Applied Soft Computing Journal 168 (2025) 112493

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

An evolutionary deep learning approach using flexible variable-length dynamic stochastic search for anomaly detection of robot joints

Qi Liu [a,b], Yongchao Yu [a,b], Boon Siew Han [a,b], Wei Zhou [a,b,*]

[a] Schaeffler Hub for Advanced Research at Nanyang Technological University, 61 Nanyang Dr, ABN-B1b-11, Singapore 637460, Singapore
[b] School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore

HIGHLIGHTS

- A flexible variable-length dynamic stochastic search is designed.
- An evolutionary deep learning approach for anomaly detection is proposed.
- An anomaly detection method based on evolutionary deep learning is developed for robot joints.
- Vibrations of robot joints are monitored by fiber Bragg grating sensors.
- Prove the feasibility and satisfactory performance of the proposed method.

ARTICLE INFO

Keywords:
Anomaly detection
Deep learning
Fiber Bragg grating
Condition monitoring
Metaheuristic algorithm

ABSTRACT

Anomaly detection is crucial for condition monitoring of robot joints. An increasing number of anomaly detection methods based on deep learning have been investigated. However, since the deep learning architectures for anomaly detection are manually designed by trial and error, the design process is time-consuming, and the designed deep learning architectures may not be optimal for specific anomaly detection tasks. In this paper, a Flexible Variable-Length Dynamic Stochastic Search (FVLDSS) is proposed by designing and embedding the encoding and alignment strategies into the original Dynamic Stochastic Search (DSS). Subsequently, an Evolutionary Deep Learning Approach Using FVLDSS (EDLDSS) is proposed to automatically search for an optimal deep learning architecture for anomaly detection. In EDLDSS, Convolutional Neural Network (CNN) is used as the feature extractor, *k*-nearest neighbors trained only with normal samples is used as the anomaly detector, and FVLDSS is used to simultaneously optimize the network structure and hyperparameters of CNN. Furthermore, an anomaly detection method based on EDLDSS is developed to detect the anomalies in robot joints, in which S-transform spectrograms of vibration signals normalized by Z-score normalization are used as the input of CNN. To validate the performance of EDLDSS, a condition monitoring system using fiber Bragg grating sensors is first established for the industrial robot to acquire vibration signals from robot joints and conduct three experiments. The statistical results demonstrate the feasibility and satisfactory performance of EDLDSS in handling the anomaly detection problem of robot joints.

1. Introduction

Industrial robots have greatly promoted the development of intelligent manufacturing by efficiently and precisely performing various complex tasks such as welding and assembly [1]. Robot joints, as the key components of industrial robots, are prone to failures due to adverse working conditions, overloads, and accidents, which leads to huge economic loss, long downtime, and even casualty [2]. Thus, condition monitoring of robot joints is a vital predictive maintenance strategy for industrial robots in Industry 4.0, which is conducive to early preventing failures of industrial robots [3]. Fault diagnosis plays an important role in condition monitoring of robot joints [4–6], in which a large number of normal and abnormal samples need to be collected for fault diagnosis methods based on deep learning. However, with the advancement of

* Corresponding author at: Schaeffler Hub for Advanced Research at Nanyang Technological University, 61 Nanyang Dr, ABN-B1b-11, Singapore 637460, Singapore.
E-mail addresses: mwzhou@ntu.edu.sg, wzhou@cantab.net (W. Zhou).

https://doi.org/10.1016/j.asoc.2024.112493
Received 1 April 2024; Received in revised form 30 October 2024; Accepted 10 November 2024
Available online 27 November 2024

technology, the reliability and robustness of industrial robots have been substantially enhanced, which means that the failures of robot joints are quite rare and it is difficult to acquire abnormal samples. Replacing the normal component with a faulty component is commonly used in fault diagnosis of robot joints [7], whereas this method cannot be applied to fault diagnosis of highly integrated robot joints because their components are difficult to disassemble. Different from fault diagnosis of robot joints, anomaly detection of robot joints, which is a one-class classification problem, can be realized without prior knowledge of abnormal samples. For this reason, anomaly detection is a promising direction for condition monitoring of robot joints and has gained increasing attention in recent years.

Numerous research efforts have been performed on anomaly detection methods based on traditional machine learning and statistical models. These anomaly detection methods rely heavily on handcrafted feature engineering and domain expertise and are only applied to low-dimensional problems with a small amount of data, which limits their generalizability and applications [8]. To overcome their limitations, anomaly detection methods based on deep learning have been prosperously investigated. Deep learning, as a subfield of machine learning, draws inspiration from the brain's structure and function and adopts multilayer neural networks to imitate the brain's complex decision-making capability [9]. Unlike the traditional machine learning and statistical models, deep learning, which has the powerful ability to automatically extract discrimination features from large-scale complex data, provides great convenience and superiority for anomaly detection. There are various deep learning models for anomaly detection, such as Autoencoder (AE) [10], Variational Autoencoder (VAE) [11], Convolutional Autoencoder (CAE) [12], Long Short-Term Memory (LSTM) [13], and Generative Adversarial Network (GAN) [14].

To handle the anomaly detection problem of robot joints, a small number of anomaly detection methods based on deep learning have been investigated. An unsupervised anomaly detection method based on Convolutional Neural Network (CNN) and VAE, which can depend only on learning the normal pattern from normal time series data, was presented by Chen et al. [15]. To detect the anomalies in robot joints, Zhong et al. designed a 1D CAE by integrating CNN and AE and adopted a sliding window algorithm for data augmentation [16]. Yun et al. provided an anomaly detection method in which time-frequency features were extracted from the Short-Time Fourier Transform (STFT) spectrogram of the sound acquired by stethoscopes as the input of the AE model [17]. For anomaly detection of robot joints, a residual shrinkage transformer relation network, in which a residual shrinkage network was used to extract representative features from vibration signals, was presented by Chen et al. [18]. The aforementioned anomaly detection methods cannot identify the failure patterns of the robot joint, whereas they can effectively judge whether the robot joint is abnormal by using only normal samples. This phenomenon indicates that the aforementioned anomaly detection methods have a strong ability to overcome the challenge of insufficient abnormal samples. However, the deep learning architectures in the aforementioned anomaly detection methods were manually designed by the trial-and-error method in terms of various anomaly detection tasks, which means that the design process is time-consuming and the designed deep learning architectures may not be optimal. Thus, it is necessary to research how to automatically search for optimal deep learning architectures for anomaly detection of robot joints.

With the advancement of artificial intelligence, metaheuristic algorithms, which aim at finding a global optimum in the search space through exploration and exploitation processes, have shown great promise in solving various optimization problems, such as simulation optimization [19–21], structural optimization [22–24], and process parameter optimization [25–27]. In particular, metaheuristic algorithms have been successfully used for the automatic design of deep learning architectures for classification. The main problems of current research are that there is no evolving deep learning architecture by metaheuristic algorithms for anomaly detection, and the automatic design process based on metaheuristic algorithms usually has a high computational cost. In addition, the no-free-lunch theorems [28] state that there is no universal metaheuristic algorithm capable of effectively solving all optimization problems, which implies that a metaheuristic algorithm may excel in solving one set of optimization problems but perform poorly on another set of optimization problems. When evolving deep learning architectures by metaheuristic algorithms, three requirements should be considered, including the avoidance of introducing additional hyperparameters other than those in Deep Neural Network (DNN), simple implementation, and fast computational speed. Moreover, since the optimal depth of DNN is usually unknown for various problems, fix-length metaheuristic algorithms cannot guarantee the diversity of the depth of the deep learning architecture. Thus, it is necessary to further investigate a simple and flexible variable-length metaheuristic algorithm with fewer control parameters to efficiently evolve deep learning architectures for anomaly detection.

On the other hand, the high-quality signal is essential to ensuring the anomaly detection accuracy for robot joints. In condition monitoring systems, the vibration signal is one of the typical signals reflecting fault characteristics [29], and it can be acquired by various types of sensors, such as fiber optic sensors [30], piezoelectric accelerometers [31], and Microelectromechanical Systems (MEMS) [32]. Among various available sensors, Fiber Bragg Grating (FBG) is a desired sensor for condition monitoring of robot joints. This is because the FBG sensor is lightweight, small, flexible, embeddable, and easy to form a sensing network. Different from electrical sensors, the FBG sensor has the advantage of anti-electromagnetic interference, which indicates that electromagnetic interference has no effect on the signal acquired by the FBG sensor [33]. In particular, multiple FBG sensors can be multiplexed on a single optical fiber, which dramatically solves the problem of complex wiring and difficult integration. It is worth noting that there is no research literature on the application of the FBG sensor to condition monitoring of robot joints. In other words, it is of great significance to apply the FBG sensor to vibration-based anomaly detection of robot joints.

The aim of this study is to propose an Evolutionary Deep Learning Approach Using Flexible Variable-Length Dynamic Stochastic Search (EDLDSS) for anomaly detection of robot joints. The main contributions of this study are given as follows:

- Aiming at monitoring the vibrations of robot joints, a condition monitoring system using FBG sensors is first established for the industrial robot by embedding six FBG sensors into the inner space of six robot joints. The use of the six FBG sensors can not only acquire high-quality vibration signals from six robot joints, but also will not increase the weight of the industrial robot and affect the movements of the industrial robot.
- Considering the variable-length characteristic of CNN, a Flexible Variable-Length Dynamic Stochastic Search (FVLDSS) is proposed by designing and embedding the encoding and alignment strategies into the original Dynamic Stochastic Search (DSS). FVLDSS has the characteristics of low computational complexity, simple implementation, and no control parameters other than the population size and the maximum number of iterations.
- To automatically search for an optimal CNN architecture for anomaly detection, EDLDSS is designed by combining CNN, *k*-Nearest Neighbors (KNN), and FVLDSS, in which CNN is used as the feature extractor, KNN trained only with normal samples is used as the anomaly detector, and the network structure and hyperparameters of CNN are simultaneously optimized by FVLDSS. This is the first time deep learning architectures for anomaly detection are evolved by the metaheuristic algorithm. In particular, EDLDSS can efficiently obtain the optimal CNN architecture with strong feature extraction capability.
- An anomaly detection method based on EDLDSS is first developed to detect the anomalies in robot joints, in which Z-score normalization

is used to normalize vibration signals and S-transform is used to transform 1D vibration signals into 2D matrices as the input of CNN. The proposed anomaly detection method is conducive to increasing the anomaly detection accuracy for robot joints because six optimal CNN architectures for anomaly detection of six robot joints are automatically obtained by EDLDSS. Moreover, since no expert knowledge is required, the difficulty and time cost of designing CNN architectures for anomaly detection of robot joints are reduced.

The organization of this paper is given as follows. The research literature on the automatic design of deep learning architectures is overviewed in Section 2. The relative theories of FBG, DSS, CNN, and KNN are presented in Section 3. Section 4 introduces the condition monitoring system of the industrial robot. The details of the EDLDSS and anomaly detection method are provided in Section 5. Section 6 analyzes the performance of EDLDSS in anomaly detection of robot joints. Finally, the conclusions and future work are given in Section 7.

2. Literature review

During the last decade, a large amount of research literature has been published on how to automatically search for optimal deep learning architectures. The automatic design of deep learning architectures, which can be considered a reinforcement learning process as well as an optimization problem, mainly consists of two objectives: network structure and hyperparameters.

The network structure significantly affects the performance of DNN. The more difficult and complex the real-world problem, the deeper the model with denser topology among layers should be. To effectively obtain appropriate and robust network structures, many network structure optimization methods have been investigated. Since most of these optimization methods are used for DNN, they are usually considered Neural Architecture Search (NAS) methods [34]. The NAS methods based on various search strategies, which automatically traverse all network structures in the search space and select a network structure with the best performance, are classified into two categories: evolutionary algorithm-based and reinforcement learning-based methods [35]. The execution of the evolutionary algorithm-based methods follows the heuristic search process of evolutionary algorithms such as Genetic Algorithm (GA) [36] and Genetic Programming (GP) [37], and the search space and evaluation in the reinforcement learning-based methods correspond to the action space and reward in reinforcement learning [38]. Moreover, considering that a large number of computational resources are required when using the NAS methods, Junior and Yen [39] and Wang et al. [40] employed Particle Swarm Optimization (PSO) and Monarch Butterfly Optimization (MBO) to automatically search for the meaningful network structure of CNN for image classification, respectively.

Besides the network structure, hyperparameters, such as the kernel size in CNN and the number of hidden layers in AE, also significantly affect the performance of DNN. The number of hyperparameters increases with the depth of DNN, and tuning hyperparameters can improve the performance of DNN. There are two common hyperparameter optimization methods: grid search and random search [41]. The grid search method can obtain reproducible optimization results, whereas it cannot be effectively used to search the high-dimensional parameter space [42]. The random search method can avoid under-sampling of important dimensions and is more efficient than the grid search method, whereas it is not adaptive [43]. Metaheuristic algorithms, as the powerful and effective optimization tool, have been widely employed to optimize the hyperparameters of traditional machine learning models. For example, Kuo and Chiu [44] adopted a hybridization of jellyfish and PSO to optimize the penalty and kernel parameters of Support Vector Machine (SVM), Li et al. [45] employed Whale Optimization Algorithm (WOA) to tune the weight and threshold of Back-Propagation Neural Network (BPNN), and Wang et al. [46] utilized Chaos Game Optimization (CGO) to optimize the weight of Extreme Learning Machine (ELM). Furthermore, metaheuristic algorithms have also been successfully applied to hyperparameter optimization of DNN [47–49]. Compared with the grid search and random search methods, metaheuristic algorithms can provide promising solutions with better performance due to their great exploration and exploitation capabilities.

The network structure optimization methods focus on automatically searching for an optimal network structure and ignore the effect of hyperparameters, whereas the hyperparameter optimization methods are the complete opposite. The main disadvantage of the two types of methods is that only a single influence factor is considered in each method. To simultaneously optimize the network structure and hyperparameters of DNN, an increasing number of evolving deep learning architectures by metaheuristic algorithms have been developed. Kanwal et al. [50] presented a multi-objective PSO to design flexible CAEs by evolving the arrangement of layers and the number of hyperparameters. Shi et al. [51], Hassanzadeh et al. [52], and Martín et al. [53] applied GA to the evolutionary architecture search for Graph Neural Network (GNN), CNN, and GAN, respectively. To simplify the hyperparameter tuning process of CNN, Chen et al. [54] evolved CNN architectures by Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D). For the compact neural architecture search, Huang et al. [55] designed a two-level variable-length PSO for the evolution of the micro-architecture and macro-architecture of CNN. Ang et al. [56] and Karthiga et al. [57] applied Teaching-Learning-Based Optimization (TLBO) and Grey Wolf Optimizer (GWO) hybridized with Artificial Bee Colony (ABC) algorithm to the optimization of the CNN architecture, respectively. As discussed above, current researchers mainly focus on developing various evolving deep learning architectures by metaheuristic algorithms for classification rather than anomaly detection.

To the best of our knowledge, there is no research literature on evolving deep learning architectures by metaheuristic algorithms for anomaly detection. The successful application of evolving deep learning architectures by metaheuristic algorithms to classification problems can provide the opportunity and guidance for the development of evolving deep learning architectures by metaheuristic algorithms for anomaly detection. However, since evolving deep learning architectures by metaheuristic algorithms for classification is computationally expensive due to the high computational complexity of metaheuristic algorithms, how to improve the efficiency of evolving deep learning architectures by metaheuristic algorithms for anomaly detection is a challenging and important task. For this purpose, this study proposes an evolutionary deep learning approach using flexible variable-length dynamic stochastic search to realize the efficient and automatic design of deep learning architectures for anomaly detection, especially anomaly detection of robot joints.

3. Relative theory

3.1. Fiber Bragg grating

An FBG is an intrinsic fiber optic sensor that can reflect particular wavelengths of incident light and transmit other wavelengths [58]. In terms of the coupled-mode theory, the reflected wavelength called the Bragg wavelength λ_B is expressed as follows:

$$\lambda_B = 2n_{neff}\Lambda \tag{1}$$

where n_{neff} denotes the effective refractive index of the Bragg grating, and Λ denotes the period of the Bragg grating.

The changes of strain ε and temperature ΔT affect the periodical variation of the effective refractive index simultaneously, resulting in the shift of the Bragg wavelength $\Delta\lambda_B$. The formula for describing the shift of the Bragg wavelength is given as follows [59]:

$$\Delta\lambda_B = \lambda_B[(1-P_e)\varepsilon + (\alpha+\xi)\Delta T] \tag{2}$$

where P_e denotes the strain-optic coefficient, α denotes the thermal expansion coefficient, and ξ denotes the thermal-optic coefficient.

3.2. Dynamic stochastic search

DSS was proposed by designing a search process to carry out exploration and exploitation processes [60]. DSS, as a population-based metaheuristic algorithm, provides the powerful ability to address continuous optimization problems. Algorithm 1 provides the pseudo-code of DSS for a continuous maximization optimization problem.

3.2.1. Population initialization

The initial population is randomly produced in the d-dimensional search space, including NP search agents (solutions). Each component of the search agent $X_i = (x_{i1}, x_{i2}, \cdots, x_{id})$ $(i = 1, 2, \cdots, NP)$ is calculated as follows:

$$x_{ij} = Lb_j + rand\cdot(Ub_j - Lb_j) \tag{3}$$

where $j = 1, 2, \cdots, d$. Ub_j and Lb_j denote the upper and lower bounds of the jth component, respectively. $rand \in [0, 1]$ denotes a random number.

Algorithm 1. Dynamic stochastic search

1. Define the fitness function $f(X)$, $X = (x_1, x_2, \cdots, x_d)^T$
2. Initialize the population size NP and the maximum number of iterations $Iter_{max}$
3. Initialize the population through Eq. (3)
4. Evaluate the fitness $f(X_i)$ of each search agent X_i $(i = 1, 2, \cdots, NP)$
5. ***While*** $(Iter \le Iter_{max})$
6. Update the dynamic search factor λ through Eq. (4)
7. Update the stochastic search control factor η through Eq. (5)
8. ***for*** each search agent X_i in the population
9. Update the Gaussian distribution's mean μ_i through Eq. (7)
10. Update the standard deviation of the Gaussian distribution σ_i through Eq. (8)
11. ***if*** $(rand < \lambda)$
12. Update the current search agent X_i^{Iter-1} through Eq. (9)
13. ***else***
14. Update the current search agent X_i^{Iter-1} through Eq. (6)
15. ***end if***
16. Check if any search agent goes beyond the search space and amend it
17. ***if*** $(f(X_i^{Iter}) > f(X_i^{Iter-1}))$
18. Accept the new search agent
19. ***end if***
20. ***if*** $(f(X_i^{Iter}) > f(X_{best}))$
21. Update the current best search agent X_{best}
22. ***end if***
23. ***end for***
24. $Iter = Iter + 1$
25. ***end while***
26. Output the best search agent and its fitness

3.2.2. Search process

To achieve a balance between exploration and exploitation processes, a balance mechanism is defined, in which a dynamic search factor λ is calculated by Eq. (4) to determine which search process is conducted.

$$\lambda = \exp(1 - Iter_{\max}/Iter) \tag{4}$$

where $Iter_{\max}$ and $Iter$ denote the maximum number of iterations and the number of iterations, respectively.

A stochastic search control factor η is calculated by Eq. (5) to manage the search scope of the search agent.

$$\eta = rand\cdot(1 - Iter/Iter_{\max}) \tag{5}$$

In the exploration process, a new search agent X_i^{Iter} is updated by a Gaussian random number $Gaussian(\mu_i, \sigma_i^2)$ to perturb the current search agent X_i^{Iter-1}, which can be formulated by Eq. (6).

$$X_i^{Iter} = X_i^{Iter-1} + Gaussian(\mu_i, \sigma_i^2)\cdot X_i^{Iter-1} \tag{6}$$

where μ_i denotes the Gaussian distribution's mean and calculated by Eq. (7), and σ_i denotes the standard deviation of the Gaussian distribution and calculated by Eq. (8).

$$\mu_i = \left[fitness(X_{best}) - fitness(X_i^{Iter-1})\right] / fitness(X_i^{Iter-1}) \tag{7}$$

where $fitness(X_{best})$ denotes the fitness of the current best search agent X_{best}, and $fitness(X_i^{Iter-1})$ denotes the fitness of X_i^{Iter-1}.

$$\sigma_i = rand\cdot\eta \tag{8}$$

In the exploitation process, a random number $p_m \in [0, 1]$ is used for the execution of centric shrink and eccentric shrink modes. The centric shrink mode is executed if $p_m < 0.5$, and the eccentric shrink mode is executed if $p_m \ge 0.5$, which can be formulated as follows:

$$\begin{cases} X_i^{Iter} = X_{best} + \eta\cdot\delta\cdot\sqrt{\left|X_{best} - X_i^{Iter-1}\right|} & p_m < 0.5 \\ X_i^{Iter} = X_{best} + \cos(2\pi\eta)\cdot\sqrt{\left|X_{best} - X_i^{Iter-1}\right|} & p_m \ge 0.5 \end{cases} \tag{9}$$

where $\delta \in [-1, 1]$ denotes a random number.

3.3. Convolutional neural network

CNN based on the visual perception of the brain is one of the most

popular DNNs. CNN is a feedforward neural network with convolutional computation, and it mainly consists of a feature extractor and a classifier [61].

The feature extractor is used to automatically extract deep features from the input and consists of four layers:

- **Convolutional layer:** It is the basic operation and core layer of CNN, which generates the feature maps by convolving the input with convolutional kernels across its spatial dimensionality [62]. There are four hyperparameters in the convolutional layer: the number of convolutional kernels, the kernel size, the stride, and the zero-padding.
- **Activation layer:** It is added after the convolutional layer to nonlinearly map the features extracted by the convolutional layer into the high-dimensional nonlinear interval. There are four common activation functions: Tanh [63], Rectified Linear Unit (ReLU) [64], Exponential Linear Unit (ELU) [65], and Gaussian Error Linear Unit (GELU) [66].
- **Batch normalization layer:** It is added between the convolutional layer and the activation layer to normalize the features extracted by the convolutional layer, which is conducive to increasing the training speed of CNN [67].
- **Pooling layer:** It aims to gradually reduce the spatial dimensionality of the input feature map [68]. The average and maximum pooling layers are commonly used in CNN. Moreover, there are three hyperparameters in the pooling layer: the pooling size, the stride, and the zero-padding.

The classifier is used to achieve the classification task and consists of multiple fully-connected layers. The fully-connected layer can be used to flatten the features extracted from several convolutional and pooling layers, and it can also be used as the final classification layer [69].

3.4. k-Nearest neighbors

KNN based on the global distance is a supervised learning method for classification and an unsupervised learning method for anomaly detection [70]. The basic principle of KNN for anomaly detection is that similar observations in the samples are close to each other, whereas dissimilar observations (i.e. anomalies or outliers) in the samples are isolated and far from the cluster of similar observations. KNN operates without presuming the distribution of the data, whereas it can only detect global anomalies. In KNN, the popular distance metrics include Euclidean distance, Manhattan distance, Minkowski distance, and Hamming distance. The Euclidean distance limited to real-valued vectors is the most commonly used and is chosen in this work.

Table 1
Parameters and installation positions of six FBG sensors.

FBG sensor	Grating length	Bragg wavelength	3 dB bandwidth	Reflectivity	Installation position
FBG_1	1.2 mm	1530.99 nm	0.7 nm	33.93 %	Joint_1
FBG_2	1.2 mm	1533.95 nm	0.7 nm	36.90 %	Joint_2
FBG_3	1.2 mm	1537.02 nm	0.7 nm	34.68 %	Joint_3
FBG_4	1.2 mm	1540.16 nm	0.7 nm	34.53 %	Joint_4
FBG_5	1.2 mm	1543.21 nm	0.7 nm	34.68 %	Joint_5
FBG_6	1.2 mm	1546.43 nm	0.7 nm	36.02 %	Joint_6

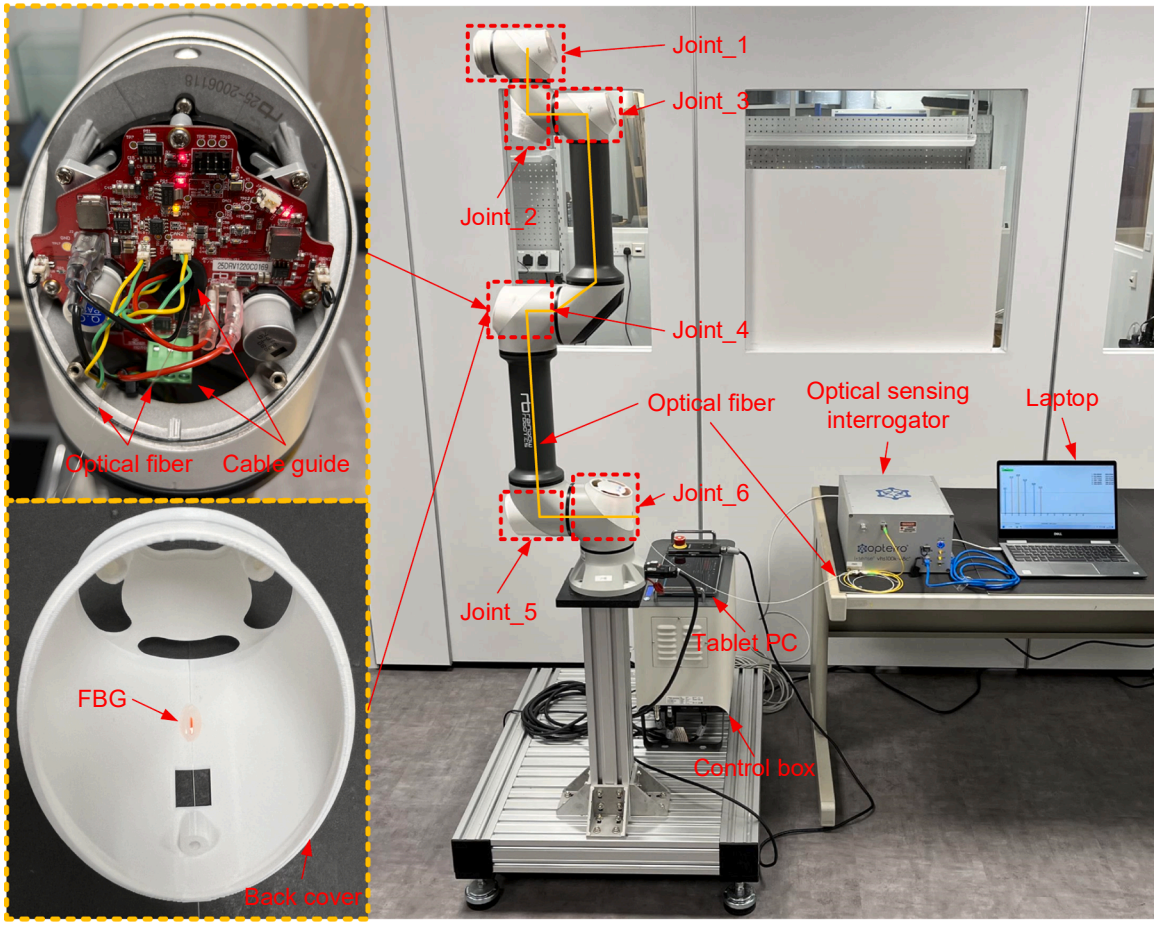

Fig. 1. Condition monitoring system of the RB5–850 robot arm.

4. Experiment setups

A condition monitoring system of an industrial robot is established and displayed in Fig. 1, in which the vibrations of six robot joints are monitored by an FBG sensing system.

4.1. Industrial robot

The industrial robot is the RB5-850 robot arm (Rainbow Robotics Inc.) and has six highly integrated robot joints. To avoid damage to robot joints in the experimental phase, the original plastic back cover of the robot joint is replaced with a polymer back cover fabricated by 3D printing. For the robot arm, its movement is controlled by feeding a user program created in a tablet PC into a control box.

4.2. FBG sensing system

In the FBG sensing system, there are six FBG sensors, an optical sensing interrogator, and a laptop. To acquire the vibration signals, the six FBG sensors are pasted on the internal surface of the back cover of each robot joint with adhesive, and their parameters and installation positions are provided in Table 1. Before beginning the installation, the coating of the optical fiber located at the grating region is removed to improve the strain transmissibility, and the optical fiber is stretched to eliminate the prestress. The optical sensing interrogator is the I*Sense® VHS100K-48C1P-PI (Opterro Inc.), and it has a maximum sampling frequency of 100 kHz and can simultaneously obtain wavelength shifts of the six FBG sensors through an optical channel. The laptop has an Intel(R) Core(TM) i7-12700H CPU @ 2.70 GHz, a 16.00 GB of RAM, and a Windows 11 64-bit operating system. Python 3.9 is installed on this laptop for signal processing and anomaly detection.

5. Research methodology

5.1. Evolutionary deep learning approach using flexible variable-length dynamic stochastic search

Different from traditional feature extraction methods [71–73], CNN has shown great potential in automatically extracting deep features from the input and can be used as the feature extractor. In addition, KNN is simple and effective for anomaly detection and can be utilized to replace the fully-connected layer in CNN for prediction. Moreover, since DSS has

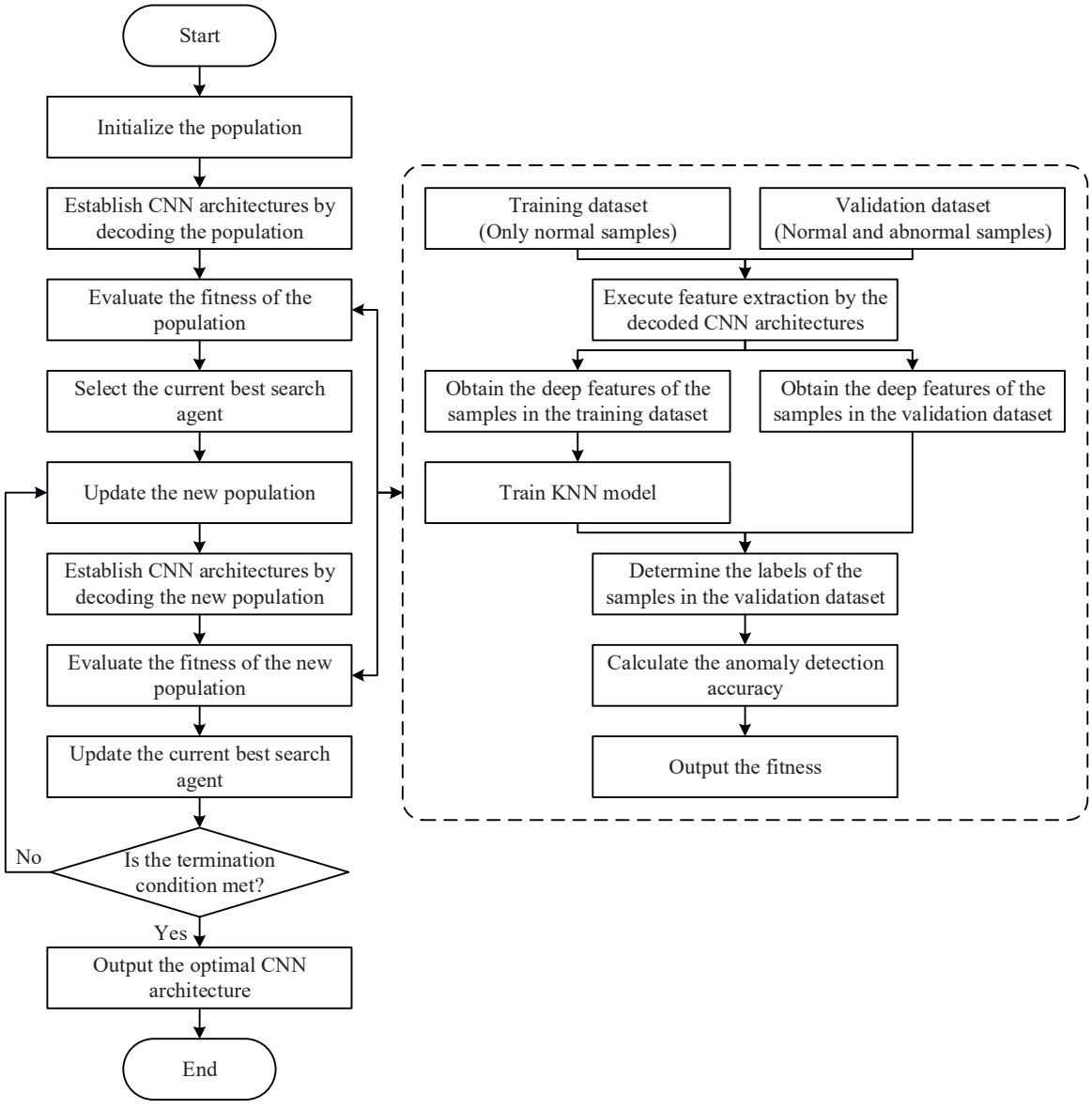

Fig. 2. Flowchart of the evolutionary deep learning approach using flexible variable-length dynamic stochastic search.

Table 2
Hyperparameters of two types of CNN layers and their ranges with an example.

Type of the CNN layer	Hyperparameter	Range	Number of bits	Example
Convolutional layer	Number of convolutional kernels	[1, 128]	7	64 (011 1111)
	Kernel size	[1,8]	3	5 (100)
	Stride	[1,4]	2	1 (00)
	Activation function	[1,4]	2	3 (10)
	Summary		14	011 1111 100 00 10
Pooling layer	Type of the pooling layer	[1,2]	1	2 (1)
	Pooling size	[1,4]	2	4 (11)
	Stride	[1,4]	2	2 (01)
	Summary		5	1 11 01

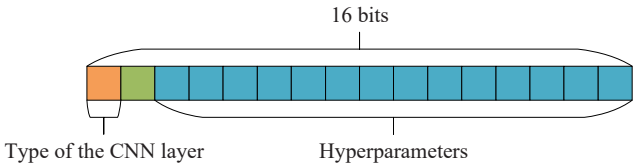

Fig. 3. Binary string and its representation.

a simple structure, simple implementation, and no control parameters other than the population size and the maximum number of iterations, it is suitable for automatically searching for an optimal CNN architecture. To make DSS consistent with the variable-length CNN, a Flexible Variable-Length DSS (FVLDSS) is proposed by introducing the encoding and alignment strategies into the original DSS. Then, an evolutionary deep learning approach for anomaly detection is designed by combining CNN, KNN, and FVLDSS, which is called EDLDSS. The flowchart of EDLDSS is provided in Fig. 2 and illustrated in detail below.

5.1.1. Encoding strategy

In EDLDSS, a search agent represents a complete CNN architecture, and each component of the search agent represents a CNN layer. The CNN architecture includes several convolutional and pooling layers. Meanwhile, the convolutional layer must be followed by a batch normalization layer and an activation layer. Considering the simultaneous optimization of the network structure and hyperparameters, the encoding strategy [74] is modified and used in this work.

The length of the binary string under this encoding strategy should be first determined. The first bit of the binary string represents the type of the CNN layer, which means that "0" represents the convolutional layer and "1" represents the pooling layer. For the convolutional layer, the number of convolutional kernels in the range of 1–128 is represented by a binary string of 7 bits, the kernel size in the range of 1–8 is represented by a binary string of 3 bits, the stride in the range of 1–4 is represented by a binary string of 2 bits, and selecting one of the activation functions (i.e. 1 for "Tanh", 2 for "ReLU", 3 for "ELU", and 4 for "GELU") is represented by a binary string of 2 bits. For the pooling layer, the type of the pooling layer (i.e. 1 for "maximum" and 2 for "average") is represented by a binary string of 1 bit, the pooling size in the range of 1–4 is represented by a binary string of 2 bits, and the stride in the range of 1–4 is represented by a binary string of 2 bits. For the convolutional and pooling layers, the zero-padding is not encoded. If the size of the input feature map is greater than the kernel size or pooling size, the zero-padding is set to "valid". Otherwise, the zero-padding is set to "same". The hyperparameters of the two types of CNN layers and their ranges with an example are given in Table 2. To match the range of the binary string to the range of the hyperparameter, the decimal number, which is obtained by decoding the binary string representing each hyperparameter in terms of the binary rule, is incremented by 1. The ranges of all hyperparameters of the CNN layer in this work are set for anomaly detection of robot joints, and they can be flexibly adjusted in terms of other various real-world applications.

Aiming at improving the ability of FVLDSS to explore the depth of the CNN architecture, an extra bit is added as the second bit of the binary string to increase the encoding space. The binary string represents the convolutional layer or the pooling layer if the extra bit is 0. The exploration of the depth of the CNN architecture is executed if the extra bit is 1, which indicates that the dimension (length) of the search agent in FVLDSS is changed. In addition, to ensure that the binary strings representing the two types of CNN layers have the same length, a placeholder of 9 bits is added to the end of the binary string representing the pooling layer. Thus, the total length of the binary string representing the convolutional layer or the pooling layer is 16 bits. Fig. 3 shows the binary string and its representation, and Fig. 4 shows the representations of binary strings for the convolutional and pooling layers.

According to this encoding strategy, in the search agent, the range of the component corresponding to the convolutional layer is from 0 to 16,383, the range of the component corresponding to the maximum pooling layer is from 32,768 to 40,959, and the range of the component corresponding to the average pooling layer is from 40,960 to 49,151. The components that are not in the three ranges are used to explore the depth of the CNN architecture. Assume the component is denoted as $V_{component}$. In the exploration and exploitation processes of FVLDSS, the CNN layer is removed if $V_{component}$ is smaller than 0, and a new CNN layer is added and $V_{component}$ is adjusted if $V_{component}$ is not smaller than 0 and not in the ranges of 0–16,383 and 32,768–49,151 (i.e. the second bit in

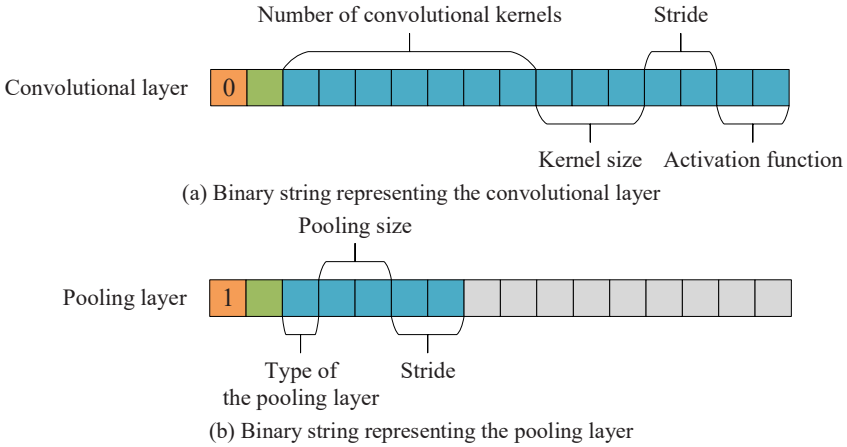

Fig. 4. Representations of binary strings for the convolutional and pooling layers.

the binary string is 1). Table 3 provides the range of $V_{component}$, the type of the CNN layer/operation, and the adjustment rule of $V_{component}$.

As shown in Table 3, we can note that the entire search space is covered in the encoding strategy. Moreover, in addition to the operation for adding the convolutional layer, two different operations for adding the pooling layer are employed. In other words, a component is split into two different components to add two different pooling layers if $V_{component}$ is in the range of 49,152–65,535, and a component is split into two same or different components to add two same or different pooling layers if $V_{component}$ is greater than 65,535. For example, if a component of the search agent updated by FVLDSS is 53,243, the component is split into two components which are 43,005 and 38,914, which means that an average pooling layer and a maximum pooling layer are added.

Fig. 5 depicts an example of the encoding and decoding of a search agent. The variable-length operation of the search agent can be realized by the encoding strategy, which ensures the diversity of the depth of the explored CNN architecture.

Algorithm 2. Population initialization in EDLDSS

1. Initialize the population size NP, the maximum number of layers $MaxNum_{layer}$, the mean number of layers $MeanNum_{layer}$, and the maximum number of pooling layers $MaxNum_{pooling}$
2. ***While*** $(i \le NP)$
3. Randomly generate an integer as the initial number of layers $InitialNum_{layer}$ from $Gaussian(MeanNum_{layer}, 1)$
4. Randomly generate an integer as the number of pooling layers $Num_{pooling}$ from $[0, MaxNum_{pooling}]$
5. Obtain the number of convolutional layers $Num_{convolutional}$ that is equal to $InitialNum_{layer} - Num_{pooling}$
6. Randomly generate the positions of the pooling layers $Position_{pooling}$
7. Determine the positions of the convolutional layers $Position_{convolutional}$
8. ***for*** each component x_{ij} $(j = 1, 2, \cdots, InitialNum_{layer})$
9. ***if*** (x_{ij} corresponds to $Position_{convolutional}$)
10. Randomly generate an integer for x_{ij} from [0, 16383]
11. ***else***
12. Randomly generate an integer for x_{ij} from [32768, 49151]
13. ***end if***
14. ***end for***
15. $i = i + 1$
16. ***end while***
17. Output the initial population

Table 3
Selection of the type of the CNN layer/operation and adjustment rule.

Range of $V_{component}$	Type of the CNN layer/operation	Adjustment rule of $V_{component}$
$(-\infty, -1]$	Remove the CNN layer	-
[0, 16383]	Convolutional layer	$V_{component}$
[16384, 32767]	Add the convolutional layer	$\lfloor V_{component}/2 \rfloor$, $\lceil V_{component}/2 \rceil$
[32768, 40959]	Maximum pooling layer	$V_{component}$
[40960, 49151]	Average pooling layer	$V_{component}$
[49152, 65535]	Add the pooling layer	$40959 + \lceil (V_{component} - 49151)/2 \rceil$, $40960 - \lceil (V_{component} - 49151)/2 \rceil$
[65536, $+\infty$)	Add the pooling layer	$\lfloor V_{component}/2 \rfloor$, $\lceil V_{component}/2 \rceil$

5.1.2. Population initialization

As shown in Fig. 2, the first step of EDLDSS is the population initialization. After determining the population size NP, the maximum number of layers $MaxNum_{layer}$, the mean number of layers $MeanNum_{layer}$, and the maximum number of pooling layers $MaxNum_{pooling}$, NP search agents with different dimensions are randomly generated. The population initialization in EDLDSS is carried out according to the following steps:

Step 1: The initial number of layers $InitialNum_{layer}$, which is the dimension of each search agent, is randomly generated by the Gaussian distribution $Gaussian(MeanNum_{layer}, 1)$ and should be regenerated if it is greater than $MaxNum_{layer}$.

Step 2: The number and positions of the convolutional and pooling layers are randomly generated by using the initial number of layers and the maximum number of pooling layers.

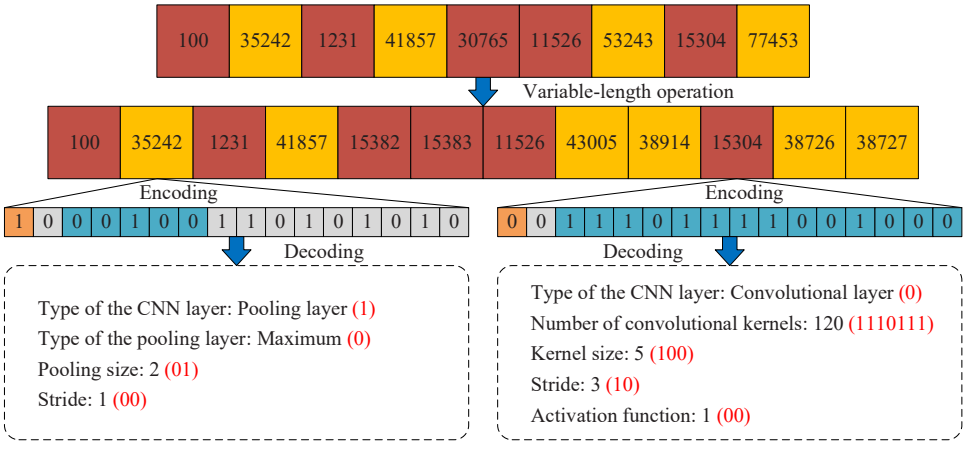

Fig. 5. Example of the encoding and decoding of a search agent.

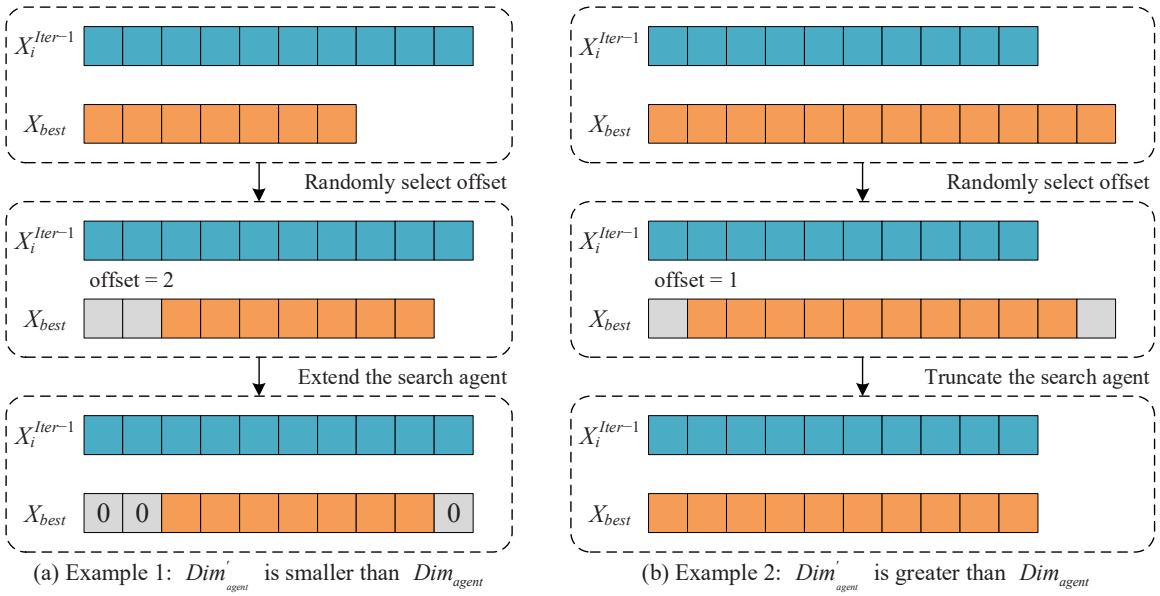

Fig. 6. Two examples of conducting the alignment strategy.

***Step 3*:** The component corresponding to the convolutional layer is randomly generated in the range of 0–16,383, and the component corresponding to the pooling layer is randomly generated in the range of 32,768–49,151.

According to the convention of the CNN construction, each component of the search agent must be an integer, the first layer must be the convolutional layer, and the number of layers must be greater than 2 and not greater than $MaxNum_{layer}$. The pseudocode of the population initialization in EDLDSS is provided in Algorithm 2.

5.1.3. Fitness evaluation

After conducting the exploration and exploitation processes of FVLDSS, the fitness of the new search agent should be evaluated. Each search agent is decoded into a CNN architecture in terms of the encoding strategy. To flatten the deep features extracted by the decoded CNN architecture, a Global Average Pooling (GAP) layer [75], which has no parameter to be tuned and can avoid overfitting at the fully-connected layer, is used to replace the traditional fully-connected layer and added to the tail of the decoded CNN architecture. Then, the deep features of the samples in the training and validation datasets are extracted by the decoded CNN architecture and input into the KNN model.

Accuracy, which indicates the percentage of the number of correctly classified samples to the total number of samples, is a common performance metric and can directly reflect the anomaly detection performance of DNN [76]. The greater the Accuracy, the better the anomaly detection performance of DNN. Thus, Accuracy is used for the fitness evaluation of search agents. After the KNN model is trained by the training dataset with only normal samples, the labels of the normal and abnormal samples in the validation dataset are predicted by the trained KNN model. Then, Accuracy is calculated as the fitness of the search agent. The formula of Accuracy is given as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{10}$$

where TP, TN, FP, and FN denote the true positive, true negative, false positive, and false negative, respectively. TP denotes that the abnormal sample is correctly classified, TN denotes that the normal sample is correctly classified, FP denotes that the normal sample is misclassified, and FN denotes that the abnormal sample is misclassified.

5.1.4. Population update

In the exploration and exploitation processes of FVLDSS, the current best search agent X_{best} must have the same dimension as the current search agent X_i^{Iter-1}. To update search agents with different dimensions, the alignment strategy [74], which is composed of the offset setting and dimension adjustment, is conducted before the exploration and exploitation processes.

For the offset setting, an integral offset is randomly generated in the range of 0 to $\left|Dim_{agent} - Dim'_{agent}\right|$ to increase the diversity of the population, where Dim_{agent} is the dimension of X_i^{Iter-1} and Dim'_{agent} is the dimension of X_{best}. For the dimension adjustment, X_{best} is extended or truncated in terms of X_i^{Iter-1}, which makes X_{best} the same dimension as X_i^{Iter-1}. Two examples of conducting the alignment strategy are shown in Fig. 6. The first example is to conduct the alignment strategy if Dim'_{agent} is smaller than Dim_{agent}, and the second example is to conduct the alignment strategy if Dim'_{agent} is greater than Dim_{agent}. In the first example, the new X_{best} is obtained by extending X_{best}, which means that the (offset + 1)th to (offset + Dim'_{agent})th components of the new X_{best} are the same as all components of X_{best}, and the other components of the new X_{best} are padded with zeros. In the second example, the new X_{best} is obtained by truncating X_{best}, which means that the (offset + 1)th to (offset + Dim_{agent}) th components of X_{best} are extracted.

The new X_{best}, which has the same dimension as X_i^{Iter-1}, can be obtained by the alignment strategy and used to update a new search agent in terms of Eq. (6) and Eq. (9) in the exploration and exploitation processes of FVLDSS. Then, the new search agent is decoded in terms of the encoding strategy to obtain the corresponding CNN architecture. Thus, the CNN architecture including the network structure and hyperparameters can be flexibly evolved by FVLDSS.

5.2. Anomaly detection method based on EDLDSS for robot joints

Since the vibrations of six robot joints are complex and different in the whole working process of the robot arm, the anomaly detection accuracy may be low when applying the same deep learning architecture to anomaly detection of the six robot joints. The primary characteristic of EDLDSS is that the manual design of the deep learning architecture is turned into an automatic process, which means that EDLDSS can be used to automatically search for the optimal deep learning architecture for

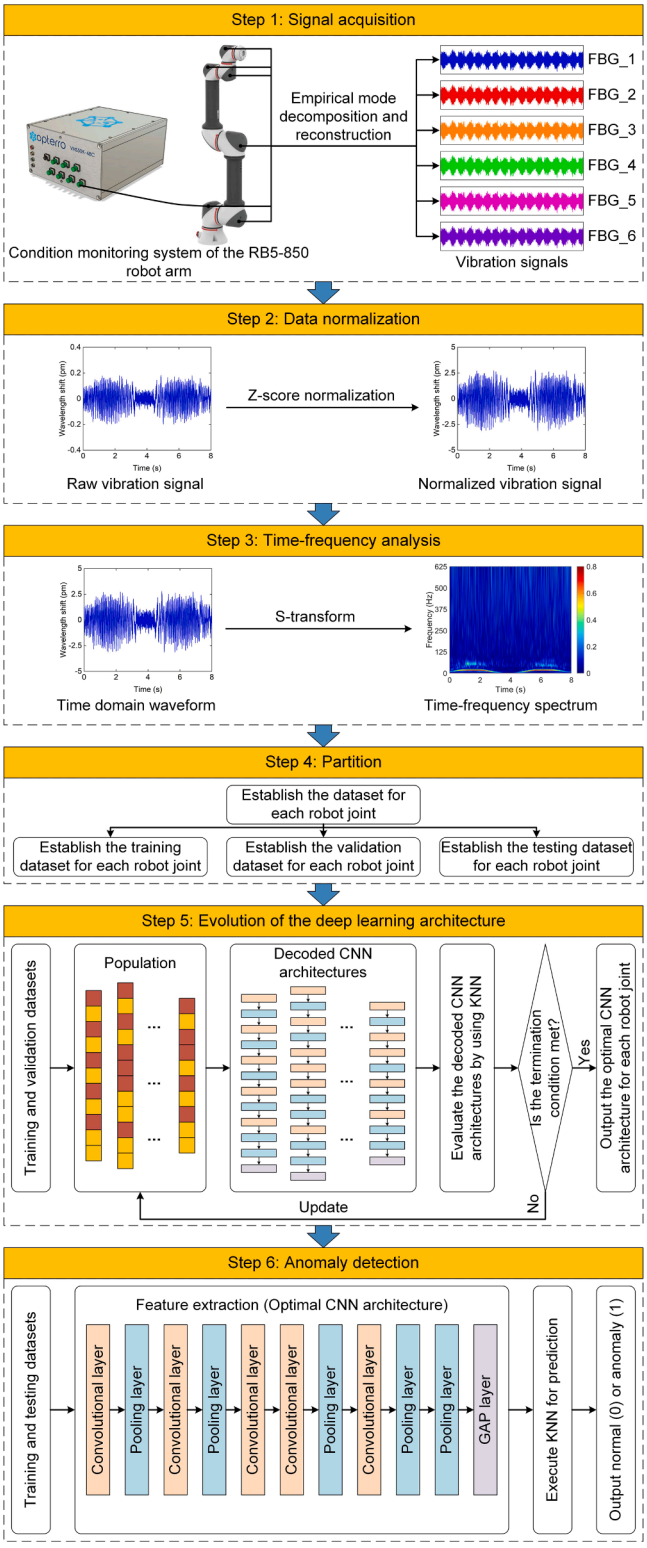

Fig. 7. Flowchart of the anomaly detection method based on EDLDSS for robot joints.

anomaly detection of each robot joint. Thus, to accurately detect the anomalies in robot joints, an anomaly detection method based on EDLDSS is developed. The steps of the proposed anomaly detection method, including signal acquisition, data normalization, time-frequency analysis, partition, evolution of the deep learning architecture, and anomaly detection, are described in Fig. 7 and introduced in detail below.

***Step 1* (signal acquisition):** After setting the sampling frequency of the optical sensing interrogator, the vibration signals from six robot joints are simultaneously acquired by six FBG sensors. Since the robot arm usually works for a long time, the effect of the temperature cannot be ignored. Considering that the frequency of the vibration is significantly higher than that of the temperature, the empirical mode decomposition method [77] is used to eliminate the effect of the temperature. The vibration signal is self-adaptively decomposed into a series of Intrinsic Mode Functions (IMFs) and a residue, and a new vibration signal is reconstructed by all IMFs because the residue is caused by the temperature. In other words, the shift of the Bragg wavelength is induced only by the strain, and Eq. (2) is rewritten as Eq. (11).

$$\Delta\lambda_B = \lambda_B(1 - P_e)\varepsilon \tag{11}$$

***Step 2* (data normalization):** To appropriately parameterize the network and accelerate the gradient descent process, Z-score normalization [78] is adopted to scale the vibration signal with different ranges to a well-proportioned range. The large deviation of the vibration signal can be eliminated by using its mean and standard deviation so that its mean and variance are equal to 0 and 1 respectively, which is formulated by Eq. (12).

$$s_i' = \frac{s_i - \mu_s}{\sigma_s} \tag{12}$$

where s_i denotes the ith data point of the vibration signal, s_i' denotes the normalized value of s_i, μ_s denotes the mean of the vibration signal, and σ_s denotes the standard deviation of the vibration signal.

***Step 3* (time-frequency analysis):** Aiming at making the vibration signal suitable for the format of the input of CNN, S-transform [79], which is one of the most popular time-frequency analysis methods and has a strong time-frequency analysis capability, is performed on the vibration signal to obtain its S-transform spectrogram. In other words, S-transform is adopted to transform 1D vibration signals into 2D matrices as the input of CNN. Meanwhile, the image of the S-transform spectrogram is resized to 256 (width) × 256 (height) to increase the computational speed of CNN.

***Step 4* (partition):** Six datasets are established for six robot joints. The sample in each dataset is composed of the S-transform spectrogram of the vibration signal and the corresponding label. Each dataset is divided into a training dataset, a validation dataset, and a testing dataset. The training dataset only includes normal samples, whereas the validation and testing datasets include normal and abnormal samples.

***Step 5* (evolution of the deep learning architecture):** The training and validation datasets are applied to EDLDSS to obtain the optimal CNN architecture for each robot joint. The population with *NP* search agents is updated and decoded into different CNN architectures, and the KNN model trained by the training dataset is adopted to evaluate the performance of each decoded CNN architecture on the validation dataset. For each robot joint, the population update, CNN construction, and performance evaluation are repeated until the maximum number of iterations is reached, and the CNN architecture with the best feature extraction capability is output.

***Step 6* (anomaly detection):** For each robot joint, the deep features of the samples in the training and testing datasets are extracted by the optimal CNN architecture. Then, the extracted deep features are input into the KNN model for training and prediction. Finally, the prediction results of the samples in the testing dataset can be used to judge whether the robot joint is abnormal, where "0" represents the robot joint is normal and "1" represents the robot joint is abnormal.

6. Experiments and discussions

After establishing the datasets by using the condition monitoring system of the RB5-850 robot arm, the performance of EDLDSS in

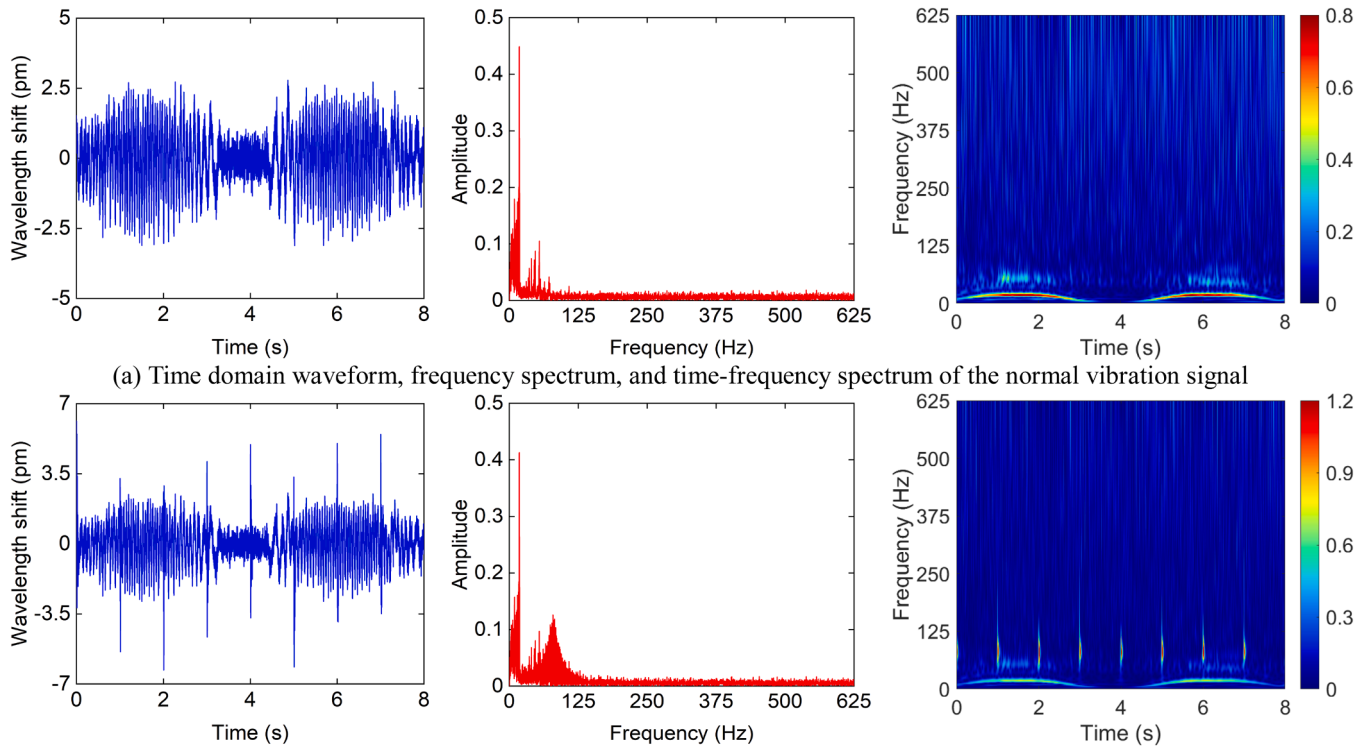

Fig. 8. Time domain waveforms, frequency spectrums, and time-frequency spectrums of the normal and abnormal vibration signals.

handling the anomaly detection problem of robot joints is analyzed by conducting three experiments.

6.1. Establishment of datasets

In this work, a dynamic movement of the RB5-850 robot arm, whose duration is about 8 s, is repeatedly carried out 91 times. First, the robot arm stays in the calibration position. Then, six robot joints are simultaneously rotated 90 degrees counterclockwise. Finally, six robot joints are simultaneously rotated 90 degrees clockwise to return the robot arm to the calibration position. In the whole working process of the robot arm, the sampling frequency is set to 1250 Hz, and the vibration signals from six robot joints are simultaneously acquired by six FBG sensors. The vibration signals acquired at this time are considered normal signals.

Since the normal component in the RB5-850 robot arm with highly integrated joints cannot be replaced with a faulty component, a periodic impulse signal is adopted to simulate the bearing fault, and its formula is given as follows [80]:

$$Fault_{periodicimpulse} = \sum_{i=1}^{T} Ae^{-g\cdot 2\pi f_{nature}t_i}\cdot \sin\left(2\pi f_{nature}t_i\cdot\sqrt{1-g^2}\right) \tag{13}$$

where the amplitude A is set to 0.4. The natural frequency f_{nature} is set to 80 Hz. The damping coefficient g is set to 0.1. The number of impulses T is set to 8. The sampling frequency is set to 1250 Hz, and the sampling time is set to 8 s.

To increase the diversity of samples, a sliding window with a length of 10000 and a stride of 5000 is used to divide each vibration signal into 180 normal vibration signals. Then, 18 randomly selected normal vibration signals are combined with the periodic impulse signal and the periodic impulse is performed on the robot joint to obtain 18 abnormal vibration signals for validation and testing, respectively. Fig. 8 shows time domain waveforms, frequency spectrums, and time-frequency spectrums of the normal and abnormal vibration signals after data normalization. According to Fig. 7, ***Steps 2-3*** are conducted to obtain S-transform spectrograms of all vibration signals. For each robot joint, the training dataset is established by 126 normal samples, and the validation and testing datasets are established by 18 normal samples and 18 abnormal samples.

6.2. Performance metrics

Four performance metrics [81,82], which include Accuracy, Precision, Recall, and F1-score, are used to measure the performance of the anomaly detection method based on EDLDSS. Precision and Recall indicate the sensitivity of the anomaly detection method to the normal sample and the abnormal sample, respectively. F1-score is the harmonic mean of Precision and Recall, and it indicates the reliability of the anomaly detection method. The formulae of Precision, Recall, and F1-score are provided as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{14}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{15}$$

$$\text{F1}-\text{score} = 2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} \tag{16}$$

In addition, the Receiver Operating Characteristic (ROC) curve is commonly used in the field of anomaly detection [83]. ROC is a probability curve plotted with the True Positive Rate (TPR) against the False Positive Rate (FPR) under various thresholds, where TPR and FPR are calculated as follows:

$$\text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{17}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}} \tag{18}$$

To show the average performance of the anomaly detection method, the Area Under the Receiver Operating Characteristic Curve (AUROC) can be calculated by conducting the integral operation for the ROC

Table 4
Comparison of EDLDSS and the comparative algorithms based on six state-of-the-art metaheuristic algorithms for anomaly detection of robot joints.

Robot joint	Metric	EDLTLBO	EDLMFO	EDLMVO	EDLGWO	EDLWOA	EDLSSA	EDLDSS
Joint_1	Accuracy	0.9361	0.9611	**0.9694**	0.9417	0.9583	0.9472	0.9639
	Precision	**0.9947**	0.9736	0.9757	0.9647	0.9695	0.9639	0.9747
	Recall	0.8778	0.9500	**0.9667**	0.9222	0.9500	0.9333	0.9556
	F1-score	0.9192	0.9592	**0.9697**	0.9387	0.9507	0.9429	0.9623
	AUROC	**0.9966**	0.9884	0.9938	0.9895	0.9877	0.9869	0.9963
	Runtime (s)	1055.0838	489.9226	483.6135	383.1407	396.6462	432.5775	**353.0079**
Joint_2	Accuracy	0.9472	0.9694	0.9639	0.9500	0.9528	0.9806	**0.9861**
	Precision	0.9789	0.9737	**0.9947**	0.9895	0.9770	0.9737	0.9842
	Recall	0.9167	0.9667	0.9333	0.9111	0.9222	**0.9889**	**0.9889**
	F1-score	0.9438	0.9677	0.9473	0.9356	0.9393	0.9811	**0.9863**
	AUROC	0.9870	0.9877	0.9963	**0.9997**	0.9867	0.9883	0.9941
	Runtime (s)	1236.7967	599.8894	590.3457	420.3964	487.5466	513.8568	**401.7256**
Joint_3	Accuracy	0.9083	0.9222	0.9028	0.9111	0.9167	0.9139	**0.9278**
	Precision	0.9765	0.9823	0.9596	0.9494	**0.9933**	0.9699	0.9832
	Recall	0.8389	0.8611	0.8444	**0.8722**	0.8389	0.8556	**0.8722**
	F1-score	0.8842	0.8973	0.8807	0.8985	0.8971	0.8959	**0.9181**
	AUROC	0.9802	0.9873	0.9644	0.9727	0.9886	0.9881	**0.9926**
	Runtime (s)	1290.6586	620.8522	616.0337	428.3522	499.1944	515.6402	**407.4043**
Joint_4	Accuracy	0.9806	**0.9833**	0.9611	0.9750	0.9639	0.9556	0.9694
	Precision	0.9647	**0.9699**	0.9651	0.9542	0.9637	0.9622	0.9689
	Recall	**1.0000**	**1.0000**	0.9556	**1.0000**	0.9667	0.9444	0.9722
	F1-score	0.9815	**0.9842**	0.9556	0.9761	0.9612	0.9501	0.9678
	AUROC	**0.9917**	0.9889	0.9869	0.9861	0.9889	0.9796	0.9877
	Runtime (s)	1112.5916	583.6341	572.6494	418.8274	471.5519	506.8157	**393.5016**
Joint_5	Accuracy	**0.9750**	0.9500	0.9556	0.9528	0.9444	0.9361	0.9583
	Precision	0.9637	0.9620	0.9575	0.9559	0.9591	0.9626	**0.9654**
	Recall	**0.9889**	0.9333	0.9556	0.9500	0.9278	0.9056	0.9500
	F1-score	**0.9753**	0.9400	0.9537	0.9511	0.9397	0.9297	0.9552
	AUROC	0.9883	0.9807	0.9838	**0.9903**	0.9790	0.9756	0.9835
	Runtime (s)	1328.1011	646.5653	634.9948	442.2009	525.9992	541.4374	**414.9360**
Joint_6	Accuracy	0.9611	0.9778	0.9500	0.9667	0.9750	0.9722	**0.9806**
	Precision	**1.0000**	0.9947	**1.0000**	0.9805	**1.0000**	**1.0000**	**1.0000**
	Recall	0.9222	**0.9611**	0.9000	0.9500	0.9500	0.9444	**0.9611**
	F1-score	0.9474	0.9768	0.9345	0.9636	0.9731	0.9682	**0.9784**
	AUROC	0.9994	**1.0000**	0.9978	0.9898	**1.0000**	**1.0000**	**1.0000**
	Runtime (s)	1369.1617	656.0717	650.9967	479.1345	569.1947	591.4029	**428.4101**
Total runtime (s)		7392.3935	3596.9353	3548.6338	2572.0520	2950.1329	3101.7305	**2398.9854**

curve. AUROC indicates the probability of accurate detection and can show how much the anomaly detection method is capable of distinguishing between classes. Thus, AUROC is chosen as the performance metric in this work. The greater the values of the five performance metrics, the better the performance of the anomaly detection method.

6.3. Performance analysis of EDLDSS

6.3.1. Parameter settings

In this work, the population size NP is set to 10, the maximum number of iterations $Iter_{\max}$ is set to 20, the maximum number of layers $MaxNum_{layer}$ is set to 50, the mean number of layers $MeanNum_{layer}$ is set to 10, the maximum number of pooling layers $MaxNum_{pooling}$ is set to 4, the number of nearest neighbors k is set to 3, and the contamination in KNN is set to 0.01.

Table 5
Friedman test results for performance metrics obtained by EDLDSS and the comparative algorithms based on six state-of-the-art metaheuristic algorithms.

Algorithm	Accuracy	Precision	Recall	F1-score	AUROC
EDLTLBO	4.833	3.000	4.667	4.500	3.333
EDLMFO	2.667	4.000	3.083	2.833	4.000
EDLMVO	4.667	3.500	4.167	4.667	4.333
EDLGWO	4.833	6.000	3.750	4.500	4.167
EDLWOA	4.333	4.000	5.000	4.667	4.333
EDLSSA	4.833	5.167	4.917	5.000	5.083
EDLDSS	**1.833**	**2.333**	**2.417**	**1.833**	**2.750**
p-value	0.068655	**0.045725**	0.275939	0.092893	0.601349

6.3.2. Comparison of FVLDSS and state-of-the-art metaheuristic algorithms

During the evolution of the deep learning architecture in EDLDSS, FVLDSS is used to simultaneously optimize the network structure and hyperparameters of CNN. To analyze the effect of the metaheuristic algorithm, we have compared FVLDSS with six state-of-the-art metaheuristic algorithms including TLBO [84], Moth-Flame Optimization (MFO) [85], Multi-Verse Optimizer (MVO) [86], GWO [87], WOA [88], and Salp Swarm Algorithm (SSA) [89]. These metaheuristic algorithms have no control parameters other than the population size and the maximum number of iterations, and their excellent performance in handling optimization problems has been demonstrated in their original studies.

To make a fair comparison, the encoding and alignment strategies provided in Section 5.1 are introduced into TLBO, MFO, MVO, GWO, WOA, and SSA to transform them into the flexible variable-length versions, which are called FVLTLBO, FVLMFO, FVLMVO, FVLGWO, FVLWOA, and FVLSSA. Then, the six metaheuristic algorithms are applied to the evolution of the deep learning architecture in terms of Section 5.1, which are called EDLTLBO, EDLMFO, EDLMVO, EDLGWO, EDLWOA, and EDLSSA. Finally, the corresponding anomaly detection methods are achieved in terms of Section 5.2. The common parameters of the six comparative algorithms are the same as those in Section 6.3.1.

All algorithms are carried out 10 times. For six robot joints, the mean values of Accuracy, Precision, Recall, F1-score, AUROC, and runtime obtained by all algorithms are summarized in Table 4. As shown in Table 4, the values of most performance metrics obtained by EDLDSS are greater than those of other algorithms for three robot joints (Joint_2, Joint_3, and Joint_6), the values of most performance metrics obtained by EDLTLBO are greater than those of other algorithms for Joint_5, the values of most performance metrics obtained by EDLMFO are greater

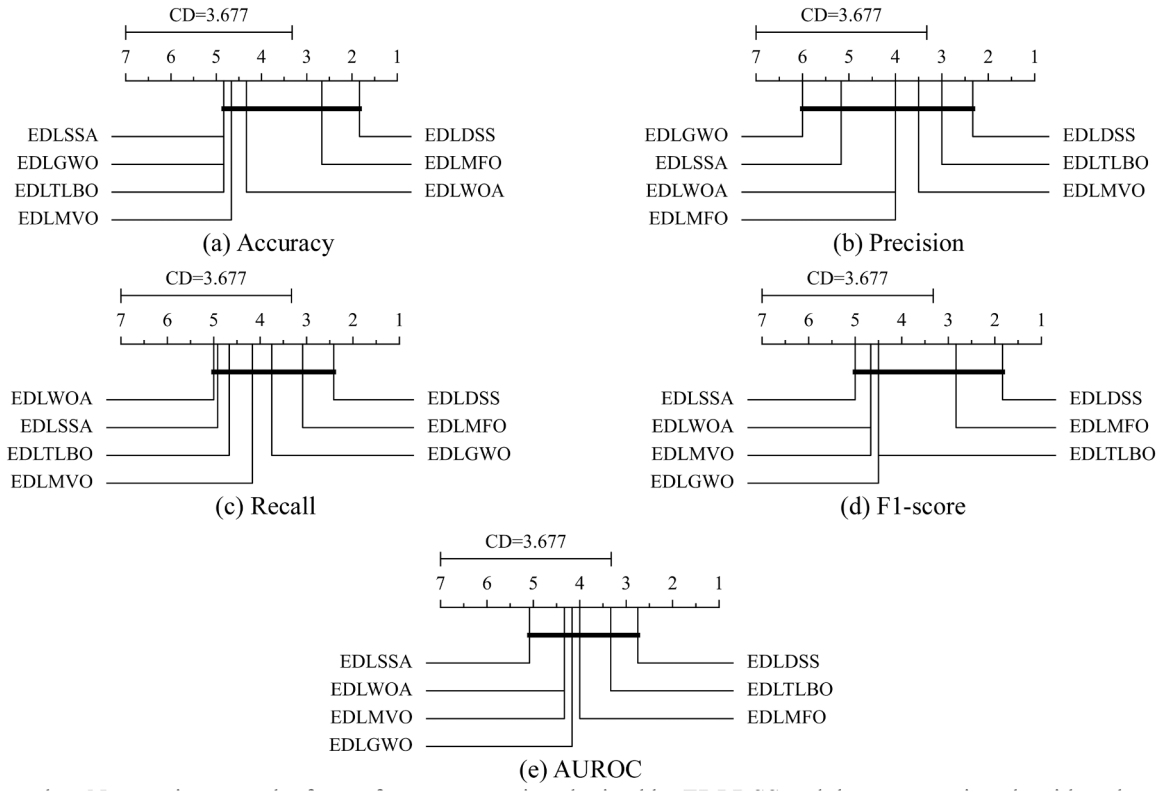

Fig. 9. Post-hoc Nemenyi test results for performance metrics obtained by EDLDSS and the comparative algorithms based on six state-of-the-art meta-heuristic algorithms.

Table 6
Optimal CNN architecture obtained by EDLDSS for Joint_1.

Layer	Type	Configuration
1.	Convolutional	Number of convolutional kernels: 82, kernel size: 7 × 7, stride: 2, padding: valid, activation function: ELU
2.	Maximum pooling	Pooling size: 4 × 4, stride: 4, padding: valid
3.	Maximum pooling	Pooling size: 4 × 4, stride: 4, padding: valid
4.	Convolutional	Number of convolutional kernels: 85, kernel size: 6 × 6, stride: 4, padding: valid, activation function: ELU
5.	Convolutional	Number of convolutional kernels: 86, kernel size: 8 × 8, stride: 3, padding: same, activation function: Tanh
6.	Maximum pooling	Pooling size: 1 × 1, stride: 2, padding: same
7.	Convolutional	Number of convolutional kernels: 95, kernel size: 1 × 1, stride: 2, padding: same, activation function: ReLU
8.	Convolutional	Number of convolutional kernels: 109, kernel size: 3 × 3, stride: 1, padding: same, activation function: ELU
9.	Convolutional	Number of convolutional kernels: 25, kernel size: 8 × 8, stride: 1, padding: same, activation function: ReLU
10.	Convolutional	Number of convolutional kernels: 111, kernel size: 6 × 6, stride: 2, padding: same, activation function: GELU
11.	Convolutional	Number of convolutional kernels: 112, kernel size: 2 × 2, stride: 2, padding: same, activation function: Tanh
12.	Maximum pooling	Pooling size: 1 × 1, stride: 3, padding: same
13.	Global average pooling	-

Table 7
Optimal CNN architecture obtained by EDLDSS for Joint_2.

Layer	Type	Configuration
1.	Convolutional	Number of convolutional kernels: 24, kernel size: 3 × 3, stride: 2, padding: valid, activation function: GELU
2.	Convolutional	Number of convolutional kernels: 124, kernel size: 8 × 8, stride: 1, padding: valid, activation function: ReLU
3.	Convolutional	Number of convolutional kernels: 77, kernel size: 6 × 6, stride: 3, padding: valid, activation function: GELU
4.	Maximum pooling	Pooling size: 4 × 4, stride: 3, padding: valid
5.	Convolutional	Number of convolutional kernels: 102, kernel size: 3 × 3, stride: 1, padding: valid, activation function: Tanh
6.	Convolutional	Number of convolutional kernels: 60, kernel size: 3 × 3, stride: 1, padding: valid, activation function: Tanh
7.	Convolutional	Number of convolutional kernels: 79, kernel size: 4 × 4, stride: 3, padding: valid, activation function: Tanh
8.	Average pooling	Pooling size: 3 × 3, stride: 1, padding: same
9.	Convolutional	Number of convolutional kernels: 90, kernel size: 2 × 2, stride: 3, padding: same, activation function: Tanh
10.	Global average pooling	-

than those of other algorithms for Joint_4, and the values of most performance metrics obtained by EDLMVO are greater than those of other algorithms for Joint_1. The statistical results indicate that EDLDSS is superior to the six comparative algorithms for most robot joints. This is because FVLDSS has better convergence performance than other meta-heuristic algorithms due to its strong exploration and exploitation capabilities. In addition, EDLDSS takes much less runtime than the six comparative algorithms, especially EDLTLBO, which indicates that EDLDSS has the low computational complexity.

To evaluate the difference among the seven algorithms, the Friedman test [90] for the five performance metrics obtained by all algorithms is conducted at a significance level of 0.05. The Friedman test is a nonparametric statistical method to check the statistical significance of the algorithms. In this work, the null hypothesis of the Friedman test is that the performance of all algorithms is equivalent in terms of the performance metric. In the Friedman test, all algorithms are ranked in terms of their performance metrics. Then, the *p*-value of the Friedman

Table 8
Optimal CNN architecture obtained by EDLDSS for Joint_3.

Layer	Type	Configuration
1.	Convolutional	Number of convolutional kernels: 116, kernel size: 1 × 1, stride: 2, padding: valid, activation function: ReLU
2.	Maximum pooling	Pooling size: 4 × 4, stride: 3, padding: valid
3.	Convolutional	Number of convolutional kernels: 72, kernel size: 7 × 7, stride: 3, padding: valid, activation function: Tanh
4.	Maximum pooling	Pooling size: 4 × 4, stride: 3, padding: valid
5.	Convolutional	Number of convolutional kernels: 26, kernel size: 6 × 6, stride: 1, padding: same, activation function: ELU
6.	Convolutional	Number of convolutional kernels: 72, kernel size: 6 × 6, stride: 3, padding: same, activation function: GELU
7.	Average pooling	Pooling size: 3 × 3, stride: 4, padding: same
8.	Convolutional	Number of convolutional kernels: 101, kernel size: 1 × 1, stride: 1, padding: same, activation function: GELU
9.	Global average pooling	-

Table 9
Optimal CNN architecture obtained by EDLDSS for Joint_4.

Layer	Type	Configuration
1.	Convolutional	Number of convolutional kernels: 36, kernel size: 3 × 3, stride: 3, padding: valid, activation function: Tanh
2.	Convolutional	Number of convolutional kernels: 89, kernel size: 2 × 2, stride: 2, padding: valid, activation function: ELU
3.	Maximum pooling	Pooling size: 2 × 2, stride: 4, padding: valid
4.	Maximum pooling	Pooling size: 3 × 3, stride: 2, padding: valid
5.	Convolutional	Number of convolutional kernels: 12, kernel size: 8 × 8, stride: 3, padding: same, activation function: Tanh
6.	Convolutional	Number of convolutional kernels: 63, kernel size: 1 × 1, stride: 4, padding: valid, activation function: ReLU
7.	Maximum pooling	Pooling size: 3 × 3, stride: 2, padding: same
8.	Convolutional	Number of convolutional kernels: 21, kernel size: 7 × 7, stride: 4, padding: same, activation function: ELU
9.	Convolutional	Number of convolutional kernels: 80, kernel size: 2 × 2, stride: 1, padding: same, activation function: ELU
10.	Maximum pooling	Pooling size: 4 × 4, stride: 4, padding: same
11.	Global average pooling	-

Table 10
Optimal CNN architecture obtained by EDLDSS for Joint_5.

Layer	Type	Configuration
1.	Convolutional	Number of convolutional kernels: 108, kernel size: 4 × 4, stride: 2, padding: valid, activation function: ReLU
2.	Maximum pooling	Pooling size: 3 × 3, stride: 2, padding: valid
3.	Maximum pooling	Pooling size: 3 × 3, stride: 2, padding: valid
4.	Convolutional	Number of convolutional kernels: 119, kernel size: 8 × 8, stride: 4, padding: valid, activation function: ReLU
5.	Convolutional	Number of convolutional kernels: 36, kernel size: 6 × 6, stride: 2, padding: same, activation function: Tanh
6.	Average pooling	Pooling size: 2 × 2, stride: 1, padding: valid
7.	Maximum pooling	Pooling size: 3 × 3, stride: 4, padding: same
8.	Convolutional	Number of convolutional kernels: 99, kernel size: 5 × 5, stride: 3, padding: same, activation function: ReLU
9.	Convolutional	Number of convolutional kernels: 98, kernel size: 6 × 6, stride: 2, padding: same, activation function: GELU
10.	Convolutional	Number of convolutional kernels: 117, kernel size: 1 × 1, stride: 4, padding: same, activation function: ELU
11.	Convolutional	Number of convolutional kernels: 117, kernel size: 1 × 1, stride: 4, padding: same, activation function: ELU
12.	Global average pooling	-

Table 11
Optimal CNN architecture obtained by EDLDSS for Joint_6.

Layer	Type	Configuration
1.	Convolutional	Number of convolutional kernels: 107, kernel size: 3 × 3, stride: 3, padding: valid, activation function: ReLU
2.	Convolutional	Number of convolutional kernels: 71, kernel size: 4 × 4, stride: 1, padding: valid, activation function: ELU
3.	Maximum pooling	Pooling size: 3 × 3, stride: 2, padding: valid
4.	Convolutional	Number of convolutional kernels: 70, kernel size: 2 × 2, stride: 1, padding: valid, activation function: Tanh
5.	Convolutional	Number of convolutional kernels: 3, kernel size: 4 × 4, stride: 3, padding: valid, activation function: ELU
6.	Convolutional	Number of convolutional kernels: 64, kernel size: 7 × 7, stride: 2, padding: valid, activation function: ELU
7.	Average pooling	Pooling size: 2 × 2, stride: 3, padding: valid
8.	Maximum pooling	Pooling size: 2 × 2, stride: 1, padding: same
9.	Global average pooling	-

test, which is used to assess how all algorithms differ, is calculated. The null hypothesis will be rejected if the p-value is smaller than the significance level of 0.05, which indicates that there is a significant difference among all algorithms in terms of the performance metric. The significant difference indicates that one algorithm is considered to be significantly superior or inferior to the other in terms of the performance metric. For each performance metric, the Friedman average ranks of the seven algorithms are provided in Table 5, and the p-values of the Friedman test are inserted at the bottom of Table 5.

Table 5 shows that the Friedman average ranks of EDLDSS for the five performance metrics are the best with 1.833, 2.333, 2.417, 1.833, and 2.750. The second-best Friedman average ranks for Accuracy, Recall, and F1-score are 2.667, 3.083, and 2.833 obtained by EDLMFO, and the second-best Friedman average ranks for Precision and AUROC are 3.000 and 3.333 obtained by EDLTLBO. The p-values indicate that there is no significant difference among the seven algorithms in terms of all performance metrics except Precision.

Since the Friedman test cannot be used to assess which algorithms differ from each other, the post-hoc Nemenyi test using Critical Distance (CD) [91] is conducted for pairwise comparisons. One algorithm is considered successful over the other if the CD is smaller than the Friedman average rank difference, and the two algorithms have no statistical difference if they are connected by a thick line. Fig. 9 shows the post-hoc Nemenyi test results to assess the statistical difference between EDLDSS and each comparative algorithm in terms of the performance metric. We can note that the CD is 3.677, and EDLDSS performs similarly to the six comparative algorithms in terms of all performance metrics.

To sum up, EDLDSS can achieve competitive performance with less runtime against the six comparative algorithms in automatically searching for optimal deep learning architectures for anomaly detection of robot joints. Thus, when using the same encoding and alignment strategies, FVLDSS is more effective and efficient than other meta-heuristic algorithms. The optimal CNN architectures obtained by EDLDSS for six robot joints are provided in Tables 6–11.

6.3.3. *Comparison of KNN and popular anomaly detectors*

During the evaluation of the decoded CNN architecture in EDLDSS, KNN is used as the anomaly detector to replace the fully-connected layer in CNN. To analyze the effect of the anomaly detector, we have compared KNN with four popular anomaly detectors including

Table 12
Comparison of EDLDSS and the comparative algorithms based on four popular anomaly detectors for anomaly detection of robot joints.

Robot joint	Metric	CA-COF	CA-IF	CA-LOF	CA-OCSVM	EDLDSS
Joint_1	Accuracy	0.6917	0.7917	0.9417	0.9361	**0.9639**
	Precision	0.9607	0.9471	0.9642	0.9447	**0.9747**
	Recall	0.4000	0.6167	0.9222	0.9278	**0.9556**
	F1-score	0.5638	0.7155	0.9220	0.9352	**0.9623**
	AUROC	0.8880	0.9412	0.9867	0.9932	**0.9963**
	Runtime (s)	377.6899	365.2775	353.3061	**347.3198**	353.0079
Joint_2	Accuracy	0.6194	0.7472	0.9500	0.9611	**0.9861**
	Precision	0.9800	**1.0000**	0.9737	0.9547	0.9842
	Recall	0.2444	0.4944	0.9278	0.9722	**0.9889**
	F1-score	0.3877	0.6306	0.9418	0.9601	**0.9863**
	AUROC	0.6370	0.9684	**0.9969**	0.9920	0.9941
	Runtime (s)	410.9879	416.3401	399.4294	**397.6393**	401.7256
Joint_3	Accuracy	0.6250	0.7222	0.9194	0.9167	**0.9278**
	Precision	0.8444	0.9333	0.9777	0.9493	**0.9832**
	Recall	0.3278	0.4611	0.8611	**0.8778**	0.8722
	F1-score	0.4550	0.5872	0.9050	0.9070	**0.9181**
	AUROC	0.5802	0.9387	0.9923	0.9790	**0.9926**
	Runtime (s)	411.5280	418.4695	409.6504	**403.9126**	407.4043
Joint_4	Accuracy	0.6806	0.8306	**0.9833**	0.9389	0.9694
	Precision	0.9350	**0.9947**	0.9689	0.8984	0.9689
	Recall	0.4000	0.6667	**1.0000**	**1.0000**	0.9722
	F1-score	0.5481	0.7413	**0.9839**	0.9446	0.9678
	AUROC	0.6090	0.9691	**1.0000**	**1.0000**	0.9877
	Runtime (s)	409.0577	415.8913	396.1515	**387.8069**	393.5016
Joint_5	Accuracy	0.6556	0.7806	0.9333	0.9306	**0.9583**
	Precision	0.9273	0.9120	0.9590	0.9254	**0.9654**
	Recall	0.3556	0.6056	0.9056	0.9389	**0.9500**
	F1-score	0.5053	0.7131	0.9275	0.9282	**0.9552**
	AUROC	0.5497	0.9188	0.9790	0.9824	**0.9835**
	Runtime (s)	431.0133	425.0886	417.6844	**404.2274**	414.9360
Joint_6	Accuracy	0.5806	0.7083	0.9639	0.9444	**0.9806**
	Precision	0.7514	0.9667	**1.0000**	**1.0000**	**1.0000**
	Recall	0.2556	0.4278	0.9278	0.8889	**0.9611**
	F1-score	0.3748	0.5655	0.9435	0.9321	**0.9784**
	AUROC	0.5852	0.9302	0.9907	0.9898	**1.0000**
	Runtime (s)	449.9889	444.8503	431.6207	**420.4014**	428.4101
Total runtime (s)		2490.2657	2485.9173	2407.8425	**2361.3075**	2398.9854

Table 13
Friedman test results for performance metrics obtained by EDLDSS and the comparative algorithms based on four popular anomaly detectors.

Anomaly detector	Accuracy	Precision	Recall	F1-score	AUROC
CA-COF	5.000	3.833	5.000	5.000	5.000
CA-IF	4.000	3.167	4.000	4.000	4.000
CA-LOF	2.000	2.417	2.583	2.500	2.083
CA-OCSVM	2.833	4.000	1.917	2.333	2.417
EDLDSS	**1.167**	**1.583**	**1.500**	**1.167**	**1.500**
p-value	**0.000157**	**0.037326**	**0.000346**	**0.000226**	**0.000416**

Connectivity-Based Outlier Factor (COF) [92], Isolation Forest (IF) [93], Local Outlier Factor (LOF) [94], and One-Class Support Vector Machine (OCSVM) [95].

To make a fair comparison, four comparative algorithms are developed by using the four popular anomaly detectors to replace KNN in EDLDSS, which are called CA-COF, CA-IF, CA-LOF, and CA-OCSVM. Then, the corresponding anomaly detection methods are achieved in terms of Section 5.2. The common parameters of the four comparative algorithms are the same as those in Section 6.3.1.

All algorithms are conducted 10 times. For six robot joints, the mean values of Accuracy, Precision, Recall, F1-score, AUROC, and runtime obtained by all algorithms are recorded in Table 12. Then, for the five performance metrics obtained by all algorithms, Table 13 summarizes the Friedman test results at a significance level of 0.05, and Fig. 10 describes the post-hoc Nemenyi test results.

As shown in Table 12, the values of most performance metrics obtained by EDLDSS are greater than those of other algorithms for all robot joints except Joint_4 and only smaller than those of CA-LOF for Joint_4. The statistical results indicate that KNN is more effective than the four popular anomaly detectors for anomaly detection of robot joints, followed by CA-LOF. Moreover, CA-OCSVM has the shortest runtime, whereas there is no significant difference in the runtime of all algorithms because the runtime is mainly affected by the metaheuristic algorithm rather than the anomaly detector.

Table 13 shows that the Friedman average ranks of EDLDSS for the five performance metrics are the best with 1.167, 1.583, 1.500, 1.167, and 1.500. The second-best Friedman average ranks for Accuracy, Precision, and AUROC are 2.000, 2.417, and 2.083 obtained by CA-LOF, and the second-best Friedman average ranks for Recall and F1-score are 1.917 and 2.333 obtained by CA-OCSVM. The *p*-values indicate that there is a significant difference among the five algorithms in terms of all performance metrics. As shown in Fig. 10, the CD is 2.490. The post-hoc Nemenyi test results indicate that EDLDSS significantly outperforms CA-COF and CA-IF in terms of all performance metrics except Precision, whereas it is similar to all comparative algorithms in terms of Precision.

In summary, EDLDSS compared with the four comparative algorithms can provide better anomaly detection performance for six robot joints in similar runtime, which indicates that it is reasonable and feasible to choose KNN as the anomaly detector in EDLDSS.

6.3.4. *Comparison of optimal CNN architectures and advanced CNN architectures*

To validate the performance of the optimal CNN architectures obtained by EDLDSS in extracting deep features for anomaly detection of robot joints, six advanced CNN architectures, including ShuffleNet-v2 [96], GoogLeNet (Inception-v4) [97], SqueezeNet [98], ResNet-152 [99], VGG16 [100], and VGG19 [100], is chosen to compare with the

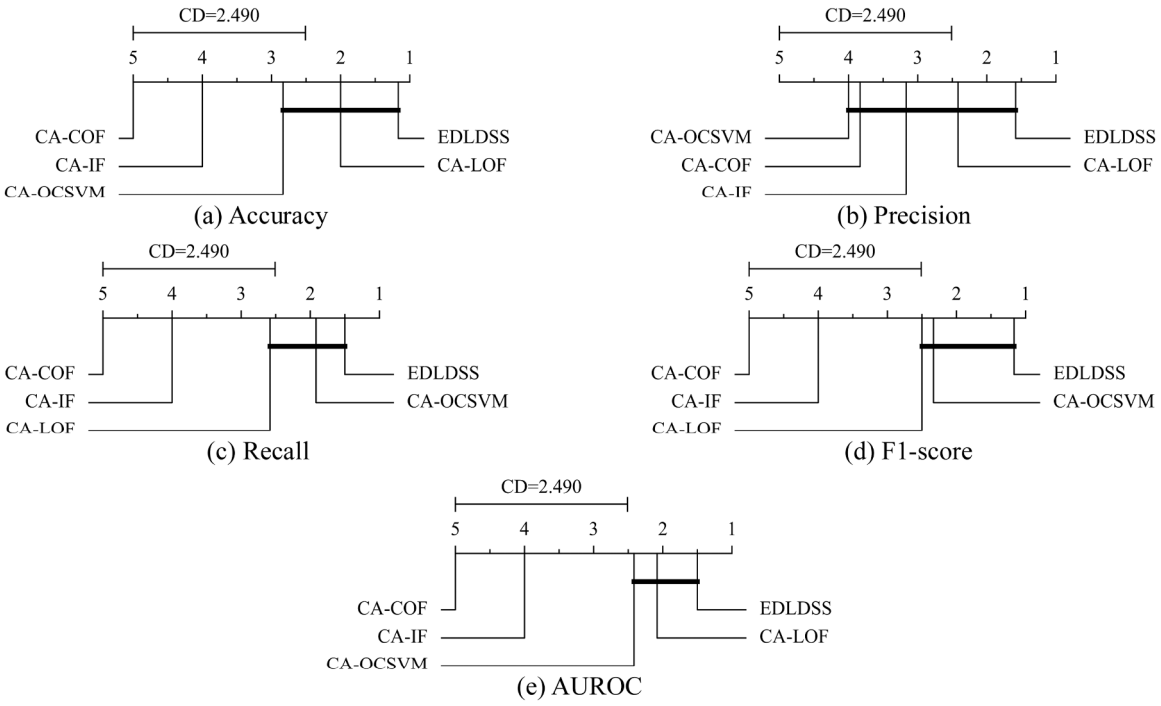

Fig. 10. Post-hoc Nemenyi test results for performance metrics obtained by EDLDSS and the comparative algorithms based on four popular anomaly detectors.

Table 14
Comparison of the anomaly detection methods based on optimal CNN architectures and six advanced CNN architectures for anomaly detection of robot joints.

Robot joint	Metric	AM-ShuffleNet	AM-GoogLeNet	AM-SqueezeNet	AM-ResNet	AM-VGG16	AM-VGG19	AM-OCNN
Joint_1	Accuracy	0.9556	0.9444	0.9444	0.8611	0.9750	**0.9944**	**0.9944**
	Precision	0.9534	0.9563	0.9236	0.9404	0.9547	0.9895	**1.0000**
	Recall	0.9611	0.9333	0.9722	0.7722	**1.0000**	**1.0000**	0.9889
	F1-score	0.9554	0.9425	0.9460	0.8477	0.9762	**0.9946**	0.9943
	AUROC	0.9972	0.9886	0.9904	0.9407	**1.0000**	**1.0000**	**1.0000**
	Runtime (s)	8.4470	10.2516	2.8502	19.4518	11.9626	15.2926	**1.1320**
Joint_2	Accuracy	**0.9972**	0.9944	0.9944	0.9833	0.9944	**0.9972**	**0.9972**
	Precision	**0.9947**	0.9895	0.9895	0.9699	0.9895	**0.9947**	**0.9947**
	Recall	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	F1-score	**0.9973**	0.9946	0.9946	0.9842	0.9946	**0.9973**	**0.9973**
	AUROC	0.9972	**1.0000**	0.9972	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	Runtime (s)	6.5840	10.8831	2.9150	18.9670	12.6313	15.2229	**1.1450**
Joint_3	Accuracy	0.9639	0.9722	0.9583	0.8222	0.9528	0.9750	**0.9917**
	Precision	0.9595	0.9474	0.9430	0.8868	0.9574	0.9526	**0.9892**
	Recall	0.9722	**1.0000**	0.9778	0.7278	0.9500	**1.0000**	0.9944
	F1-score	0.9627	0.9730	0.9594	0.7848	0.9511	0.9757	**0.9917**
	AUROC	0.9856	0.9836	0.9769	0.8858	0.9894	0.9855	**0.9966**
	Runtime (s)	8.1865	10.4351	3.3332	19.2595	12.5488	15.6168	**0.9745**
Joint_4	Accuracy	0.9833	0.9889	0.9944	0.9861	0.9722	0.9861	**0.9972**
	Precision	0.9689	0.9789	0.9895	0.9737	0.9479	0.9742	**0.9947**
	Recall	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	F1-score	0.9839	0.9892	0.9946	0.9865	0.9731	0.9866	**0.9973**
	AUROC	0.9944	0.9917	0.9972	0.9972	**1.0000**	**1.0000**	**1.0000**
	Runtime (s)	3.1918	10.5006	3.0056	18.9282	12.5224	15.5535	**0.5166**
Joint_5	Accuracy	0.9750	0.9917	0.9833	0.9611	0.9806	0.9917	**0.9944**
	Precision	0.9842	0.9842	**0.9895**	0.9587	0.9632	**0.9895**	**0.9895**
	Recall	0.9667	**1.0000**	0.9778	0.9667	**1.0000**	0.9944	**1.0000**
	F1-score	0.9729	0.9919	0.9826	0.9608	0.9811	0.9917	**0.9946**
	AUROC	0.9963	0.9972	**1.0000**	0.9860	**1.0000**	**1.0000**	**1.0000**
	Runtime (s)	3.2629	10.4259	3.2525	19.1952	12.4850	15.5410	**0.9827**
Joint_6	Accuracy	0.9944	0.9944	**0.9972**	0.9889	0.9917	**0.9972**	**0.9972**
	Precision	0.9900	**1.0000**	0.9947	0.9795	0.9847	**1.0000**	0.9947
	Recall	**1.0000**	0.9889	**1.0000**	**1.0000**	**1.0000**	0.9944	**1.0000**
	F1-score	0.9947	0.9941	**0.9973**	0.9893	0.9920	0.9971	**0.9973**
	AUROC	0.9969	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
	Runtime (s)	3.2963	10.3691	3.2567	19.3447	12.5229	15.7286	**1.8396**
Total runtime (s)		32.9686	62.8654	18.6133	115.1464	74.6731	92.9554	**6.5904**

Table 15
Friedman test results for performance metrics obtained by the anomaly detection methods based on optimal CNN architectures and six advanced CNN architectures.

CNN architecture	Accuracy	Precision	Recall	F1-score	AUROC
AM-ShuffleNet	4.417	4.083	4.583	4.333	5.417
AM-GoogLeNet	3.917	3.667	4.083	4.000	4.917
AM-SqueezeNet	3.917	4.250	4.000	3.750	4.667
AM-ResNet	6.583	6.500	5.250	6.667	5.333
AM-VGG16	5.333	5.167	3.417	5.333	2.500
AM-VGG19	2.417	2.583	3.500	2.500	2.833
AM-OCNN	**1.417**	**1.750**	**3.167**	**1.417**	**2.333**
p-value	**0.000494**	**0.002769**	0.320378	**0.000578**	**0.004222**

optimal CNN architectures.

For a fair comparison, the Anomaly Detection Method Based on Optimal CNN Architectures (AM-OCNN) is designed in terms of Section 5.2, in which ***Step 5*** is removed and the optimal CNN architectures shown in Tables 6–11 are used in ***Step 6***. Meanwhile, six anomaly detection methods can be achieved by replacing the optimal CNN architectures with the six advanced CNN architectures, which are called AM-ShuffleNet, AM-GoogLeNet, AM-SqueezeNet, AM-ResNet, AM-VGG16, and AM-VGG19. The common parameters of all anomaly detection methods are the same as those in Section 6.3.1.

All anomaly detection methods are carried out 10 times. For six robot joints, the mean values of Accuracy, Precision, Recall, F1-score, AUROC, and runtime obtained by all anomaly detection methods are presented in Table 14. Then, for the five performance metrics obtained by all anomaly detection methods, the Friedman test and post-hoc Nemenyi test results at a significance level of 0.05 are provided in Table 15 and Fig. 11, respectively.

As shown in Table 14, the values of most performance metrics obtained by AM-OCNN are greater than those of other anomaly detection methods for three robot joints (Joint_3, Joint_4, and Joint_5), the values of most performance metrics obtained by AM-VGG19 are greater than those of other anomaly detection methods for Joint_1, the values of most performance metrics obtained by AM-OCNN and AM-VGG19 are the same and greater than those of other anomaly detection methods for Joint_2, and the values of most performance metrics obtained by AM-OCNN and AM-SqueezeNet are the same and greater than those of other anomaly detection methods for Joint_6. The statistical results indicate that the optimal CNN architectures obtained by EDLDSS have a great ability to extract deep features for anomaly detection of robot joints. Moreover, AM-OCNN takes much less runtime than other anomaly detection methods.

In terms of Table 15, the Friedman average ranks of AM-OCNN for the five performance metrics are the best with 1.417, 1.750, 3.167, 1.417, and 2.333. The second-best Friedman average ranks for Accuracy, Precision, and F1-score are 2.417, 2.583, and 2.500 obtained by AM-VGG19, and the second-best Friedman average ranks for Recall and AUROC are 3.417 and 2.500 obtained by AM-VGG16. The *p*-values indicate that there is a significant difference among the seven anomaly detection methods in terms of all performance metrics except Recall. In addition, Fig. 11 shows that the CD is 3.677. The post-hoc Nemenyi test results indicate that AM-OCNN is significantly superior to AM-ResNet and AM-VGG16 in terms of Accuracy and F1-score and to AM-ResNet in terms of Precision, whereas it performs similarly to other anomaly detection methods in terms of Recall and AUROC.

Furthermore, the loss curves of the anomaly detection methods based on optimal CNN architectures and six advanced CNN architectures on the training dataset for six robot joints are shown in Fig. 12. The loss function is the binary cross-entropy. AM-OCNN converges faster than other anomaly detection methods for Joint_4, and it converges slower only than AM-VGG16 and AM-VGG19 for other robot joints. For all robot joints, AM-OCNN, AM-VGG16, and AM-VGG19 can rapidly converge to 0 within 35 epochs, whereas other anomaly detection methods show worse convergence performance within the finite 50 epochs. In particular, for Joint_4, Joint_5, and Joint_6, there is no significant difference in the convergence speed among AM-OCNN, AM-VGG16, and AM-VGG19. Meanwhile, the convergence speeds of AM-

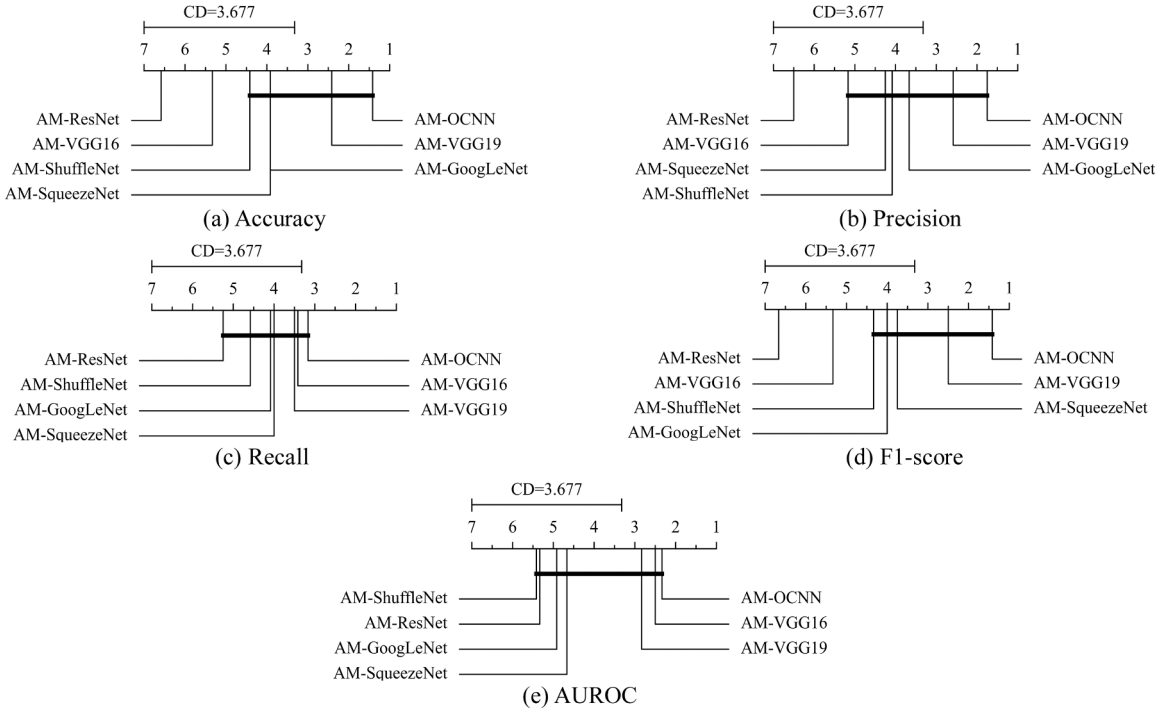

Fig. 11. Post-hoc Nemenyi test results for performance metrics obtained by the anomaly detection methods based on optimal CNN architectures and six advanced CNN architectures.

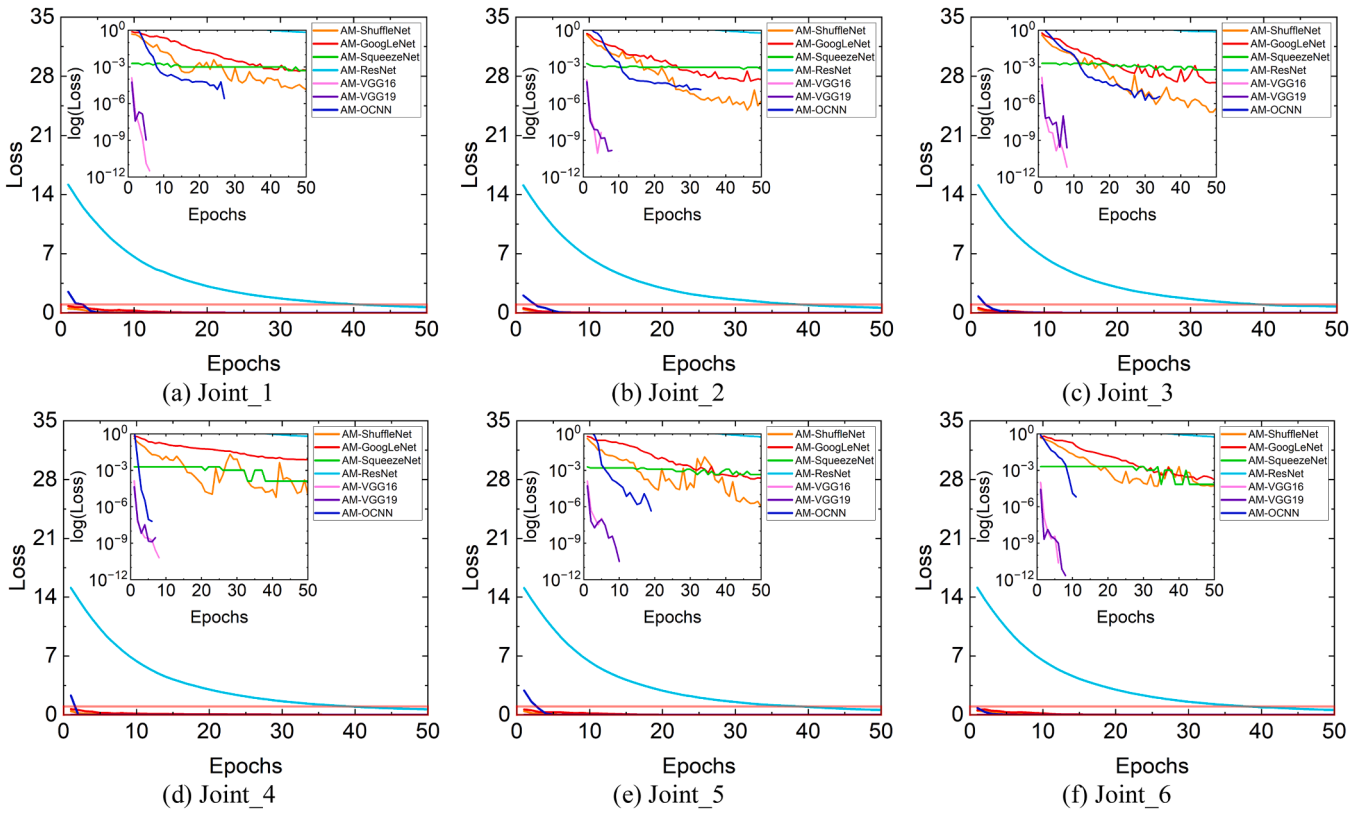

Fig. 12. Loss curves of the anomaly detection methods based on optimal CNN architectures and six advanced CNN architectures on the training dataset for six robot joints.

OCNN for different robot joints are different because six optimal CNN architectures obtained by EDLDSS for six robot joints are used in AM-OCNN.

In conclusion, AM-OCNN has better anomaly detection performance than the anomaly detection methods based on six advanced CNN architectures, which indicates that using EDLDSS to automatically obtain the optimal CNN architectures for six robot joints can significantly improve the performance of the anomaly detection method. In other words, the optimal CNN architectures obtained by EDLDSS exhibit stronger feature extraction capability than the six advanced CNN architectures in handling the anomaly detection problem of robot joints.

6.4. Discussion

In Section 6.3, a comprehensive performance analysis of EDLDSS is provided by conducting three comparisons. In Section 6.3.2, we have analyzed the effect of the metaheuristic algorithm on EDLDSS by comparing FVLDSS with six state-of-the-art metaheuristic algorithms including TLBO, MFO, MVO, GWO, WOA, and SSA. FVLDSS provides more competitive results and higher efficiency than the comparative metaheuristic algorithms in simultaneously optimizing the network structure and hyperparameters of CNN. In Section 6.3.3, we have analyzed the effect of the anomaly detector on EDLDSS by comparing KNN with four popular anomaly detectors including COF, IF, LOF, and OCSVM. KNN provides superior results and similar efficiency to the comparative anomaly detectors, which demonstrates the rationality of KNN as the anomaly detector of EDLDSS. According to Section 6.3.2 and Section 6.3.3, we can note that the efficiency of EDLDSS is mainly affected by the metaheuristic algorithm rather than the anomaly detector. In Section 6.3.4, we have compared the optimal CNN architectures obtained by EDLDSS with six advanced CNN architectures including ShuffleNet-v2, GoogLeNet, SqueezeNet, ResNet-152, VGG16, and VGG19. When using the same anomaly detector, the anomaly detection method based on the optimal CNN architectures obtained by EDLDSS outperforms the anomaly detection methods based on the comparative CNN architectures in anomaly detection accuracy and efficiency. This phenomenon indicates that the optimal CNN architectures obtained by EDLDSS have a better ability to extract deep features for anomaly detection of robot joints than the comparative CNN architectures.

To the best of our knowledge, EDLDSS, as an integration of CNN, KNN, and FVLDSS, is first developed to automatically search for optimal deep learning architectures for anomaly detection, especially anomaly detection of robot joints. Since the deep learning architectures in current anomaly detection methods are manually designed by researchers, they may not be optimal for anomaly detection of robot joints. Moreover, evolving deep learning architectures by metaheuristic algorithms is not only inefficient but also mainly developed for classification rather than anomaly detection. As discussed above, EDLDSS has the great ability to efficiently and automatically search for optimal deep learning architectures for anomaly detection of robot joints, and the use of the optimal CNN architectures obtained by EDLDSS makes the anomaly detection method achieve high anomaly detection accuracy for six robot joints in a short time. Thus, the design of EDLDSS and its application to anomaly detection of robot joints are highly valuable in the field of predictive maintenance of industrial robots.

7. Conclusions

In this paper, an Evolutionary Deep Learning Approach Using Flexible Variable-Length Dynamic Stochastic Search (EDLDSS) is proposed for anomaly detection of robot joints. In EDLDSS, a Flexible Variable-Length DSS (FVLDSS) is developed by designing and embedding the encoding and alignment strategies into the original DSS, and it is used to simultaneously optimize the network structure and hyperparameters of CNN. Meanwhile, CNN is adopted as the feature extractor, a GAP layer is

adopted to flatten the extracted deep features, and KNN is adopted as the anomaly detector. Subsequently, an anomaly detection method based on EDLDSS is further developed to detect the anomalies in robot joints, and its steps include signal acquisition, data normalization, time-frequency analysis, partition, evolution of the deep learning architecture, and anomaly detection. The vibration signals from six robot joints are acquired by establishing a condition monitoring system using FBG sensors for the industrial robot. After conducting data normalization, the S-transform spectrograms of the vibration signals are obtained to establish the training, validation, and testing datasets for each robot joint, and the training dataset only includes normal samples. Finally, anomaly detection of six robot joints is realized by using six optimal CNN architectures obtained by EDLDSS.

The innovation of this study is: (1) The FBG sensor is first adopted and embedded into the inner space of robot joints, which can effectively acquire high-quality vibration signals from robot joints without increasing the weight of the industrial robot and affecting the movements of the industrial robot; (2) The variable-length operation of the search agent in FVLDSS is consistent with the variable-length CNN, which is conducive to increasing the diversity of the depth of the explored CNN architecture; (3) Evolving deep learning architectures by the flexible variable-length metaheuristic algorithm for anomaly detection (i.e. EDLDSS) is first investigated, which not only has fewer control parameters but also can efficiently generate an optimal CNN architecture for anomaly detection; (4) The anomaly detection method based on EDLDSS is first developed for robot joints, in which applying six optimal CNN architecture automatically obtained by EDLDSS to anomaly detection of six robot joints can significantly improve the anomaly detection accuracy.

Three experiments are conducted. The first experiment investigates the effect of the metaheuristic algorithm. The statistical results demonstrate the superiority of FVLDSS compared with six state-of-the-art metaheuristic algorithms in simultaneously optimizing the network structure and hyperparameters of CNN. The second experiment investigates the effect of the anomaly detector. The statistical results show that KNN is more suitable as the anomaly detector of EDLDSS because it outperforms four popular anomaly detectors. The third experiment compares the optimal CNN architectures with six advanced CNN architectures. The statistical results show that the optimal CNN architectures obtained by EDLDSS can provide better performance in extracting deep features for anomaly detection of robot joints.

In terms of the aforementioned analysis, EDLDSS is considered a significantly powerful and reliable algorithm for anomaly detection of robot joints. However, there are still several limitations to overcome. First, although EDLDSS can automatically and efficiently search for an optimal CNN architecture for a specific anomaly detection task, the anomaly detection method based on EDLDSS is time-consuming because six optimal CNN architectures need to be obtained by EDLDSS for six robot joints. Thus, we will modify EDLDSS to make it have the ability to obtain an optimal CNN architecture which ensures that the anomaly detection method can simultaneously provide high anomaly detection accuracy for six robot joints. Second, since EDLDSS mainly focuses on the evolution of the CNN architecture without paying attention to the effect of the hyperparameters of KNN, the improved EDLDSS, in which the hyperparameters of KNN are evolved along with the evolution of the CNN architecture, will be investigated to enhance the performance of EDLDSS in future work. Finally, the designed encoding strategy in FVLDSS is layer-based, which makes EDLDSS suitable only for the evolution of the layer-based CNN architecture. Considering the positive effect of sophisticated building blocks (e.g. ResNet block and DenseNet block) on the complexity and performance of the CNN architecture, a block-based encoding strategy will be further designed to enable EDLDSS to be used to evolve the block-based CNN architectures for more complex anomaly detection problems.

Furthermore, the findings of this study have significant implications for the efficient and automatic design of deep learning architectures for high-accuracy anomaly detection of robot joints without expert knowledge. EDLDSS not only has the potential to extend its application to other industrial robots, but also provides a new perspective on anomaly detection problems in other scientific and engineering fields, such as condition monitoring of wind turbines [101], quality control in semiconductor manufacturing [102], and structural health monitoring of heritage buildings [103].

CRediT authorship contribution statement

Qi Liu: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Yongchao Yu:** Writing – review & editing, Visualization, Validation, Software, Investigation, Formal analysis, Conceptualization. **Boon Siew Han:** Validation, Supervision, Resources, Project administration, Funding acquisition, Data curation. **Wei Zhou:** Validation, Supervision, Resources, Project administration, Funding acquisition, Data curation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Agency for Science, Technology and Research (A*STAR) under its IAF-ICP Programme I2001E0067 and the Schaeffler Hub for Advanced Research at Nanyang Technological University.

Data availability

Data will be made available on request.

References

[1] K. Lu, C. Chen, T. Wang, L. Cheng, J. Qin, Fault diagnosis of industrial robot based on dual-module attention convolutional neural network, Auton. Intell. Syst. 2 (2022) 12, https://doi.org/10.1007/s43684-022-00031-5.
[2] Y. He, C. Zhao, X. Zhou, W. Shen, MJAR: a novel joint generalization-based diagnosis method for industrial robots with compound faults, Robot. Comput. Integr. Manuf. 86 (2024) 102668, https://doi.org/10.1016/j.rcim.2023.102668.
[3] X. Wang, M. Liu, C. Liu, L. Ling, X. Zhang, Data-driven and knowledge-based predictive maintenance method for industrial robots for the production stability of intelligent manufacturing, Expert Syst. Appl. 234 (2023) 121136, https://doi.org/10.1016/j.eswa.2023.121136.
[4] C. Chen, C. Liu, T. Wang, A. Zhang, W. Wu, L. Cheng, Compound fault diagnosis for industrial robots based on dual-transformer networks, J. Manuf. Syst. 66 (2023) 163–178, https://doi.org/10.1016/j.jmsy.2022.12.006.
[5] J. Long, Y. Qin, Z. Yang, Y. Huang, C. Li, Discriminative feature learning using a multiscale convolutional capsule network from attitude data for fault diagnosis of industrial robots, Mech. Syst. Signal Process. 182 (2023) 109569, https://doi.org/10.1016/j.ymssp.2022.109569.
[6] Z. Pu, D. Cabrera, Y. Bai, C. Li, Generative adversarial one-shot diagnosis of transmission faults for industrial robots, Robot. Comput. -Integr. Manuf. 83 (2023) 102577, https://doi.org/10.1016/j.rcim.2023.102577.
[7] J. Long, J. Mou, L. Zhang, S. Zhang, C. Li, Attitude data-based deep hybrid learning architecture for intelligent fault diagnosis of multi-joint industrial robots, J. Manuf. Syst. 61 (2021) 736–745, https://doi.org/10.1016/j.jmsy.2020.08.010.
[8] G. Li, J.J. Jung, Deep learning for anomaly detection in multivariate time series: approaches, applications, and challenges, Inf. Fusion 91 (2023) 93–102, https://doi.org/10.1016/j.inffus.2022.10.008.
[9] Y. Gao, X. Yin, Z. He, X. Wang, A deep learning process anomaly detection approach with representative latent features for low discriminative and insufficient abnormal data, Comput. Ind. Eng. 176 (2023) 108936, https://doi.org/10.1016/j.cie.2022.108936.
[10] A. Abhaya, B.K. Patra, An efficient method for autoencoder based outlier detection, Expert Syst. Appl. 213 (2023) 118904, https://doi.org/10.1016/j.eswa.2022.118904.

[11] C. Sun, Z. He, H. Lin, L. Cai, H. Cai, M. Gao, Anomaly detection of power battery pack using gated recurrent units based variational autoencoder, Appl. Soft Comput. 132 (2023) 109903, https://doi.org/10.1016/j.asoc.2022.109903.

[12] W. Li, Z. Shang, J. Zhang, M. Gao, S. Qian, A novel unsupervised anomaly detection method for rotating machinery based on memory augmented temporal convolutional autoencoder, Eng. Appl. Artif. Intell. 123 (2023) 106312, https://doi.org/10.1016/j.engappai.2023.106312.

[13] C. Liu, J. Pan, J. Wang, An LSTM-based anomaly detection model for the deformation of concrete dams, Struct. Health Monit. 23 (3) (2024) 1914–1925, https://doi.org/10.1177/14759217231199569.

[14] X. Xia, X. Pan, N. Li, X. He, L. Ma, X. Zhang, N. Ding, GAN-based anomaly detection: a review, Neurocomputing 493 (2022) 497–535, https://doi.org/10.1016/j.neucom.2021.12.093.

[15] T. Chen, X. Liu, B. Xia, W. Wang, Y. Lai, Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder, IEEE Access 8 (2020) 47072–47081, https://doi.org/10.1109/ACCESS.2020.2977892.

[16] Z. Zhong, Y. Zhao, A. Yang, H. Zhang, D. Qiao, Z. Zhang, Industrial robot vibration anomaly detection based on sliding window one-dimensional convolution autoencoder, Shock Vib. 2022 (2022) 1179192, https://doi.org/10.1155/2022/1179192.

[17] H. Yun, H. Kim, Y.H. Jeong, M.B.G. Jun, Autoencoder-based anomaly detection of industrial robot arm using stethoscope based internal sound sensor, J. Intell. Manuf. 34 (2023) 1427–1444, https://doi.org/10.1007/s10845-021-01862-4.

[18] Z. Chen, K. Wu, J. Wu, C. Deng, Y. Wang, Residual shrinkage transformer relation network for intelligent fault detection of industrial robot with zero-fault samples, Knowl. -Based Syst. 268 (2023) 110452, https://doi.org/10.1016/j.knosys.2023.110452.

[19] A. Ghasemi, A. Ashoori, C. Heavey, Evolutionary learning based simulation optimization for stochastic job shop scheduling problems, Appl. Soft Comput. 106 (2021) 107309, https://doi.org/10.1016/j.asoc.2021.107309.

[20] A. Ghasemi, F. Farajzadeh, C. Heavey, J. Fowler, C.T. Papadopoulos, Simulation optimization applied to production scheduling in the era of industry 4.0: A review and future roadmap, J. Ind. Inf. Integr. 39 (2024) 100599, https://doi.org/10.1016/j.jii.2024.100599.

[21] E. Ghaedy-Heidary, E. Nejati, A. Ghasemi, S.A. Torabi, A simulation optimization framework to solve stochastic flexible job-shop scheduling problems – Case: Semiconductor manufacturing, Comput. Oper. Res. 163 (2024) 106508, https://doi.org/10.1016/j.cor.2023.106508.

[22] V.H. Truong, S. Tangaramvong, G. Papazafeiropoulos, An efficient LightGBM-based differential evolution method for nonlinear inelastic truss optimization, Expert Syst. Appl. 237 (2024) 121530, https://doi.org/10.1016/j.eswa.2023.121530.

[23] E.V.W. Trentini, G.A. Parsekian, T.N. Bittencourt, Multiobjective optimization of bridge and viaduct design: comparative study of metaheuristics and parameter calibration, Eng. Struct. 312 (2024) 118252, https://doi.org/10.1016/j.engstruct.2024.118252.

[24] O. Contreras-Bejarano, J.D. Villalba-Morales, On the use of the differential evolution algorithm for truss-type structures optimization, Appl. Soft Comput. 161 (2024) 111372, https://doi.org/10.1016/j.asoc.2024.111372.

[25] Y. Jin, T. Wang, T. Liu, T. Yang, S. Dowden, A. Neogi, N.B. Dahotre, Gradient process parameter optimization in additive friction stir deposition of aluminum alloys, Int. J. Mach. Tools Manuf. 195 (2024) 104113, https://doi.org/10.1016/j.ijmachtools.2023.104113.

[26] A.M.C. Baek, E. Park, M. Seong, J. Koo, I.D. Jung, N. Kim, Multi-objective robust parameter optimization using the extended and weighted k-means (EWK-means) clustering in laser powder bed fusion (LPBF), Expert Syst. Appl. 236 (2024) 121349, https://doi.org/10.1016/j.eswa.2023.121349.

[27] D. Pendokhare, S. Chakraborty, A comparative analysis of preying behavior-based metaheuristic algorithms for optimization of laser beam drilling processes, Decis. Anal. J. 10 (2024) 100412, https://doi.org/10.1016/j.dajour.2024.100412.

[28] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, https://doi.org/10.1109/4235.585893.

[29] Y. Kim, J. Park, K. Na, H. Yuan, B.D. Youn, C.S. Kang, Phase-based time domain averaging (PTDA) for fault detection of a gearbox in an industrial robot using vibration signals, Mech. Syst. Signal Process. 138 (2020) 106544, https://doi.org/10.1016/j.ymssp.2019.106544.

[30] D. Anastasopoulos, E.P.B. Reynders, S. François, G. De Roeck, G. Van Lysebetten, P. Van Itterbeeck, N. Huybrechts, Vibration-based monitoring of an FRP footbridge with embedded fiber-Bragg gratings: influence of temperature vs. damage, Compos. Struct. 287 (2022) 115295, https://doi.org/10.1016/j.compstruct.2022.115295.

[31] Y. He, J. Chen, X. Zhou, S. Huang, In-situ fault diagnosis for the harmonic reducer of industrial robots via multi-scale mixed convolutional neural networks, J. Manuf. Syst. 66 (2023) 233–247, https://doi.org/10.1016/j.jmsy.2022.12.001.

[32] M. Mousavi, M. Alzgool, B. Davaji, S. Towfighian, Event-driven MEMS vibration sensor: integration of triboelectric nanogenerator and low-frequency switch, Mech. Syst. Signal Process. 187 (2023) 109921 https://doi.org.remotexs.ntu.edu.sg/10.1016/j.ymssp.2022.109921.

[33] T.T.V. Nguyen, H.D. Le, H.C. Hsu, C.N. Nguyen, C.C. Chiang, A medium-high frequency FBG accelerometer based on a V-shaped flexible hinge, Measurement 224 (2024) 113865, https://doi.org/10.1016/j.measurement.2023.113865.

[34] Z.H. Zhan, J.Y. Li, J. Zhang, Evolutionary deep learning: A survey, Neurocomputing 483 (2022) 42–58, https://doi.org/10.1016/j.neucom.2022.01.099.

[35] Y. An, C. Zhang, X. Zheng, Knowledge reconstruction assisted evolutionary algorithm for neural network architecture search, Knowl. -Based Syst. 264 (2023) 110341, https://doi.org/10.1016/j.knosys.2023.110341.

[36] L. Wen, L. Gao, X. Li, H. Li, A new genetic algorithm based evolutionary neural architecture search for image classification, Swarm Evol. Comput. 75 (2022) 101191, https://doi.org/10.1016/j.swevo.2022.101191.

[37] M. Suganuma, M. Kobayashi, S. Shirakawa, T. Nagao, Evolution of deep convolutional neural networks using cartesian genetic programming, Evol. Comput. 28 (1) (2020) 141–163, https://doi.org/10.1162/evco_a_00253.

[38] B. Lyu, S. Wen, K. Shi, T. Huang, Multiobjective reinforcement learning-based neural architecture search for efficient portrait parsing, IEEE Trans. Cybern. 53 (2) (2023) 1158–1169, https://doi.org/10.1109/TCYB.2021.3104866.

[39] F.F.E. Junior, G.G. Yen, Particle swarm optimization of deep neural networks architectures for image classification, Swarm Evol. Comput. 49 (2019) 62–74, https://doi.org/10.1016/j.swevo.2019.05.010.

[40] Y. Wang, X. Qiao, G.G. Wang, Architecture evolution of convolutional neural network using monarch butterfly optimization, J. Ambient Intell. Human. Comput. 14 (2023) 12257–12271, https://doi.org/10.1007/s12652-022-03766-4.

[41] H. Chen, Z. Zhang, W. Yin, C. Zhao, F. Wang, Y. Li, A study on depth classification of defects by machine learning based on hyper-parameter search, Measurement 189 (2022) 110660, https://doi.org/10.1016/j.measurement.2021.110660.

[42] S. Kaur, H. Aggarwal, R. Rani, Hyper-parameter optimization of deep learning model for prediction of Parkinson's disease, Mach. Vis. Appl. 31 (2020) 32, https://doi.org/10.1007/s00138-020-01078-1.

[43] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (10) (2012) 281–305. ⟨https://jmlr.org/papers/v13/bergstra12a.html⟩.

[44] R.J. Kuo, T.H. Chiu, Hybrid of jellyfish and particle swarm optimization algorithm-based support vector machine for stock market trend prediction, Appl. Soft Comput. 154 (2024) 111394, https://doi.org/10.1016/j.asoc.2024.111394.

[45] X. Li, J. Song, L. Yang, H. Li, S. Fang, Source term inversion coupling kernel principal component analysis, whale optimization algorithm, and backpropagation neural networks (KPCA-WOA-BPNN) for complex dispersion scenarios, Prog. Nucl. Energy 171 (2024) 105171, https://doi.org/10.1016/j.pnucene.2024.105171.

[46] X. Wang, Y. Zhao, Z. Wang, N. Hu, An ultrafast and robust structural damage identification framework enabled by an optimized extreme learning machine, Mech. Syst. Signal Process. 216 (2024) 111509, https://doi.org/10.1016/j.ymssp.2024.111509.

[47] Y. Wang, H. Zhang, G. Zhang, cPSO-CNN: An efficient PSO-based algorithm for fine-tuning hyper-parameters of convolutional neural networks, Swarm Evol. Comput. 49 (2019) 114–123, https://doi.org/10.1016/j.swevo.2019.06.002.

[48] R. Mohakud, R. Dash, Designing a grey wolf optimization based hyper-parameter optimized convolutional neural network classifier for skin cancer detection, J. King Saud. Univ. Comput. Inf. Sci. 34 (8) (2022) 6280–6291, https://doi.org/10.1016/j.jksuci.2021.05.012.

[49] A. Jain, A.C.S. Rao, P.K. Jain, Y.C. Hu, Optimized levy flight model for heart disease prediction using CNN framework in big data application, Expert Syst. Appl. 223 (2023) 119859, https://doi.org/10.1016/j.eswa.2023.119859.

[50] S. Kanwal, I. Younas, M. Bashir, Evolving convolutional autoencoders using multi-objective particle swarm optimization, Comput. Electr. Eng. 91 (2021) 107108, https://doi.org/10.1016/j.compeleceng.2021.107108.

[51] M. Shi, Y. Tang, X. Zhu, Y. Huang, D. Wilson, Y. Zhuang, J. Liu, Genetic-GNN: Evolutionary architecture search for graph neural networks, Knowl. -Based Syst. 247 (2022) 108752, https://doi.org/10.1016/j.knosys.2022.108752.

[52] T. Hassanzadeh, D. Essam, R. Sarker, EvoDCNN: An evolutionary deep convolutional neural network for image classification, Neurocomputing 488 (2022) 271–283, https://doi.org/10.1016/j.neucom.2022.02.003.

[53] A. Martín, A. Hernández, M. Alazab, J. Jung, D. Camacho, Evolving generative adversarial networks to improve image steganography, Expert Syst. Appl. 222 (2023) 119841, https://doi.org/10.1016/j.eswa.2023.119841.

[54] Y. Chen, Q. Lin, W. Wei, J. Ji, K.C. Wong, C.A.C. Coello, Intrusion detection using multi-objective evolutionary convolutional neural network for internet of things in fog computing, Knowl. -Based Syst. 244 (2022) 108505, https://doi.org/10.1016/j.knosys.2022.108505.

[55] J. Huang, B. Xue, Y. Sun, M. Zhang, G.G. Yen, Particle swarm optimization for compact neural architecture search for image classification, IEEE Trans. Evol. Comput. 27 (5) (2023) 1298–1312, https://doi.org/10.1109/TEVC.2022.3217290.

[56] K.M. Ang, E.S.M. El-kenawy, A.A. Abdelhamid, A. Ibrahim, A.H. Alharbi, D. S. Khafaga, S.S. Tiang, W.H. Lim, Optimal design of convolutional neural network architectures using teaching-learning-based optimization for image classification, Symmetry 14 (11) (2022) 2323, https://doi.org/10.3390/sym14112323.

[57] M. Karthiga, V. Santhi, S. Sountharrajan, Hybrid optimized convolutional neural network for efficient classification of ECG signals in healthcare monitoring, Biomed. Signal Process. Control 76 (2022) 103731, https://doi.org/10.1016/j.bspc.2022.103731.

[58] Q. Liu, F. Wang, M. Liu, W. Xiao, A two-step localization method using wavelet packet energy characteristics for low-velocity impacts on composite plate structures, Mech. Syst. Signal Process. 188 (2023) 110061, https://doi.org/10.1016/j.ymssp.2022.110061.

[59] Q. Fan, Z.A. Jia, D. Feng, Z. Yong, Highly sensitive FBG pressure sensor based on square diaphragm, Optik 225 (2021) 165559, https://doi.org/10.1016/j.ijleo.2020.165559.

[60] Q. Liu, M. Liu, F. Wang, W. Xiao, A dynamic stochastic search algorithm for high-dimensional optimization problems and its application to feature selection, Knowl. -Based Syst. 244 (2022) 108517, https://doi.org/10.1016/j.knosys.2022.108517.

[61] W. Zhang, C. Li, G. Peng, Y. Chen, Z. Zhang, A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load, Mech. Syst. Signal Process. 100 (2018) 439–453, https://doi.org/10.1016/j.ymssp.2017.06.022.

[62] H. Wang, W. Wu, T. Chen, X. Dong, G. Wang, An improved neural network for TOC, S1 and S2 estimation based on conventional well logs, J. Petrol. Sci. Eng. 176 (2019) 664–678, https://doi.org/10.1016/j.petrol.2019.01.096.

[63] P. Rai, N.D. Londhe, R. Raj, Fault classification in power system distribution network integrated with distributed generators using CNN, Electr. Power Syst. Res. 192 (2021) 106914, https://doi.org/10.1016/j.epsr.2020.106914.

[64] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 2010, pp. 807-814, ⟨https://dl.acm.org/doi/10.5555/3104322.3104425⟩.

[65] D.A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), 2015, arXiv preprint arXiv: 1511.07289.

[66] D. Hendrycks, K. Gimpel, Gaussian Error Linear Units (Gelus), 2016, arXiv preprint arXiv:1606.08415.

[67] I. Sergey, S. Christian, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the Thirty Second International Conference on Machine Learning, Lille, France, 2015, 448-456, ⟨https://proceedings.mlr.press/v37/ioffe15.html⟩.

[68] W. Huang, J. Cheng, Y. Yang, G. Guo, An improved deep convolutional neural network with multi-scale information for bearing fault diagnosis, Neurocomputing 359 (2019) 77–92, https://doi.org/10.1016/j.neucom.2019.05.052.

[69] G. Sun, Y. Gao, Y. Xu, W. Feng, Data-driven fault diagnosis method based on second-order time-reassigned multisynchrosqueezing transform and evenly mini-batch training, IEEE Access 8 (2020) 120859–120869, https://doi.org/10.1109/ACCESS.2020.3006152.

[70] M. Goldstein, S. Uchida, A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data, PLoS ONE 11 (4) (2016) e0152173, https://doi.org/10.1371/journal.pone.0152173.

[71] T.W. Rauber, A.L. da Silva Loca, F. de Assis Boldt, A.L. Rodrigues, F.M. Varejão, An experimental methodology to evaluate machine learning methods for fault diagnosis based on vibration signals, Expert Syst. Appl. 167 (2021) 114022, https://doi.org/10.1016/j.eswa.2020.114022.

[72] S. Gao, L. Xu, Y. Zhang, Z. Pei, Rolling bearing fault diagnosis based on SSA optimized self-adaptive DBN, ISA Trans. 128 (2022) 485–502, https://doi.org/10.1016/j.isatra.2021.11.024.

[73] J. Shen, F. Xu, Method of fault feature selection and fusion based on poll mode and optimized weighted KPCA for bearings, Measurement 194 (2022) 110950, https://doi.org/10.1016/j.measurement.2022.110950.

[74] J. Huang, B. Xue, Y. Sun, M. Zhang, A flexible variable-length particle swarm optimization approach to convolutional neural network architecture design, in: 2021 IEEE Congress on Evolutionary Computation, Kraków, Poland, 2021, pp. 934-941, https://doi.org/10.1109/CEC45853.2021.9504716.

[75] M. Lin, Q. Chen, S. Yan, Network in Network, 2013, arXiv preprint arXiv: 1312.4400.

[76] K.L. Keung, C.K.M. Lee, L. Xia, C. Liu, B. Liu, P. Ji, A cyber-physical robotic mobile fulfillment system in smart manufacturing: the simulation aspect, Robot. Comput. Integr. Manuf. 83 (2023) 102578, https://doi.org/10.1016/j.rcim.2023.102578.

[77] M. Yumnam, D. Ghosh, H. Gupta, Empirical mode decomposition based techniques for imaging of shallow delamination in concrete using impact echo, Mech. Syst. Signal Process. 184 (2023) 109668, https://doi.org/10.1016/j.ymssp.2022.109668.

[78] D. Singh, B. Singh, Investigating the impact of data normalization on classification performance, Appl. Soft Comput. 97 (2020) 105524, https://doi.org/10.1016/j.asoc.2019.105524.

[79] Y. Liu, H. Xiang, Z. Jiang, J. Xiang, Second-order transient-extracting S transform for fault feature extraction in rolling bearings, Reliab. Eng. Syst. Saf. 230 (2023) 108955, https://doi.org/10.1016/j.ress.2022.108955.

[80] K. Zhang, P. Chen, M. Yang, L. Song, Y. Xu, The Harmogram: a periodic impulses detection method and its application in bearing fault diagnosis, Mech. Syst. Signal Process. 165 (2022) 108374, https://doi.org/10.1016/j.ymssp.2021.108374.

[81] S. Yan, H. Shao, Y. Xiao, B. Liu, J. Wan, Hybrid robust convolutional autoencoder for unsupervised anomaly detection of machine tools under noises, Robot. Comput. -Integr. Manuf. 79 (2023) 102441, https://doi.org/10.1016/j.rcim.2022.102441.

[82] S. Zavrak, M. İskefiyeli, Anomaly-based intrusion detection from network flow features using variational autoencoder, IEEE Access 8 (2020) 108346–108358, https://doi.org/10.1109/ACCESS.2020.3001350.

[83] C. Cai, B. Gou, M. Khishe, M. Mohammadi, S. Rashidi, R. Moradpour, S. Mirjalili, Improved deep convolutional neural networks using chimp optimization algorithm for Covid19 diagnosis from the X-ray images, Expert Syst. Appl. 213 (2023) 119206, https://doi.org/10.1016/j.eswa.2022.119206.

[84] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems, Comput. Aided Des. 43 (3) (2011) 303–315, https://doi.org/10.1016/j.cad.2010.12.015.

[85] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowl. -Based Syst. 89 (2015) 228–249, https://doi.org/10.1016/j.knosys.2015.07.006.

[86] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization, Neural Comput. Appl. 27 (2016) 495–513, https://doi.org/10.1007/s00521-015-1870-7.

[87] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, https://doi.org/10.1016/j.advengsoft.2013.12.007.

[88] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67, https://doi.org/10.1016/j.advengsoft.2016.01.008.

[89] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: a bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (2017) 163–191, https://doi.org/10.1016/j.advengsoft.2017.07.002.

[90] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1) (1940) 86–92, https://doi.org/10.1214/aoms/1177731944.

[91] P.B. Nemenyi, Distribution-free multiple comparisons (Ph.D. thesis), Princeton University, 1963.

[92] J. Tang, Z. Chen, A.W.C. Fu, D.W. Cheung, Enhancing effectiveness of outlier detections for low density patterns, in: M.S. Chen, P.S. Yu, B. Liu (Eds.), Advances in Knowledge Discovery and Data Mining, Springer, Berlin, Heidelberg, 2002, pp. 535–548, https://doi.org/10.1007/3-540-47887-6_53.

[93] F.T. Liu, K.M. Ting, Z.H. Zhou, Isolation forest, in: Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, Italy, 2008, 413-422, https://doi.org/10.1109/ICDM.2008.17.

[94] M.M. Breunig, H.P. Kriegel, R.T. Ng, J. Sander, LOF: Identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, TX, USA, 2000, 93-104, https://doi.org/10.1145/342009.335388.

[95] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471, https://doi.org/10.1162/089976601750264965.

[96] N. Ma, X. Zhang, H.T. Zheng, J. Sun, ShuffleNet V2: practical guidelines for efficient CNN architecture design, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), Computer Vision - ECCV 2018, Springer, Cham, 2018, pp. 122–138, https://doi.org/10.1007/978-3-030-01264-9_8.

[97] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, Inception-ResNet and the impact of residual connections on learning, in: Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, CA, USA (2017) 4278–4284, https://doi.org/10.1609/aaai.v31i1.11231.

[98] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5 MB model size, 2016, arXiv preprint arXiv:1602.07360.

[99] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the Thirty First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016, 770-778, https://doi.org/10.1109/CVPR.2016.90.

[100] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.

[101] K. Lin, J. Pan, Y. Xi, Z. Wang, J. Jiang, Vibration anomaly detection of wind turbine based on temporal convolutional network and support vector data description, Eng. Struct. 306 (2024) 117848, https://doi.org/10.1016/j.engstruct.2024.117848.

[102] A. Ghasemi, R. Azzouz, G. Laipple, K.E. Kabak, C. Heavey, Optimizing capacity allocation in semiconductor manufacturing photolithography area – case study: Robert Bosch, J. Manuf. Syst. 54 (2020) 123–137, https://doi.org/10.1016/j.jmsy.2019.11.012.

[103] F. Falchi, M. Girardi, G. Gurioli, N. Messina, C. Padovani, D. Pellegrini, Deep learning and structural health monitoring: temporal fusion transformers for anomaly detection in masonry towers, Mech. Syst. Signal Process. 215 (2024) 111382, https://doi.org/10.1016/j.ymssp.2024.111382.