuristic trainings of each RBM within 100 epochs (generations). The dynamic adjustment process of DBN model is illustrated by the relation curves between adjustable hidden-layer neuron number of 3 RBMs (indicated on the horizontal axis) and estimated fault classification accuracy (indicated on the vertical axis), as shown in Fig.12 and Fig.13.

TABLE 3. Variable values in bat algorithm.

	ε		κ	
	Dataset A	Dataset B	Dataset A	Dataset B
Initial value	0.01	0.01	0.01	0.01
Optimized value	0.015	0.01	0.7	0.7

The explicit structure of this DBN could be initially represented by '6-11-7-21-13' with fixed 6 (input neurons as indicated in Fig.5) and fixed 13 (output neurons, representing 13 fault categories) and adjustable '11-7-21'.

Various ways in which hidden-layer neuron number of 3 RBMs can be combined lead to different correct rate results. For example, for Category 1(fault 1) in dataset A, when the hidden-layer neuron number combination is 7-14-15 (as illustrated in Fig.12(a)), the highest correct rate 94.8% is achieved. Since there is contextual correlation between data in both datasets, for fault 2, the fixed combination 7-14-15 therefore identically leads to the highest correct rate 94%. For fault 9 and fault 11, the numbers can both reach up to 97.6%. For no fault circumstance (represented by Category 13), this correct rate peaks to 97.9%, that is, a set of samples which are equivalent to an error of 2.1% in accurate rate have been misclassified. This is mainly for the reasons that ① the inertial calculation weakens the feature differences between samples (6-dimensional fault vectors) ② the derived fault categories by the classifier 'Soft-max' with probability calculations appear to be multivalued, viz., one single fault could be simultaneously classified into more than one category.

Analogously, 3 representative faults including fault 1, fault 9 (corresponds to the highest correct rate except for fault 13) and fault 13 (indicates no fault is happening) are chosen as an illustration of how the dynamic adjustments of possible hidden-layer neuron number combinations perform in dataset B. The corresponding relation curves are given in Fig.13. By a method similar to that adopted for classification accuracy analysis on dataset A, the calculated hidden-layer neuron number combination in this case is '6-7-27'. Also, the corresponding value changes of variables ε and κ in bat algorithm during this adjustment process are given in Table 3.

As stated, 20% of the total data is used for test sets, we further present the fault diagnosis accuracy results with the given test sets on both dataset A and dataset B, as illustrated by the detailed accuracy estimates of 13 faults in Fig.14. For a better

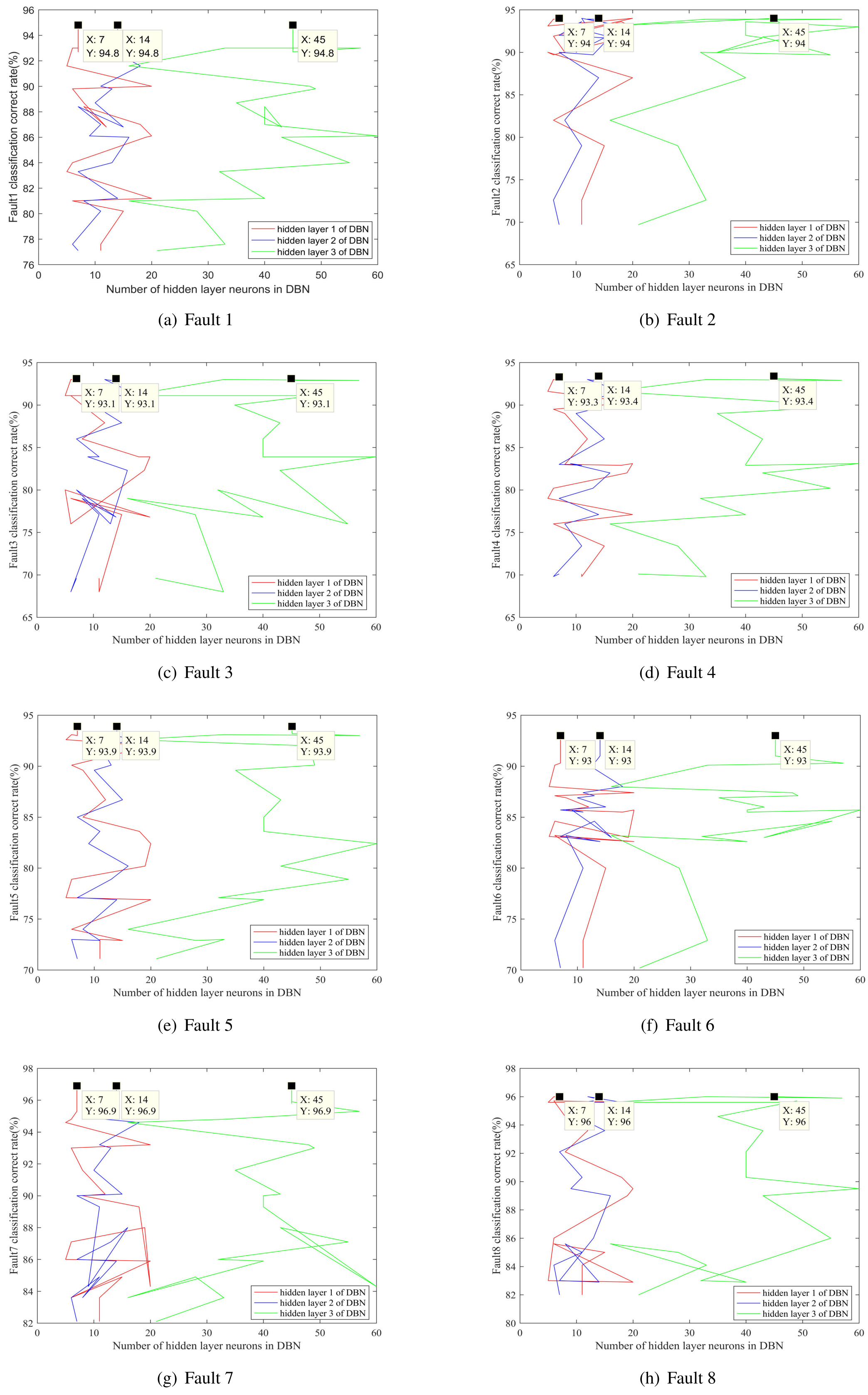
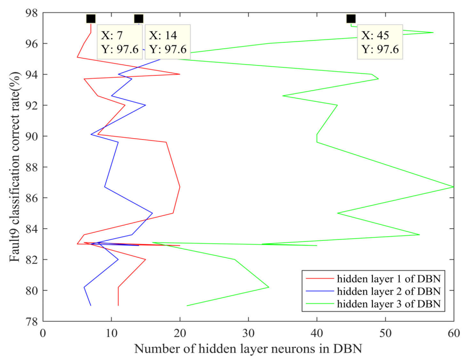
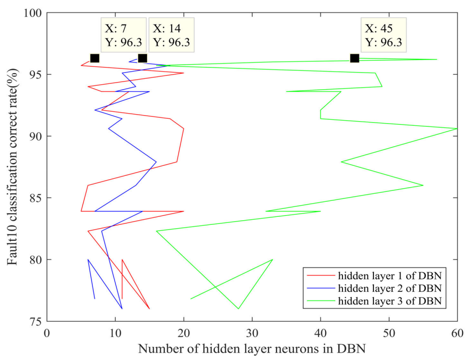


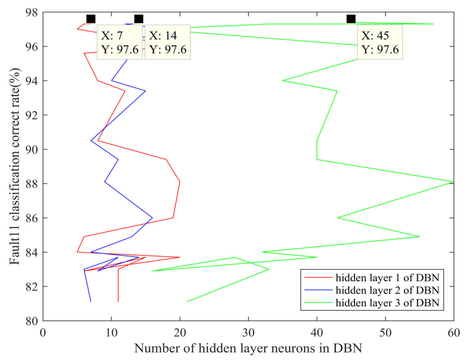
FIGURE 12. **The relation curves between adjustable hidden-layer neuron number of 3 RBMs and estimated fault classification correct rate for 13 types of faults (Dataset A).**



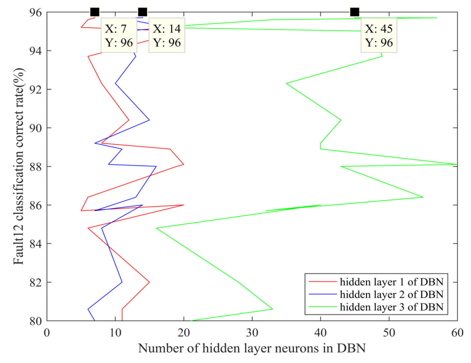
(i) Fault 9



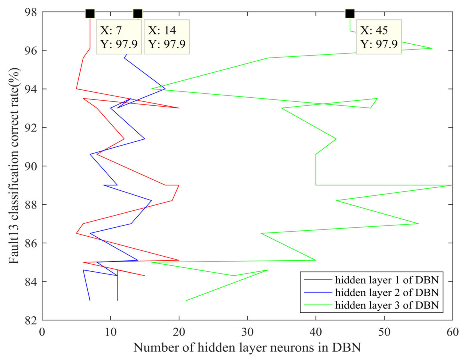
(j) Fault 10



(k) Fault 11



(l) Fault 12



(m) Fault 13

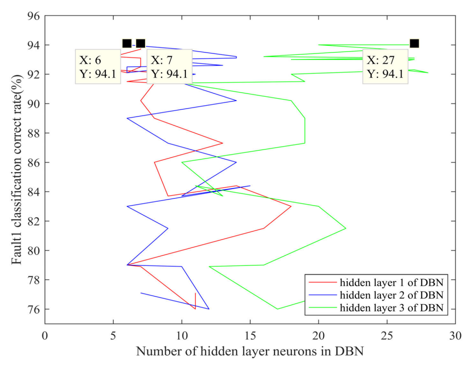
FIGURE 12. ***(Continued.)*** **The relation curves between adjustable hidden-layer neuron number of 3 RBMs and estimated fault classification correct rate for 13 types of faults (Dataset A).**

quantitative interpretation of fault diagnosis accuracy estimates, we define a threshold value N ($N \in [0.0001, 0.0009]$). For the given N, we apply a set of rules:

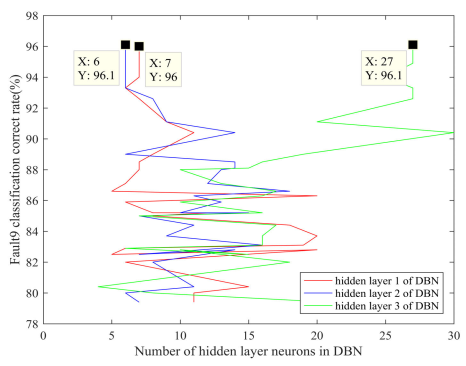
① For every single 13-dimentional vector derived form classifier 'soft-max', if the differences between the maximum and other elements of the vector are bounded on [0.0001,0.0009], then it is permitted that one single sample (one known category) can be classified into up to 3 categories. Among this 3, if the correct category (since the category for testing is known beforehand) is included, then the sample will be denoted by 'inexact classifying samples (ICS)'. Otherwise, with correct category not included, we denote by 'wrong sample'.

② If 4 or more than 4 categories that a single sample may belong to are determined, that sample will be directly denoted by 'wrong sample'. It is mainly because that, in fault diagnosis field, 4 (or more than 4) categories that are simultaneously derived by one estimator could suggest an enlarged range of fault correcting. It makes no sense for the fault diagnosis efficiency.

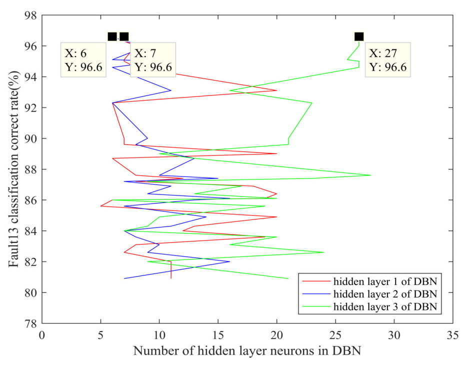
③ The fault accuracy estimates of ICS follow the basic rules: For every single sample that is simultaneously classified into 2 categories, we denote the correct classifying number by '0.5'. Similarly, for every single sample that may belong to 3 categories, we denote by '0.33'.



(a) Fault 1



(b) Fault 9



(c) Fault 13

FIGURE 13. The relation curves between adjustable hidden-layer neuron number of 3 RBMs and estimated fault classification correct rate for 13 types of faults (Dataset B).

TABLE 4. Classifying result of Fault 1(Dataset A).

	Case of correctly classified		ICS case						Wrong case
			2 categories				3 categories		
Sample number and fault category	568	Fault 1	1	Fault 1	3	Fault 1	2	Fault 1	26
				Fault 3		Fault 13		Fault 2	
								Fault 13	
Fault diagnosis accuracy	$(568 + 4 \times 0.5 + 2 \times 0.33) \div 600 \approx 95.1\%$ (Total number is 600)								

TABLE 5. The numbers of ICS for Fault 1~Fault 13.

	Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Fault 6	Fault 7	Fault 8	Fault 9	Fault 10	Fault 11	Fault 12	Fault 13
Dataset A	6	4	4	4	4	2	0	2	0	2	2	0	2
Dataset B	0	4	6	0	4	0	2	0	4	6	2	0	4

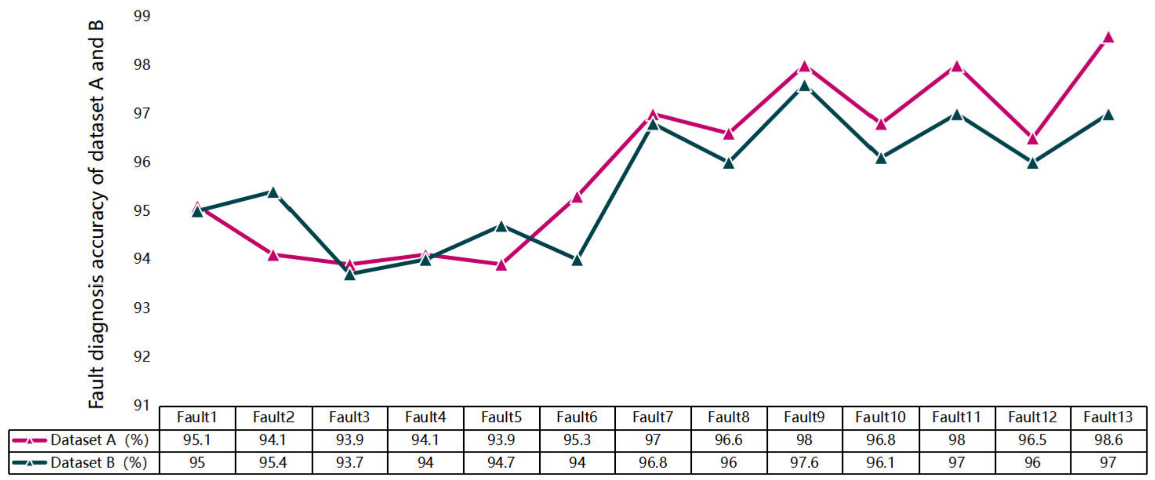
To take Fault 1 of dataset A as a specific example, Table 4 quantitatively lists its fault classifying result in terms of numbers of category 1 that are correctly classified or misclassified (includes 'ICS' case and 'wrong' case).

By a method similar to that adopted for fault diagnosis accuracy calculation in Table 4, the other 12 accuracy estimates on both dataset A and dataset B would be solvable, as illustrated in Fig.14. In addition, Table 5 explicitly lists the numbers of ICS for 13 individual faults.

It's obvious that the numbers of ICS for Fault 7 $\sim$ Fault 12 are slightly less than those for Fault 1 $\sim$ Fault 6. Referring to Fig.14 (for quantitative fault diagnosis accuracy), we deduce the conclusion that the fault features of accelerators, when compared to gyroscopes, exhibit both obvious and stable, making themselves to be easily identified and further classified in the process.

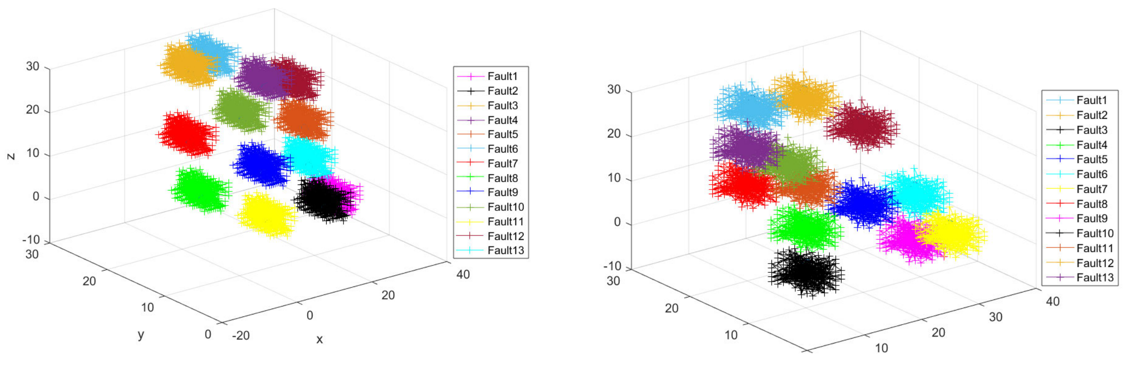
B. THE OPTIMIZED-DBN BASED FAULT CLASSIFICATION EVALUATION

This section is devoted to the fault classification evaluation with the trained DBN models. We use the remaining 10% of total data volume (the data that are not already involved) as validation sets. For dataset A, the definite structural framing of DBN designed would be '6-11-7-21-13' with fixed $\varepsilon = 0.015$ and $\kappa = 0.7$, for dataset B, the so-called DBN anatomy



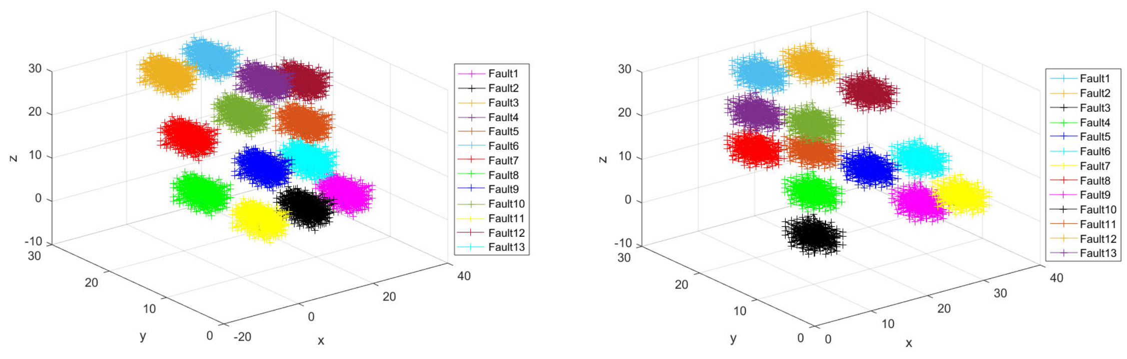
	Fault1	Fault2	Fault3	Fault4	Fault5	Fault6	Fault7	Fault8	Fault9	Fault10	Fault11	Fault12	Fault13
Dataset A (%)	95.1	94.1	93.9	94.1	93.9	95.3	97	96.6	98	96.8	98	96.5	98.6
Dataset B (%)	95	95.4	93.7	94	94.7	94	96.8	96	97.6	96.1	97	96	97

FIGURE 14. **The fault diagnosis accuracy of dataset A and B (with given test sets).**



(a) Fault classification result with original DBN (b) Fault classification result with optimized DBN

FIGURE 15. **The fault classification evaluation (Dataset A).**

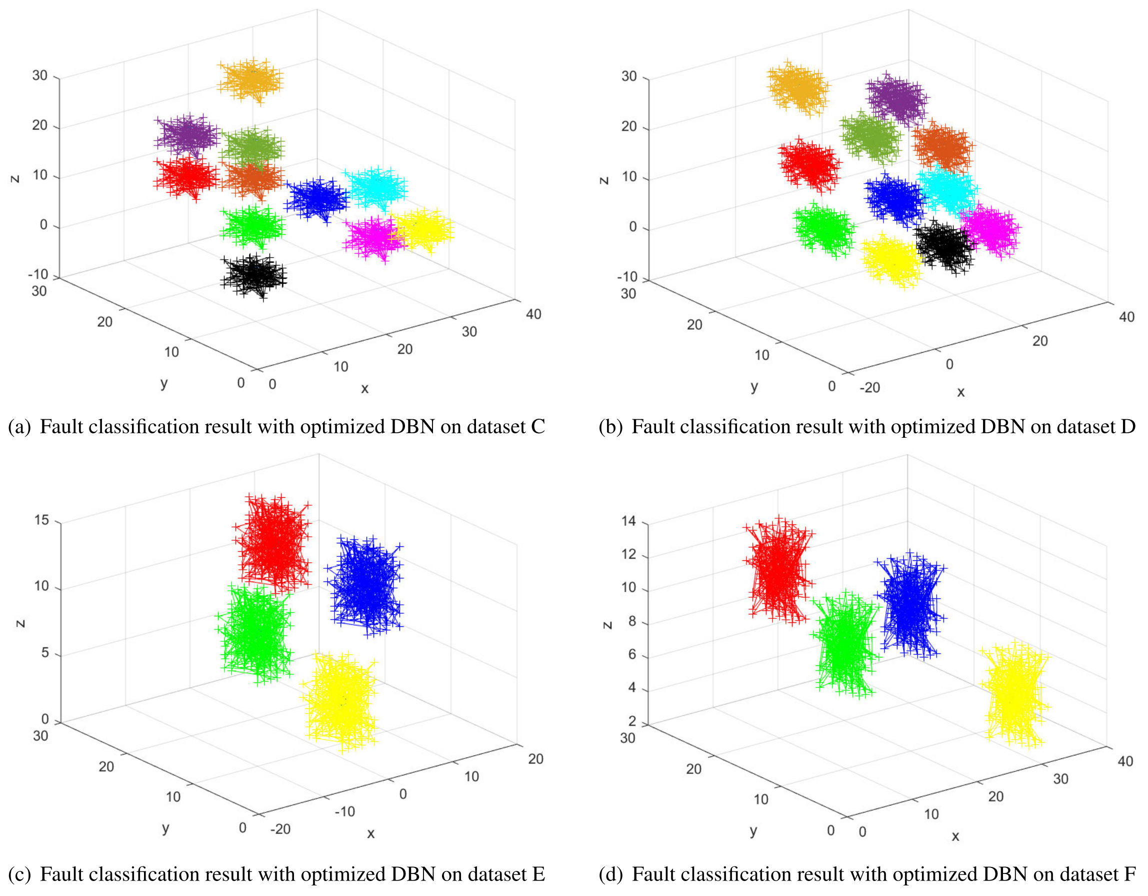


(a) Fault classification result with original DBN (b) Fault classification result with optimized DBN

FIGURE 16. **The fault classification evaluation (Dataset B).**

would be '6-6-7-27-13' with fixed $\varepsilon = 0.01$ and $\kappa = 0.7$. The fault classification results are spatially presented with colored clusters in Fig. 15 and Fig.16.

In Fig.15 (a) and Fig.16 (a), 'original DBN' refers to the DBN in initial structure '6-11-7-21-13' with no inexact LSA-GA based weight value optimization or BA based dynamic number adjustment of hidden-layer neuron. To illustrate, each color represents one single fault. Quite clear, there are more overlapping areas between certain faults in Fig.15 (a) (or Fig.16 (a)) in comparison with those in matched Fig.15 (b) (or Fig.16 (b)), the optimized DBN models are therefore proved to be more precise in differentiating varieties of faults in stead of classifying one single fault into two or three categories.



(a) Fault classification result with optimized DBN on dataset C
(b) Fault classification result with optimized DBN on dataset D
(c) Fault classification result with optimized DBN on dataset E
(d) Fault classification result with optimized DBN on dataset F

FIGURE 17. **Generalization ability tests.**

TABLE 6. **The comparative accurate rate, STD and consumption time results.**

Category	Dataset A			Dataset B		
	Correct rate (%)	STD	Time(s)	Correct rate (%)	STD	Time(s)
The original DBN	92.865	0.6144	367.92	92.97	0.6613	389.67
The optimized DBN	95.9923	0.2758	341.5	95.6385	0.2235	359.38

Under same conditions, in terms of accurate rate, standard deviation (STD) and consumption time, a comparative result between the original DBN and optimized DBN is also quantitatively given in Table 6. The time consumed is recorded by a timer in platform ‘Matlab 2019b’ (Intel(R) Core(TM) i5-8265U, 8GB RAM). It is fair to say the optimized DBN leads to better classifying performances without any load in computational efficiency.

The following Fig.17 attempts to testify the learning performances of proposed DBN models in terms of generalization ability. By a method similar to that adopted for the data collection in Section IV part *B*, the dataset C, D, E and F with similar data features may be created and defined by fault sets. Dataset C and D respectively correspond to uniform linear motion and uniform circular motion, whose 10 types of faults (with $c = 3.5$ *deg* and $\zeta = 10.5$ for deviation and drift representations) are randomly selected from 13 types of faults described in Table 2. Analogously, dataset E and F respectively correspond to uniform linear motion and uniform circular motion, and they both consist of 4 types of faults (randomly chosen from Table 2) with $c = 4.5$ *deg* and $\zeta = 11.5$.

It is obvious to indicate that, the optimized DBN models identically lead to ideal classification results on dataset C, D, E and F. There is no need to conduct data reconstruction or feature extraction in the process, their abilities in learning useful data features and deep level data association among complex data structures could be guaranteed. With the wheeled robot applied in practice, the dynamic DBNs may provide the reference models for the design and development of large scale prior data based fault diagnosis.

VI. SUMMARY

A dynamic DBN model design with inexact LSA-GA based weight value optimization for fault diagnosis was employed. With algorithmically constructing the modules in which the numbers of hidden-layer neuron of 3 constituent RBMs dynamically changes with respect to the given datasets, which was experimentally assessed with a MPU6050 module and further applied to practical wheeled robot fault diagnosis. It demonstrated that, corresponding to the uniform motions of robot bodies, the established datasets function as training, test and validation sets appear to have their applicabilities to classification problems, and the simulated deviation and drift faults of IMUs can be used for deep level data association extraction between data and fault categories. The numerical simulation results indicate that the designed DBN models lead to better fault diagnosis accuracy and generalization ability with optimized momentum and learning rate, and were validated via spatially represented fault classification results. We conclude that the optimized DBN design is generally to be preferred to unadjustable DBN anatomies since fewer categories are simultaneously misclassified for numbers of single faults. Heavy emphases would need to be placed upon the analyses of robustness and reliability for this large scale data based fault diagnosis method, also, if it were desired to enhance the efficiency, the problem of computational load may not be neglected. The following investigation will focus on these essential issues.

REFERENCES

[1] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014, doi: 10.1109/ACCESS.2014.2302442.

[2] X. Zhao, S. Wang, J. Zhang, Z. Fan, and H. Min, "Real-time fault detection method based on belief rule base for aircraft navigation system," *Chin. J. Aeronaut.*, vol. 26, no. 3, pp. 717–729, Jun. 2013, doi: 10.1016/j.cja.2013.04.039.

[3] J. Yu, "A particle filter driven dynamic Gaussian mixture model approach for complex process monitoring and fault diagnosis," *J. Process Control*, vol. 22, no. 4, pp. 778–788, Apr. 2012, doi: 10.1016/j.jprocont.2012.02.012.

[4] L. Liu and J. Fu, "Improved state-X^2 fault detection of navigation systems based on neural network," in *Proc. IEEE. Control Decis. Conf.* Xuzhou, China, May 2010, pp. 3932–3937.

[5] C. Zhang, X. Lu, and S. Gao, "Residual chi-square test and davar based on fault diagnosis and positioning," *Navigat. Control*, vol. 17, no. 2, pp. 25–31, Apr. 2018.

[6] C. Yang, A. Mohammadi, and Q.-W. Chen, "Multi-sensor fusion with interaction multiple model and chi-square test tolerant filter," *Sensors*, vol. 16, no. 11, pp. 1835–1860, Nov. 2016, doi: 10.3390/s16111835.

[7] B. Khaldi, F. Harrou, F. Cherif, and Y. Sun, "Monitoring a robot swarm using a data-driven fault detection approach," *Robot. Auto. Syst.*, vol. 97, pp. 193–203, Nov. 2017, doi: 10.1016/j.robot.2017.06.002.

[8] Y. Xiao and Y. He, "A novel approach for analog fault diagnosis based on neural networks and improved kernel PCA," *Neurocomputing*, vol. 74, no. 7, pp. 1102–1115, Mar. 2011, doi: 10.1016/j.neucom.2010.12.003.

[9] Q. Liu, S. J. Qin, and T. Chai, "Decentralized fault diagnosis of continuous annealing processes based on multilevel PCA," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 3, pp. 687–698, Jul. 2013, doi: 10.1109/tase.2012.2230628.

[10] A. Benaicha, G. Mourot, K. Benothman, and J. Ragot, "Determination of principal component analysis models for sensor fault detection and isolation," *Int. J. Control, Automat. Syst.*, vol. 11, no. 2, pp. 296–305, Feb. 2017, doi: 10.1007/s12555-012-0142-x.

[11] Y. Chai, S. Tao, W. Mao, K. Zhang, and Z. Zhu, "Online incipient fault diagnosis based on kullback-leibler divergence and recursive principle component analysis," *Can. J. Chem. Eng.*, vol. 96, no. 2, pp. 426–433, Feb. 2018, doi: 10.1002/cjce.22962.

[12] X. Gu, F. Deng, X. Gao, and R. Zhou, "An improved sensor fault diagnosis scheme based on TA-LSSVM and ECOC-SVM," *J. Syst. Sci. Complex.*, vol. 31, no. 2, pp. 372–384, Apr. 2018, doi: 10.1007/s11424-017-6232-3.

[13] S. Zhang, Y. Wang, M. Liu, and Z. Bao, "Data-based line trip fault prediction in power systems using LSTM networks and SVM," *IEEE Access*, vol. 6, pp. 7675–7686, 2018, doi: 10.1109/ACCESS.2017.2785763.

[14] F. Deng, S. Guo, R. Zhou, and J. Chen, "Sensor multifault diagnosis with improved support vector machines," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 1053–1063, Apr. 2017, doi: 10.1109/TASE.2015.2487523.

[15] Z. Chenglin, S. Xuebin, S. Songlin, and J. Ting, "Fault diagnosis of sensor by chaos particle swarm optimization algorithm and support vector machine," *Expert Syst. Appl.*, vol. 38, no. 8, pp. 9908–9912, Aug. 2011, doi: 10.1016/j.eswa.2011.02.043.

[16] M. Yang, B. Huang, B. Jiang, and S. Lin, "Real-time prediction for wind power based on Kalman filter and suport vector mahines," *J. Northeast Electr. Power Univ.*, vol. 37, no. 2, pp. 45–51, Apr. 2017, doi: 10.19718/j.issn.1005-2992.2017.02.008.

[17] H. Ying, L. Jiang, X. Li, and L. Lin, "Transient stability assessment in bulk power grid using v-nonparallel support vector machine," *J. Northeast Electr. Power Univ.*, vol. 38, no. 5, pp. 31–40, Oct. 2018.

[18] K. Sun, G. Li, and H. Chen, "A novel efficient SVM-based fault diagnosis method for multi-split air conditioning system's refrigerant charge fault amount," *Appl. Thermal Eng.*, vol. 108, no. 9, pp. 989–998, Sep. 2016, doi: 10.1016/j.applthermaleng.2016.07.109.

[19] T. Cui, X. Zhou, W. Liu, Z. Wang, J. Ma, and J. Zheng, "Gear fault diagnosis based on Hilbert envelope spectrum and SVM," *J. Northeast Electr. Power Univ.*, vol. 37, no. 6, pp. 56–61, Dec. 2017, doi: 10.19718/j.issn.1005-2992.2017.06.010.

[20] M. S. Khireddine, K. Chafaa, N. Slimane, and A. Boutarfa, "Fault diagnosis in robotic manipulators using artificial neural networks and fuzzy logic," in *Proc. World Congr. Comput. Appl. Inf. Syst. (WCCAIS)*, Hammamet, Tunisia, Jan. 2014, pp. 1–6.

[21] X. Wang and B. Xiao, "Research on technologies of self healing control on smart distribution network," *J. Northeast Electr. Power Univ.*, vol. 38, no. 5, pp. 80–84, Oct. 2018.

[22] J. Sun, Y. Chai, C. Su, Z. Zhu, and X. Luo, "BLDC motor speed control system fault diagnosis based on LRGF neural network and adaptive lifting scheme," *Appl. Soft Comput.*, vol. 14, no. 1, pp. 609–622, Jan. 2014, doi: 10.1016/j.asoc.2013.10.010.

[23] P. Witczak, K. Patan, M. Witczak, and M. Mrugalski, "A neural network approach to simultaneous state and actuator fault estimation under unknown input decoupling," *Neurocomputing*, vol. 250, pp. 65–75, Aug. 2017, doi: 10.1016/j.neucom.2016.10.076.

[24] P. Li, X. Liu, and J. Li, "Application Study of self-organizing map network based on immune genetic algorithm," *J. Northeast Electr. Power Univ.*, vol. 38, no. 6, pp. 82–85, Dec. 2018.

[25] S. Liu, Y. Sun, and L. Zhang, "A novel fault diagnosis method based on noise-assisted MEMD and functional neural fuzzy network for rolling element bearings," *IEEE Access*, vol. 6, pp. 27048–27068, 2018, doi: 10.1109/ACCESS.2018.2833851.

[26] P. Cao, S. Zhang, and J. Tang, "Preprocessing-free gear fault diagnosis using small datasets with deep convolutional neural network-based transfer learning," *IEEE Access*, vol. 6, pp. 26241–26253, 2018, doi: 10.1109/ACCESS.2018.2837621.

[27] F. Jia, Y. Lei, L. Guo, and S. Xing, "A neural network constructed by deep learning technique and its application to intelligent fault diagnosis of machines," *Neurocomputing*, vol. 272, no. 10, pp. 619–628, Jan. 2018, doi: 10.1016/j.neucom.2017.07.032.

[28] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang, "A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals," *Sensors*, vol. 17, no. 2, pp. 425–435, Feb. 2017, doi: 10.3390/s17020425.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[30] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/j.neunet.2014.09.003.

[31] J. Yin and W. Zhao, "Fault diagnosis network design for vehicle on-board equipments of high-speed railway: A deep learning approach," *Eng. Appl. Artif. Intell.*, vol. 56, pp. 250–259, Nov. 2016, doi: 10.1016/j.engappai.2016.10.002.

[32] S. Kamada, T. Ichimura, and T. Harada, "Knowledge extraction of adaptive structural learning of deep belief network for medical examination data," *Int. J. Semantic Comput.*, vol. 13, no. 01, pp. 67–86, Mar. 2019, doi: 10.1142/S1793351X1940004X.

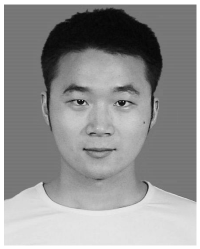
[33] A. Rajendra Kurup, M. Ajith, and M. Martínez Ramón, "Semi-supervised facial expression recognition using reduced spatial features and deep belief networks," *Neurocomputing*, vol. 367, pp. 188–197, Nov. 2019, doi: 10.1016/j.neucom.2019.08.029.

[34] K. Adem, S. Kiliçarslan, and O. Cömert, "Classification and diagnosis of cervical cancer with stacked autoencoder and softmax classification," *Expert Syst. Appl.*, vol. 115, pp. 557–564, Jan. 2019, doi: 10.1016/j.eswa.2018.08.050.

[35] M. R. Ramli, Z. A. Abas, M. I. Desa, Z. Z. Abidin, and M. B. Alazzam, "Enhanced convergence of bat algorithm based on dimensional and inertia weight factor," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 31, no. 4, pp. 452–458, Oct. 2019, doi: 10.1016/j.jksuci.2018.03.010.

LINLIN XIA received the B.S. degree in automation and the M.S. and Ph.D. degrees in navigation, guidance, and control from Harbin Engineering University, Harbin, China, in 2003, 2006, and 2008, respectively. She has served for her bachelor's and master's degrees with the Education Development Centre, Education Ministry of China, as a Communication Review Expert. She is currently a Professor with the School of Automation Engineering, Northeast Electric Power University, Jilin, China. Her research interests include multisource information processing for integrated navigation system, navigation and positioning of robots, artificial intelligence and its applications, and advanced control.

SHUFENG ZHANG received the B.S. degree in electrical engineering and automation electrical from Qufu Normal University, Shangdong, China, in 2016. She is currently pursuing the master's degree with the School of Automation Engineering, Northeast Electric Power University, Jilin, China. Her research interest includes application of deep learning in navigation system fault diagnosis.



RAN SHEN received the B.S. degree in automation from the Yancheng Institute of Technology, Yancheng, China, in 2018. He is currently pursuing the master's degree with the School of Automation Engineering, Northeast Electric Power University, Jilin, China. His research interests include visual inertial odometer of SLAM, state estimation, and multisensor fusion.

JIASHUO CUI received the B.S. degree in electronics and information engineering from Tiangong University, Tianjin, China, in 2017. He is currently pursuing the master's degree with the School of Automation Engineering, Northeast Electric Power University, Jilin, China. His research interests include Semantic SLAM, state estimation, and deep learning.