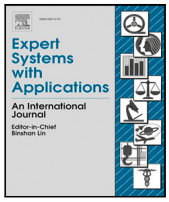
Expert Systems With Applications 233 (2023) 120678

Contents lists available at ScienceDirect

Expert Systems With Applications

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



An evolutionary ensemble convolutional neural network for fault diagnosis problem

Mohammad Hassan Tayarani Najaran

University of Hertfordshire, Hatfield, UK

ARTICLE INFO

Keywords:
Fault diagnosis
Evolutionary algorithms
Convolutional Neural Networks
Deep learning
Evolving deep learning

ABSTRACT

Automatic fault diagnosis in many systems is performed via the analysis of vibration data in the time or frequency domain. In the literature, many approaches for extracting featured from signals for fault diagnosis have been proposed. In this paper, we apply a variety of transform functions including Fourier, Wavelet, etc. to extract features from the vibration data, which includes the statistical features and some features automatically extracted via Convolutional Neural Networks (CNNs). For each of these feature extraction approaches, a learning algorithm is trained to diagnose the faults and their results are aggregated in an ensemble machine learning algorithm. The weights of the base learner algorithms are optimized via an evolutionary algorithm to achieve the best-weighted voting scheme. The architecture of CNNs has a significant effect on the performance of the algorithm, thus, in this paper, an evolutionary algorithm is proposed to find the best architecture for CNNs in fault diagnosis. The CNNs are trained via gradient descent algorithms which suffer from getting stuck in local optima. To manage this, we propose an evolutionary algorithm that benefits from the speed of gradient descent and the global search of evolutionary algorithms. The proposed algorithm is tested on a number of benchmark problems and the experimental results are presented.

1. Introduction

Fault diagnosis in many systems is usually performed by analyzing vibration data. The data are usually processed in time-domain or frequency-domain (Haidong, Junsheng, Hongkai, Yu, & Zhantao, 2020; Liu, Mu, Chen, Li, & Guo, 2020; Lu, Yan, Liu and Wang, 2019). Because the time-domain data better preserve the basic characteristics of the signal, many research extract features like pulse, kurtosis, peak, and waveform from the time-domain. However, there are some difficulties with these features, as some are suitable for some type of faults, while others may be suitable for other types (He, Shao, Zhang, Cheng, & Yang, 2019; Hu, Qin, Zhang, He, & Sun, 2018; Zhao, Liu, Xu, & Deng, 2019). Thus, there have been attempts to discover new features for fault diagnosis. Many research use frequency-domain data to extract features, example of which includes Fourier transform (Burriel-Valencia, Puche-Panadero, Martinez-Roman, Sapena-Bano, & Pineda-Sanchez, 2017; Zhang, Wang, & Wang, 2013), Wavelets transform (Lou & Loparo, 2004; Yan, Gao, & Chen, 2014), Hilbert–Huang transform (Rai & Mohanty, 2007; Wang, Ma, Zhu, Liu, & Zhao, 2014), generalized synchrosqueezing transform (Li & Liang, 2012), Walsh transform (Xiang, Zhou, Li, Li, & Luo, 2009), Cosine transform (Lindu & Zhaohan, 1996), etc.

Despite their success, traditional fault diagnosis methods require manually extracting features that demand professional prior knowledge. This reduces the flexibility of the produced models which restricts its application to particular equipment. In this sense, efforts to design automated feature extraction techniques that do not suffer from these deficiencies have been conducted by many researchers. Convolutional Neural Networks (CNNs) have attracted the attention of many works as they have been successful in the design of automatic feature extraction methods. Although promising, the use of CNNs in extracting features for fault diagnosis is in its infancy and requires more attention.

In Huang, Cheng, Yang, and Guo (2019), an improved CNN called multi-scale cascade CNN is proposed which bears at its input a new layer that constructs a new signal of more distinguishable information. After this layer, are inserted the convolutional layers. In Guo, Chen, and Shen (2016), a hierarchical learning rate-adaptive deep CNN is proposed which consists of a fault pattern determination layer and a fault size evaluation layer. In order to manage the dynamic information in raw data, Long Short-Term Memory (LSTM) neural network has been used in Zhao, Sun, and Jin (2018) for fault diagnosis. A novel method called deep decoupling CNN is proposed in Huang, Liao, Zhang, and Li (2018), which employs a 1-D CNN to learn the features from the raw signal. Then multi-stack capsules are designed to perform as decoupling classifiers. The fault signals are 1-D signals, while CNNs are more suitable for 2-D image signals. To manage this, a method is proposed in Lu, Lin et al. (2019) which transforms the 1-D signal into

E-mail address: m.tayaraninajaran@herts.ac.uk.

https://doi.org/10.1016/j.eswa.2023.120678
Received 16 September 2021; Received in revised form 31 May 2023; Accepted 31 May 2023
Available online 23 June 2023

a 2-D graph. The features of the resulting signal are then extracted via CNNs.

In some works, mathematical transform methods with CNNs have been used in fault diagnosis. Wavelet transform is used in Han, Tang, and Deng (2018), to construct multi-level wavelet coefficient matrices which are fed to several parallel CNNs. The output of these CNNs is then aggregated via an ensemble scheme. The authors argue that the parallel design helps the algorithm deal with the over-fitting problem. In Zhu, Peng, Chen, and Gao (2019), a short-time Fourier transform is applied to the signal to transform it into two-dimensional graphs. The resulting graph is then fed to CNNs to extract features.

CNNs usually require large sets of training data to prevent over-fitting. To manage this, a CNN is proposed in Cao, Zhang, and Tang (2018) which consists of a pre-trained CNN that extracts the features and a fully connected part for classification. Labeling large data-sets requires huge efforts. To overcome this, an unsupervised Categorical Adversarial Autoencoder is proposed in Liu et al. (2018). Other examples of the research that use CNNs to extract features for fault diagnosis can be found in Azamfar, Singh, Bravo-Imaz, and Lee (2020), Hoang and Kang (2019), Jiao, Zhao, Lin, and Zhao (2018), Jing, Zhao, Li, and Xu (2017), Wu, Jiang, Ding, Feng, and Chen (2019) and Wen, Li, Gao, and Zhang (2017).

By providing diversity and accuracy, ensemble learning improves the performance of learning algorithms (Kuncheva, 2005). Research shows that the ensemble of learning algorithms usually perform better compared to individual algorithms (Benediktsson, Sveinsson, Ersoy, & Swain, 1997; Breiman, 1996b; Hansen & Salamon, 1990; Hashem, 1997). Two main approaches of Heterogeneous and Homogeneous are employed in the literature to achieve diversity. Heterogeneous approaches achieve diversity by employing different classifiers. In order to reach diversity, the ensemble should consist of a set of classifiers that misclassify different instances (Polikar, 2006). The diversity in Homogeneous approaches is achieved by injecting randomness into the training phase of the classification. This can be done by, for example, manipulating the feature set. Examples of Homogeneous methods include Bagging (Breiman, 1996a), random subspace (Ho, 1998) and diversification of algorithm parameters (Ranawana & Palade, 2006). Bagging methods create diversity by changing the distribution of the training data and building different training sets. In random subspace approaches, the diversity is achieved by randomly selecting subsets of features, and diversification of algorithm parameters makes the diversity by incorporating diversity within the learning algorithms.

In Bühlmann, Yu, et al. (2002) and Buja and Stuetzle (2006) theoretical frameworks for bagging have been presented. A variation of Bagging, called *Pasting small votes* is designed in Breiman (1999) for large data-sets. Two versions of the algorithm called *Rvotes* and *Ivotes* were then proposed. In *Rvotes* the data subsets are created at random and in *Ivotes* consecutive data-sets are created based on important data instances. The important data instances are the data records that promote diversity. In Bryll, Gutierrez-Osuna, and Quek (2003) a method called attribute bagging was proposed that, via a wrapper scheme, establishes an appropriate attribute subset size. Bootstrap sampling with more advanced methods of feature selection was integrated in Stefanowski (2007). The feature selection methods in the research are performed based on an analysis of the relationship between the features and the target class. To achieve diversity, a method is proposed in Shirai, Kudo, and Nakamura (2008) which selects both the samples and features at the same time. In order to improve the performance of bagging, a method is proposed in Cai, Peng, and Zhang (2008) which considers the diversity of classification margins in feature sub-spaces. To do so, the task is converted into the optimization problem of finding the best weight of feature sub-spaces. In Cruz et al. (2021) an ensemble learning algorithm is proposed which consists of five learning algorithms that are aggregated in a voting scheme. The weights the learning algorithms in the voting scheme is optimized via a search algorithm. A two-stage ensemble of deep CNNs is presented in Uddamvathanak et al. (2018), in which for each base-learner, multiple rounds of training are preformed based on sub-sampling the training dataset. The output of the base-learners are aggregated via the MinMax median.

In order to detect fault in rotary systems many sets of features, including wavelet, Fourier, Hilbert–Huang, etc. have been proposed in the literature, and all these methods have shown successful results. However, there is not many research that try to employ all representative features in one system. To cover this, in this paper, we propose a method to take advantage of all these feature sets in an ensemble learning paradigm. In the proposed algorithm, for each set of these features, a base learner algorithm is designed to diagnose the faults. Then, an ensemble scheme is proposed to aggregate the output of these base learners. The proposed ensemble learning is a weighted voting scheme. Finding the optimal weight of each base-learner in the voting system is an optimization problem which highly affects the performance of the final algorithm. The proposed algorithm employs an evolutionary algorithm that finds the optimal voting weights. Most of the research in the literature, employ statistical features for diagnosis. In this paper, apart from statistical features that are used in the literature, we also use CNNs to automatically extract features from the raw and transformed signals. These new sets of features which have not been studied before show promising results.

The architecture of CNNs has a great deal of effect on their performance so finding the optimal architecture is a matter of importance for these algorithms. Finding the optimum architecture for a CNN is an optimization process which is usually performed manually by experts. There are many architectures proposed in the literature. Some well-known works include CAE-2 (Rifai, Vincent, Muller, Glorot, & Bengio, 2011), TIRBM (Sohn & Lee, 2012), PGBM+DN-1 (Sohn & Lee, 2012), ScatNet-2 (Bruna & Mallat, 2013), RandNet-2 (Chan et al., 2015), LDANet-2 (Chan et al., 2015), SVM+RBF (Larochelle, Erhan, Courville, Bergstra, & Bengio, 2007), SVM+Poly (Larochelle et al., 2007), NNet (Larochelle et al., 2007), SAAA-3 (Larochelle et al., 2007), SqweezNet (SQNet) (Iandola et al., 2016), MobileNetV2 (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018), DBN-3 (Larochelle et al., 2007), LSTM and the algorithms proposed in Baldominos, Saez, and Isasi (2018), Fernandes Junior and Yen (2019b), Gottapu and Dagli (2020), Ma, Li, Xia, and Zhang (2020) and Sun, Xue, Zhang, and Yen (2018). Manually optimizing the architecture has two limitations. First it requires an expert with good understanding of the problem and the learning algorithm. Second, the designed architecture is not generic and is tweaked to perform best for a given problem. In this respect, designing an automated method that can optimize the architecture for a given problem is a matter of importance. In many research, Evolutionary algorithms are shown to be successful in optimizing the architecture of machine learning algorithms. In order to optimize the architecture of CNNs in diagnosing the faults, an evolutionary algorithm is proposed in this paper. The architecture of a CNN is considered in this paper as the graphical architecture and the numerical parameters of the network. The evolutionary algorithm tries to optimize both the graphical architecture and numerical parameters of the CNNs.

Training CNNs involves the optimization of a large number of weighting parameters. Because of the large search space, the training is often performed via gradient-based algorithms which, due to their local search scheme, suffer from getting stuck in local optima. The search space in the training process, specially for deep learning algorithms is very large and incorporating a global search algorithm can improve the performance. To manage this, an evolutionary algorithm is proposed in this paper which uses the gradient descent algorithm to find the local optima in the search space. Then these good solutions are combined via evolutionary operators to generate new individuals. The new individuals are then optimized via gradient descent to find better local optima. The process continues until better solutions are found. This paradigm manages the local optima problem in gradient descent algorithms. In this scheme, the fast local search, the gradient descent has the role of finding the local optima and the evolutionary part has

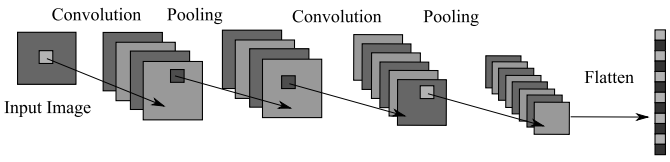


Fig. 1. The architecture of CNNs, consisting of a combination of convolution and pooling operators.

the role of performing the global search and helping the algorithm to escape from local optima. The proposed algorithm is tested on some benchmark problems and the results are presented.

The rest of this paper is organized as follows. Section 2 introduces Convolutional Neural Networks and the way they are used and their architecture is optimized in this paper. The proposed algorithm is introduced in Section 3. Section 4 performs experiments on some benchmark problems and finally, Section 5 concludes the paper.

2. Convolutional neural network structure

CNNs are a group of feed-forward artificial neural network algorithms that have successfully been applied to a wide range of pattern recognition problems. As presented in Fig. 1 a CNN consists of a number of convolutional and pooling layers. The CNN in this figure has two convolutional, two pooling, and one flattening layer. There are four groups of feature maps followed by a fully connected layer at the tail. The input of the CNN is a signal (usually a 2-D image signal) that is fed to the network and is processed in the layers. The convolutional and pooling layers process the input image and extract features. The features are then classified via the fully connected layers. To generate a feature map, a filter should be defined for the convolutional operators. The filter is a matrix that performs convolution by sliding over the image with the step size called stride. The pixels in the feature map are the dot product of the corresponding pixels in the filter and those in the image. The parameters of the convolutional operators are *connection weight*, *filter width*, *filter height*, *number of feature maps*, the *stride width* and the *stride height*.

Similar to the convolutional operators, the pooling operators process the image by traversing over the image. The operator performs computations, via a matrix called kernel to collect the average or maximum values in the previous layer. This operator is devised to streamline the underlying computations by reducing the size of the representations to reduce the computations, number of parameters, and the required memory. The pooling operator reduces the dimension of the data by combining the output of a number of neurons at one layer into a single neuron in the next layer. The parameters of the pooling operator are the *stride height*, *stride width*, *kernel height*, *kernel width* and *pooling type*.

CNNs have one or more fully connected layers at their tail. The fully connected layer performs high-level reasoning and classification. The neurons in a fully connected layer are connected to all the neurons in the previous layer.

The CNN in Fig. 1 consists of a particular ordering of the convolutional, pooling, and fully connected layers. This ordering, known as the CNN architecture greatly affects the performance of these algorithms. Also, the numerical parameters of the operators, namely kernel, filter, and stride size affect the performance. Therefore, when solving a problem, it is very important to use an architecture that is specifically suitable for the problem.

There are some works that use evolutionary algorithms to optimize the architecture of CNNs. In Sun, Xue, Zhang, and Yen (2020) a block-based Genetic Algorithm is proposed to optimize the architecture of CNNs. In order to improve the optimization of CNN architectures, in Xie, Chen, Ma, and Xu (2022) a triplet attention mechanism is incorporated into the architecture of the algorithm. This work uses a random forest-based performance predictor in the fitness evaluation process. In Bingham, Macke, and Miikkulainen (2020), a tree-based search space of activation functions is defined to optimize the activation functions of CNNs via evolutionary algorithms. In order to create a tradeoff between complexity and accuracy in CNNs, in Johner and Wassner (2019), an evolutionary method is presented to optimize the architecture of a CNN. The accurate prediction of the remaining useful life of industrial components is a crucial task in many companies. In Mo, Custode, and Iacca (2021) argues that many Deep Neural Networks (DNNs) have been proposed for the problem, and the search the best architecture of these algorithms is ongoing. To find the best architecture of DNNs, evolutionary algorithms are employed in this work that requires minimum computational resources. In Junior and Yen (2019a), particle swarm optimization algorithm is used to optimize the architecture of CNNs for image classification.

In this paper, we use a Genetic Algorithm to determine the best architecture for the fault diagnosis problem. In the proposed GA, the ordering of the operators and the numerical operators of CNN are optimized at the same time. Algorithm 1 is used in this paper for this purpose. Here the simple version of GA with the individual representation in Fig. 2 is used for the optimization. Similar approaches have been conducted before in the previous works that have been mentioned in this work.

Algorithm 1 CNN Architecture Optimization Algorithm

begin
  set the parameters m, n and l
  $\tau = 0$
1. initialize the population Y^0,
2. **while** not termination condition do
  begin
3. evaluates the individuals in Y^τ
4. select the parent solutions via tournament selection
5. generate offsprings via crossover and store them in O^τ
6. perform mutation on O^τ with probability m
7. perform environmental selection on $Y^\tau \cup O^\tau$ and store the results in $Y^{\tau+1}$
  $\tau = \tau + 1$
  end
8. **return** the best solution in Y^τ
end

In the beginning, the parameters of the algorithm, m the mutation rate, n the population size, and l the maximum length of the CNNs is set. The parameter l determines the maximum number of layers in the CNN. This controls the computational cost of the algorithm. Although a greater number of layers in a CNN does not always translate into better performance, having more layers means that the algorithm has more complexity and is capable of extracting more detailed features. On the other hand, larger CNNs require more computational power for both training and the classification phase. Therefore in setting l a trade-off should be performed between the computational cost and the required performance. Based on our computation/time budget, we set $l = 8$ in this paper.

In step 1 of the algorithm, the population is initialized. In this step, n individuals X^i, $i = 1, \ldots, n$ are generated at random. The number of layers in each individual must be 2 at a minimum because, in each CNN, the first layer is always convolutional and the last layer is always a fully connected layer. For each individual, a random number between 3 and l is generated which determines the number of layers in the CNN. Thus in

the initialization step, individuals with a different numbers of layers are generated. This allows the algorithm to search in the space of finding the best number of layers for the CNNs. A convolutional operator is placed at the first layer and a fully connected layer is inserted at the last layer. For the layers in between, with the probability of half a pooling or a convolutional layer is placed until the required number of layers is achieved.

The numerical parameters of the operators are also initialized at random. This is performed for the pooling and convolutional operators. For example, for the convolutional operators the parameters *connection weight*, *convolution type*, *filter width*, *filter height*, *number of feature maps*, the *stride width* and the *stride height* are randomly set. The values of the filter or kernel matrices are also set at random. All these values are optimized during the optimization process. For the sake of simplicity and reducing the size of the search space, all the convolutional layers in an individual have the same value for their numerical parameters. Due to the huge number of connection weights, the optimization of these values via the genetic algorithm is arduous work and so is performed during the training phase. However, the mean and standard deviation of the Gaussian distribution functions for generating these values are optimized via the GA.

After initialization, the while loop performs the GA until the termination condition is satisfied. The termination condition can be a maximum number of generations, a convergence measure, or the desired performance. In step 3 of the algorithm, the individuals are evaluated. In order to evaluate an individual, a CNN with the architecture suggested by the individual is generated, trained, and tested on the data records. Then the performance of the CNN in the classification process is taken as the fitness of the individual. The performance of the learning algorithm in terms of the accuracy is used as the fitness of a particular architecture. The fitness is measured as, Eq. (4).

The gradient descent algorithm is used in this paper to train the CNNs. Note that due to the huge number of weights that should be optimized in the training phase of the CNNs, using evolutionary algorithms is not practical and thus the gradient descent algorithm is the best choice. In step 4 of the algorithm, a simple tournament selection algorithm is used to select the individuals.

In step 5 of the algorithm, the crossover is performed on the selected individuals to generate offsprings. The crossover operator for structure optimization is presented in Fig. 2. In this figure, I stands for input layer, C stands for convolutional layer, P stands for pooling layer and F stands for fully connected layer. The crossover chooses randomly two crossover points on the two individuals. Then generates the offspring by selecting one part from the first parent and the second part from the second parent. Note that the size of the two-parent individuals is different and the crossover is flexible to combine and build different size individuals. In this paper, we define three types of mutation operators, each serving a specific role. The mutation operators are as follows.

- Choose one operator randomly and swap its type (if it is convolutional, change it to pooling and vice versa).
- Choose a pooling and convolutional operators and swap their position. This mutation explores to find the best ordering of the existing operators on CNN.
- Randomly remove or insert an operator in the CNN. By changing the number of layers in the CNN, this mutation explores to find the best number of layers for the algorithm.

These operators are designed to explore and find the best ordering of the operators within CNN. The architecture of CNNs in this paper is defined as the ordering of the operators and the numerical parameters of the operators. To optimize the numerical parameters of the operators, the following crossover and mutation operators are devised. The numerical parameters (*connection weight*, *filter width*, *filter height*, etc.) are coded as a vector of numbers, for which a simple one-point crossover is used in this paper. The numerical mutation is devised as randomly choosing one value in the parameter vector and changing it to a random value.

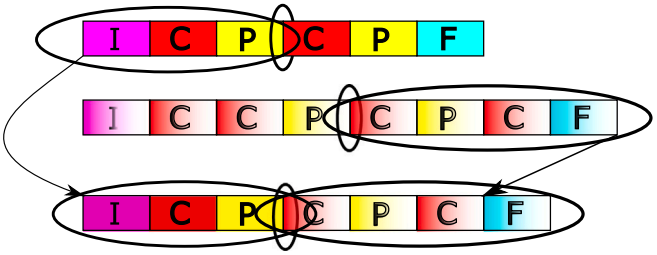


Fig. 2. The crossover operator for the CNN architecture optimization.

In this scheme, the individuals consist of two parts, one codes the architecture which is shown in Fig. 2 and one is the numerical parameters which are a vector of numbers. The individuals representing the architecture, as shown in Fig. 2, are flexible and can be of different sizes. The figure shows how this is managed in the crossover process. The numerical parameters on the other hand are of the same size among all the individuals, so performing crossover is simple. The number of filters in the architecture is fixed and the number of neurons is adaptively optimized via the architectural crossover operator as of Fig. 2.

Many different CNN architectures have been proposed in the literature. However, these architectures are tuned for particular problems or are designed for generic problems. We believe that there is no optimal architecture that fits all problems, and each problem requires its own architecture tuning.

3. The proposed algorithm

In the literature, different sets of features have been used to diagnose faults in systems. These include time-domain signal features like pulse, kurtosis, peak, etc. While these features have been successful in diagnosing faults (Ben Ali, Fnaiech, Saidi, Chebel-Morello, & Fnaiech, 2015), there are some properties of the faults that are not reflected in these features. Faults in systems result in the significant transient processes. The transient signals with some high-frequency components are usually representative of different types of faults. Time-domain features do not always reflect these properties and transforming the signals into the frequency domain is sometimes necessary. To capture this, Fourier transform has been used which is a very good choice for finding what frequencies occur in the signal (Tran, AlThobiani, Ball, & Choi, 2013). Although Fourier transform is very good at identifying the frequency components of the signals, they do not provide time–frequency information of the signals. In other words, Fourier transform tells what frequencies exist in the signal, but does not tell where in the signal these frequencies occur. To manage this, the wavelet transform has been proposed which provides time–frequency information about the signal. In this sense, using features from the wavelet transform has been successful in diagnosing faults. Although successful in extracting time–frequency information from the signals, wavelet transform has the shortcoming that the mother wavelet is difficult to be selected (Gawali, Hasabe, & Vaidya, 2015). The Hilbert–Huang transform is proposed to manage this problem. The HHT is suitable to analyze the nonlinear and non-stationary signals. The transform consists of two steps: the empirical mode decomposition (EMD) and Hilbert transform. The EMD decomposes the original signal into a number of intrinsic modes function components. The Hilbert transform is then applied to each of these components so the time–frequency of the signal is extracted. Apart from these signal processing methods, CNNs have been used as an automatic feature extraction method to extract features from the fault signals.

Each of these feature extraction methods has its advantages and shortcomings. We believe that combining these approaches can help the learning algorithms to take advantage of all these methods.

3.1. The proposed ensemble algorithm

In this paper, we propose an ensemble algorithm that is a combination of many learning algorithms, where each base learner algorithm is trained via a particular set of features. In other words, the diversity in the ensemble learning algorithm is achieved by using different sets of features for each learning algorithm. In this paper, the following methods are used to extract features from the signals.

- *Time-Domain Features*: The first base learner uses time-domain features to identify the faults. The features include Mean, Entropy Error, Entropy estimation, histogram lower, histogram upper, RMS, Kurtosis, Skewness, Peak to Peak, Crest Factor, Shape Factor, Impulse Factor, Margin Factor, Add Factor 1, and Add Factor 2. For a more detailed description and mathematical formula of these features please see Ben Ali et al. (2015).
- *Fourier Transform Features*: The Fourier transform is applied to the signal and the Frequency center, RMS variance frequency, and Root variance frequency are used for classification. A more detailed description of these features and the extraction process please see Tran et al. (2013).
- *Wavelet Transform Features*: To extract this set of features, DWT is applied to the signals, then the statistical features of the first six decomposition levels are extracted. The statistical features include Mean, Entropy Error, Entropy estimation, histogram lower, histogram upper, RMS, Kurtosis, Skewness, Peak to Peak, Crest Factor, Shape Factor, Impulse Factor, Margin Factor, Add Factor 1, and Add Factor 2. For more information on how to extract these wavelet features from a signal please see Hu, He, Zhang, and Zi (2007).
- *Walsh Transform*: The Walsh transform is applied to the signal and the statistical features of the resulting signal are used for classification. For more information on this transform, function please see Xiang et al. (2009).
- *Hilbert–Huang Transform*: The Hilbert–Huang Transform (HHT) is a combination of Empirical Mode Decomposition and Hilbert Transform and is a time–frequency analysis. The transform is specifically designed for non-linear and non-stationary data. HHT is applied to the signal and the statistical features are extracted. For more information please see Konar and Chattopadhyay (2015).

For all these signals, apart from the statistical features explained above, also CNNs are used to extract features. In order to extract features via CNN, from the time-domain features, the signal is first transformed into an image. Fig. 3 shows the process that converts one-dimensional signals into an image to be processed via a CNN. Note that in order for the CNNs to process the images, all the input images should be of the same size. In this method, the signal is sliced into M none overlapping pieces of size M. Then, in the second dimension, these pieces are placed on top of each other to build the $M \times M$ image. Because the size of the input signal is greater than M^2, the slices are chosen from random places in the signal. Also, because the signal is converted into an image, the values of the pixels should be in the range of [0–255]. Thus, the values of the signal should be normalized in this range.

In this paper, CNNs are used to extract features from the Fourier transform of the signals. Applying Short-Term Fourier Transform (STFT) allows extracting both time and frequency information of a signal. The result of an STFT of a signal is a two-dimensional signal that can be analyzed as an image. This image can be processed via CNNs to extract features. More information on this can be found in Zhang, Xing, Bai, Sun, and Meng (2020). The output of the wavelet transform on a one-dimensional signal is a two-dimensional signal that can be processed via CNNs. In this paper, we use the method proposed in Liang et al. (2019) to extract features via wavelet and a CNN.

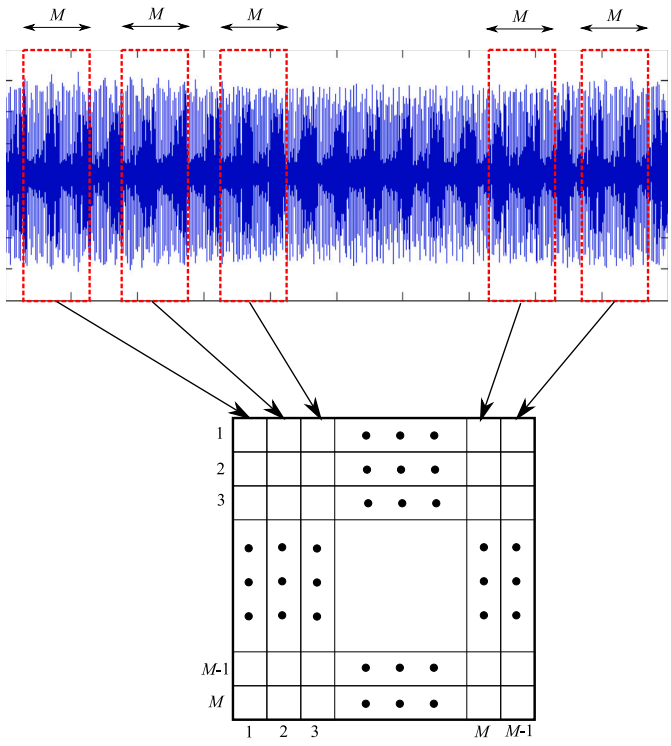


Fig. 3. The procedure to convert a signal to an image.

Extracting features from the Hilbert–Huang transform (HHT) via CNN is performed in this paper. We use the method presented in Guo, Yang, and Chen (2019) to extract these features.

The training in CNNs is the process of finding the best weights and the best filters and kernels for the network. In the literature, gradient-based algorithms are used to find the best values for these parameters. The advantage of gradient descent (GD) algorithms compared to an exhaustive or evolutionary search is that they are fast. This is specifically true for CNNs in which there is a huge number of parameters that should be optimized in the learning process. Despite this advantage, the GD algorithms are very prone to get stuck in local optima.

Fig. 4 represents the proposed ensemble algorithm. In this algorithm, 9 feature extraction methods are used and the features are fed to 9 classifiers. In this paper, we use Feed Forward Neural Networks (FFNN) as classifiers. For the CNN feature extractors, FFNN is used as the classifiers at the fully connected layers and the training is performed as back-propagation through the whole network.

The proposed approach collects a large number of features via different feature extraction methods. Instead of feeding all the features into a single learning algorithm, an ensemble approach is proposed where each base learner is trained via a subset of features. Using the proposed scheme has a number of advantages over using all the features in a single learning algorithm. (1) In many cases, feeding too many features to a learning algorithm results in poor performance and a feature selection mechanism should be adopted. The proposed approach has the advantage that each of the base learners is trained based on a smaller number of features. (2) Having a number of separate feature sets results in diversity which in turn allows us to use ensemble algorithms. This results in improved performance. (3) This scheme is composed of a number of independent learning algorithms that can be trained and tested in parallel. This allows implementing the algorithms in a distributed paradigm. (4) Because each of the base learners is architecturally designed and trained via a particular set of features, they become tuned to learn the patterns of the feature set. For example, one learning algorithm is assigned to learn only the patterns

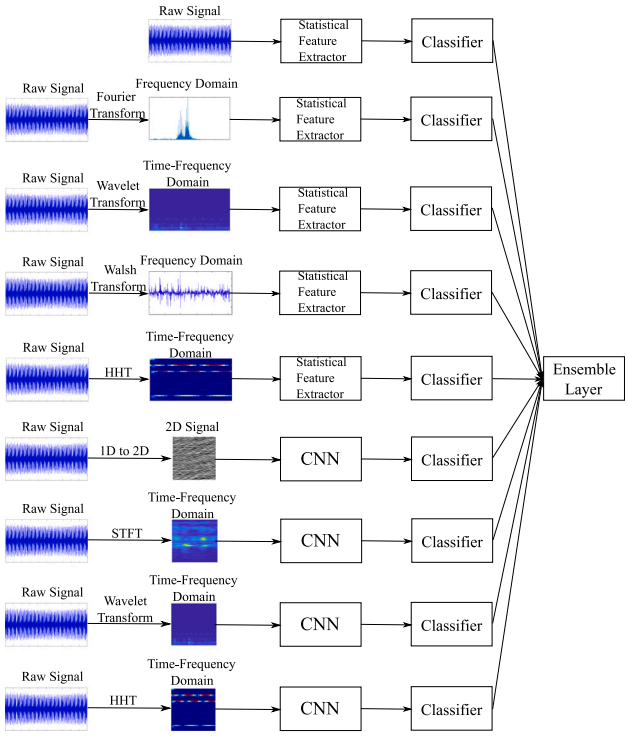


Fig. 4. The proposed ensemble algorithm.

in the wavelet coefficients of the signal, so the learning algorithm gets specialized in discovering these patterns. This learning algorithm is not required to discover the patterns in the statistical features of the signal. (5) The proposed paradigm splits the feature space into smaller pieces. Thus, the training process for each of the base learners consists of performing a search in a smaller search space with much fewer local optima. This increases the chance of the algorithm finding better optima.

3.2. Optimization of the weights in the voting scheme

The ensemble layer in this algorithm is weighted voting among the classifiers. These weight values should be adjusted to find the best classification. To do so, we first train all the base learners via the training data. Then the final output is found as

$$P = \frac{\sum_{i=1}^{c} w_i p_i}{\sum_{i=1}^{c} w_i}, \tag{1}$$

where P are the overall prediction of the ensemble, p_i is the prediction of ith base learner, c is the number of base learners and w_i is the weight of the ith base learner. In this paper, Genetic Algorithms are used to optimize the weight values (w_i) of the ensemble layer. The individuals in this algorithm are vectors of real numbers.

In the initialization step of GA, first the values of the weights are set to random numbers between (0,1) ($w_i = R(0,1)$ for $i = 1 \dots c$). Algorithm 2 shows the way the fitness of an individual is calculated. In order to measure the fitness of an individual, the training data are partitioned into 4 equal size partitions. Then in a 4-fold scheme, the base learners are trained and used to predict the output. The weight vector suggested by the individual x is used to aggregate the base learners via weighted voting. The total number of correctly classified data records is then used as the fitness value of the individual x. Note that in step 4 of the algorithm 2, because it is only the weights of the ensemble layers that are being optimized, the base learners do not need to be trained and tested every time the fitness function is called. The base learners are trained and tested in the 4-fold scheme once and their output is stored in a lookup table. Then, every time the fitness function is called, the table is used to find the output of the base learners.

Because the weights in the voting scheme should be positive numbers, in the optimization process, we set a threshold to make sure the weights do not go below zero.

The advantage of the proposed ensemble learning is that not only it considers a large set of features for classification which improves the performance, but also through the weighted voting scheme finds the features that are more likely to provide better results. In the proposed optimization algorithm for the voting scheme, if there is a set of features (statistical or extracted via a CNN) that do not perform well in predicting the final output, the weight of the classifier is automatically set to a small value, so it will have small or no role in the final classification. Conversely, if a set of features is better at predicting the output, the algorithm chooses the weight in a way to give a better role to the classifier. This simply means that the user does not need to worry about a particular feature extractor to perform well. The not good feature extractors are automatically eliminated from the voting in the proposed algorithm. Note, however, that we have carefully chosen and used the set of features that are shown in the literature or in our experiments to provide accurate results.

Algorithm 2 Ensemble Weight Optimization Fitness Function

Gets as input the individual x and returns the fitness
begin
1. partition the training data T into 4 equal partitions T_j, $j = 1 \dots 4$
2. $S = 0$
3. **for** $j = 1 \rightarrow 4$ do
   begin
4. train the base learners with $T - T_j$
5. test the base learners with T_j
6. use the weights in x to aggregate the base learners and store the number of correctly classified cases in s
7. $S = S + s$
   end
8. **return** S

end

3.3. Evolutionary training of CNNs

The training process in learning algorithms is and optimization process which is usually performed via gradient descent algorithm. The advantage of gradient descent algorithms is their speed, specially for training the learning algorithms in which there usually are a large number of training weights. However, just like any other local search algorithm, Gradient Descent algorithms are prone to getting stuck in local optima. This is specifically true in the case of CNNs, where there is a huge number of learning parameters (weights, filter values, etc.) that should be optimized. In such huge fitness landscapes, the chance of a local search algorithm to find the global optimum is slim. In this respect, introducing a global search algorithm into the process can improve the chance of the algorithm in finding the optimal solution. In order to design a training algorithm for CNNs that benefits from the advantages of gradient descent and evolutionary algorithms, in this paper, we propose an evolutionary algorithm that uses the gradient descent as a local search and evolutionary operators as the global search. In algorithm 3 the gradient descent algorithm is performed to find the local optima in the fitness landscape. The evolutionary operators are then applied to the local optima to perform a global search in the fitness landscape. A description of the algorithm is as follows.

Algorithm 3 The Proposed Evolutionary Training Algorithm

begin
set the parameters m, α, N and n
$\tau = 0$
1. initialize N sub-populations Y_j^0, $j = 1 \dots N$
2. **while** not termination condition do
begin
3. **for** $j = 1 \dots N$
begin
4. **for** all individuals in Y_j^τ
begin
5. partition the training data T into two partitions T_{train} and T_{test}
6. train the individual with T_{train} via GD
7. evaluates the individual with T_{test}
end
8. select the parent solutions from Y_j^τ via tournament selection
9. generate offsprings via crossover and store them in O_j^τ
10. perform mutation on O_j^τ with probability of m
11. perform environmental selection on $Y_j^\tau \cup O_j^\tau$ and store the results in $Y_j^{\tau+1}$
end
12. perform migration with probability of α
$\tau = \tau + 1$
end
13. **return** the best solution in Y_τ
end

At the beginning of the algorithm, the algorithm parameters, i.e. mutation rate m, the migration rate α, and the population size, n are determined.

In step 1 of the algorithm, the individuals are randomly initialized. Each individual represents a set of training parameters (i.e. connection weights and filters in convolutional layers) of the CNN and initialization is performed via randomly setting the learning parameters of the CNNs. CNNs usually have a large number of learning parameters that could be of the order of hundreds of thousands. To increase diversity in the population and help the algorithm not to get stuck in local optima we devise a multi-population evolutionary algorithm. The population in the proposed algorithm is divided into N sub-populations of equal size. In this paradigm, each sub-population explores the search space independently, so each will search a different area of the fitness landscape. This way, a wider area of the search space will be explored by the individuals.

In step 2 of the algorithm, the training process continues until the termination condition is reached. The termination condition can be a maximum number of generations, a predefined performance, convergence, or when the algorithm does not reach better results for a number of iterations. The *for* loop in step 3 performs operations on all sub-populations.

The individuals in the proposed evolutionary algorithm, represent a set of the training parameters of a CNN. At each generation of the evolutionary algorithm, the gradient descent is performed on the CNNs (individuals) and the networks are trained. In order to measure the fitness of the individuals, the performance of the corresponding CNNs in classification is estimated and used as the fitness. To estimate the performance of the CNNs, in step 5 of the algorithm, the training data are partitioned into two pieces of T_{train} and T_{test}. Here, T_{train} is used to train the individuals (CNNs) via gradient descent and T_{test} is used to test and evaluate their fitness. The partitioning is performed via a random paradigm, where 3/4 of the data in T are randomly selected and put in T_{train} and 1/4 of them are inserted in T_{test}.

Table 1
The range of each optimization parameter.

Parameter	Range
Connection weight	$[-\infty, \infty]$
Filter width	[2,8]
Filter height	[2,8]
# of feature maps	[4–24]
Stride width	[2,8]
Stride height	[2,8]
Kernel height	[2,8]
Kernel width	[2,8]
Pooling type	{max,min,avg}

The individuals are trained in step 6 of the algorithm on the data set T_{train}. The CNNs are trained via the gradient descent algorithm which starts from the learning parameters of the individual and perform the search until it reaches a local optimum. Note that in the first iteration of the genetic algorithm, the individuals have been initialized randomly, so the gradient descent starts the search from a random point in the search space. As the genetic algorithm progresses, the gradient descent optimizes the individuals that are the result of crossover and mutation operators. That is the gradient descent algorithm takes the individuals that are the result of GA operators as the starting point in its optimization process.

The individuals are evaluated in step 7 of the algorithm. The CNNs are trained on the data T_{train} and evaluated base on T_{test}. This process manages the over-fitting problem in two ways. First, the data T_{test} that is used to evaluate the individuals are not present in the gradient descent training phase. Second is that for each individual, the T_{train} and T_{test} are different. Because the data on which the individuals are evaluated, T_{test}, are different and set randomly for each individual, the global search algorithm is less prone to over-fitting (compared to using the same T_{test} for evaluating all the individuals).

In step 8 of the algorithm, the parent solutions are selected via tournament selection and in step 9 crossover is performed to generate new individuals. Note that the crossover operator is only applied to the individuals within the same sub-population. This scheme is devised to help the algorithm preserve diversity. The learning parameters in CNNs are the connection weights and the filters. The connection weights are a number of matrices and the crossover is defined in this paper as a simple two-point crossover on matrices. The filters are also matrices and a similar crossover is devised for them. In step 10, the mutation operator is applied to the individuals. Because the number of learning parameters is very large, the mutation operator should change a fair amount of parameters. The proposed mutation changes randomly 5% of the learning parameters in the matrices.

After the mutation and crossover operators are applied and a new generation of individuals are created, the new individuals are then optimized via gradient descent in step 6 of the algorithm. The crossover and mutation operators change the individuals, moving them away from the local optimum they have reached during the gradient descent. This gives the gradient descent another chance to find another local optimum. By randomly perturbing the individuals, the mutation operator injects diversity into the population. The crossover operator combines two individuals and produces an individual that inherits properties from both parents. The gradient descent then starts from this solution and performs a search until it reaches another local optimum. The new individual is located in a region in the search space in between its parents. Starting from this point, the gradient descent algorithm searches until it reaches a local optimum which, in expectation, has better fitness than its parents.

In step 12, the migration is performed, which selects at random two individuals from two sub-populations and swaps the sub-population to

Table 2
The accuracy of different algorithms (Eq. (4)) on the Tennessee Eastman benchmark (case 1). The data are averaged over 30 runs.

Algorithm	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	Cls 6	Cls 7	Cls 8	Cls 9	Cls 10	Avg
SVM	96.85	97.91	28.76	98.33	92.98	62.37	**99.96**	49.75	30.34	36.25	69.35
ANN	94.53	95.03	28.58	93.57	72.93	98.30	98.29	87.31	32.45	16.68	71.76
SAE	93.90	94.16	33.09	97.84	71.74	51.29	99.94	49.14	35.35	29.50	65.60
DBN	94.12	94.24	31.74	97.90	71.96	49.53	98.78	57.70	32.01	**40.72**	66.87
BLS	93.88	94.07	34.57	97.86	70.85	79.55	99.54	54.23	28.21	18.50	67.12
Raw Stat.	94.71	96.94	28.83	93.26	83.97	51.81	94.78	57.67	29.64	32.36	66.40
Fourier Stat.	93.77	93.75	34.80	95.39	83.71	54.96	87.69	53.05	35.07	36.44	66.86
Wavelet Stat.	93.72	95.44	30.54	93.28	74.44	39.37	80.87	52.81	30.38	29.99	62.08
Walsh Stat.	96.21	98.32	31.72	95.99	93.43	74.90	91.11	77.60	31.58	36.95	72.78
HHT Stat.	96.29	98.06	28.18	94.88	91.96	53.21	94.87	71.05	33.43	27.67	68.96
1D to 2D CNN	94.99	98.50	33.53	97.11	79.74	70.29	89.69	76.75	31.17	31.35	70.31
STFT CNN	94.76	96.83	30.28	97.29	89.56	70.16	88.92	56.90	34.41	26.23	68.53
Wavelet CNN	96.52	98.51	32.71	96.18	91.27	53.53	98.84	81.50	**37.66**	37.71	72.44
HHT CNN	94.56	98.06	34.48	98.02	90.69	70.64	97.71	65.25	31.93	23.78	70.51
EECNN	**98.14**	**99.88**	32.32	**99.66**	**94.61**	**99.77**	99.88	**88.84**	34.52	38.65	**78.63**
1D to 2D CNN_	95.00	98.25	**35.65**	97.25	78.25	69.00	89.45	70.10	35.60	26.60	69.51
STFT CNN_	94.70	96.00	31.65	96.35	86.25	66.60	88.05	55.50	32.30	24.45	67.19
Wavelet CNN_	96.20	97.55	33.55	95.75	86.65	50.60	97.30	79.05	35.05	31.35	70.31
HHT CNN_	94.35	96.70	29.75	97.05	85.95	69.35	96.90	65.10	33.50	30.70	69.94
Cruz	97.30	98.60	31.55	99.20	91.05	96.10	98.20	86.30	33.95	34.55	76.68
Uddamvathanak	97.20	98.00	34.95	98.40	92.85	97.60	97.45	82.25	36.30	37.35	77.24

Table 3
The true positive rate of different algorithms (Eq. (2)) on the Tennessee Eastman benchmark (case 1). The data are averaged over 30 runs.

Algorithm	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	Cls 6	Cls 7	Cls 8	Cls 9	Cls 10	Avg
SVM	96.78	95.74	46.93	96.86	96.55	44.62	100.00	60.71	33.85	40.13	71.22
ANN	98.23	97.50	30.58	91.38	97.30	**99.83**	100.00	92.03	33.96	30.70	77.15
SAE	98.35	97.91	40.86	99.06	99.12	51.11	100.00	49.56	46.03	37.86	72.01
DBN	97.99	97.17	41.44	99.64	99.00	49.84	100.00	55.19	43.14	**41.53**	72.49
BLS	98.48	98.36	36.06	98.74	98.18	72.58	99.61	53.36	32.38	30.55	71.83
Raw Stat.	97.09	96.27	31.53	91.08	97.58	51.13	99.71	56.19	36.54	33.93	69.10
Fourier Stat.	96.92	97.16	41.39	94.11	97.32	52.84	99.93	76.08	34.19	38.11	72.81
Wavelet Stat.	97.39	97.01	33.31	91.08	96.45	44.80	99.52	52.81	33.22	34.22	67.98
Walsh Stat.	97.75	97.90	42.19	94.47	98.06	66.87	99.75	75.19	43.30	38.82	75.43
HHT Stat.	97.67	97.58	42.13	90.68	98.29	51.59	99.81	70.86	39.93	36.13	72.47
1D to 2D CNN	97.13	98.18	39.21	96.81	98.34	63.24	99.99	76.83	36.50	34.61	74.08
STFT CNN	98.13	96.22	38.40	95.15	97.80	62.83	99.53	67.91	34.48	35.60	72.60
Wavelet CNN	97.60	97.48	33.07	93.53	96.77	52.26	99.65	81.74	37.81	39.10	72.90
HHT CNN	97.38	97.41	42.77	96.98	98.41	63.43	99.62	64.95	41.84	36.87	73.97
EECNN	98.27	**98.99**	**47.21**	**99.68**	**99.33**	99.79	**100.00**	**92.17**	**46.46**	41.44	**82.30**
1D to 2D CNN_	97.11	98.00	38.30	96.84	97.91	61.98	99.80	72.61	35.66	33.40	73.16
STFT CNN_	98.19	96.23	37.68	94.21	97.41	60.41	99.70	66.83	34.67	35.00	72.03
Wavelet CNN_	97.60	96.84	32.82	93.61	96.91	50.32	99.80	77.95	37.73	37.37	72.09
HHT CNN_	97.21	96.70	40.19	95.66	98.00	62.24	99.80	64.04	40.91	35.24	73.00
Cruz	98.41	98.70	44.07	98.71	99.00	93.08	99.90	88.04	44.47	39.56	80.40
Uddamvathanak	**98.61**	98.30	43.30	97.91	99.00	96.49	99.90	82.87	44.70	40.73	80.18

Table 4
The positive predictive rate of different algorithms (Eq. (3)) on the Tennessee Eastman benchmark (case 1). The data are averaged over 30 runs.

Algorithm	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	Cls 6	Cls 7	Cls 8	Cls 9	Cls 10	Avg
SVM	97.11	100.00	34.50	**99.85**	89.96	**100.00**	99.45	50.30	31.84	43.58	74.66
ANN	92.03	92.44	34.71	95.59	64.98	96.50	97.03	83.86	36.67	54.85	74.87
SAE	90.61	90.81	33.99	97.20	64.03	93.80	99.19	84.61	36.99	55.21	74.64
DBN	91.02	91.80	**36.17**	96.55	64.15	97.96	97.17	83.69	**37.71**	51.98	74.82
BLS	91.01	90.73	33.67	96.87	64.12	94.46	99.43	85.65	37.17	54.30	74.75
Raw Stat.	92.72	97.24	33.53	96.60	76.85	94.99	90.84	67.35	33.64	43.79	72.75
Fourier Stat.	91.00	91.85	35.45	96.93	76.95	95.62	80.58	51.82	35.00	47.48	70.27
Wavelet Stat.	90.74	94.71	33.32	95.78	67.17	95.96	72.73	51.24	31.75	47.98	68.14
Walsh Stat.	94.54	99.05	35.78	97.66	89.82	97.78	84.66	82.40	33.32	54.67	76.97
HHT Stat.	95.28	98.42	34.31	99.02	87.08	95.80	91.37	72.88	36.71	60.70	77.16
1D to 2D CNN	93.36	98.54	35.80	97.16	71.51	98.91	83.04	78.15	35.27	47.46	73.92
STFT CNN	92.39	97.41	33.46	98.92	83.17	94.85	82.29	54.71	33.01	59.93	73.01
Wavelet CNN	95.13	99.80	34.80	99.24	87.61	95.25	97.77	82.04	35.48	53.66	78.08
HHT CNN	92.26	99.11	35.47	98.70	85.43	97.92	95.58	68.17	36.16	52.89	76.17
EECNN	**97.28**	**100.00**	36.11	99.73	**90.67**	100.00	**99.51**	**85.73**	37.49	61.03	**80.72**
1D to 2D CNN_	93.20	98.50	35.44	97.71	70.34	98.60	82.70	69.73	35.16	46.51	72.79
STFT CNN_	91.74	95.85	33.54	98.69	79.69	94.76	80.84	54.29	32.89	59.28	72.16
Wavelet CNN_	94.95	98.30	34.67	98.20	80.45	95.30	95.05	80.24	35.19	51.97	76.43
HHT CNN_	91.97	96.70	35.20	98.61	78.95	97.48	94.33	67.27	35.92	51.39	74.78
Cruz	96.30	98.50	36.06	99.70	85.44	99.60	96.62	85.34	36.86	57.83	79.22
Uddamvathanak	95.93	97.72	35.87	98.90	88.17	98.80	95.22	82.14	37.06	**61.22**	79.10

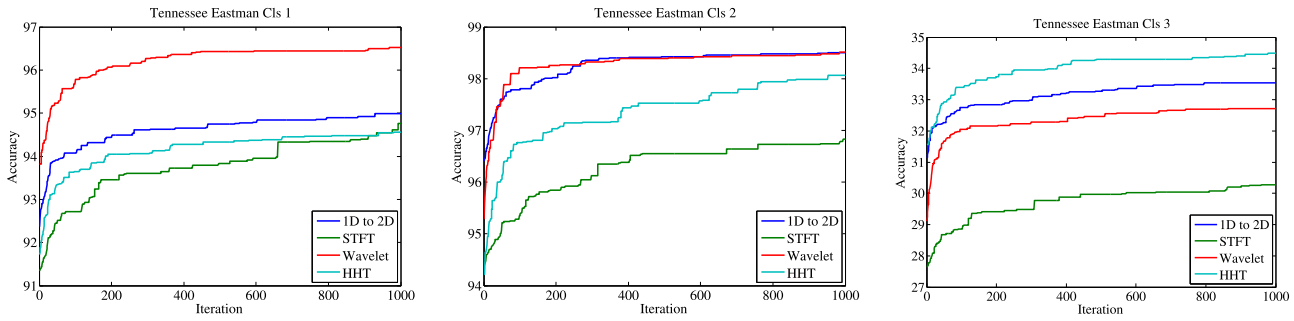


Fig. 5. The progress of the genetic algorithm in the optimization of CNN structure for three classes of Tennessee Eastman problem.

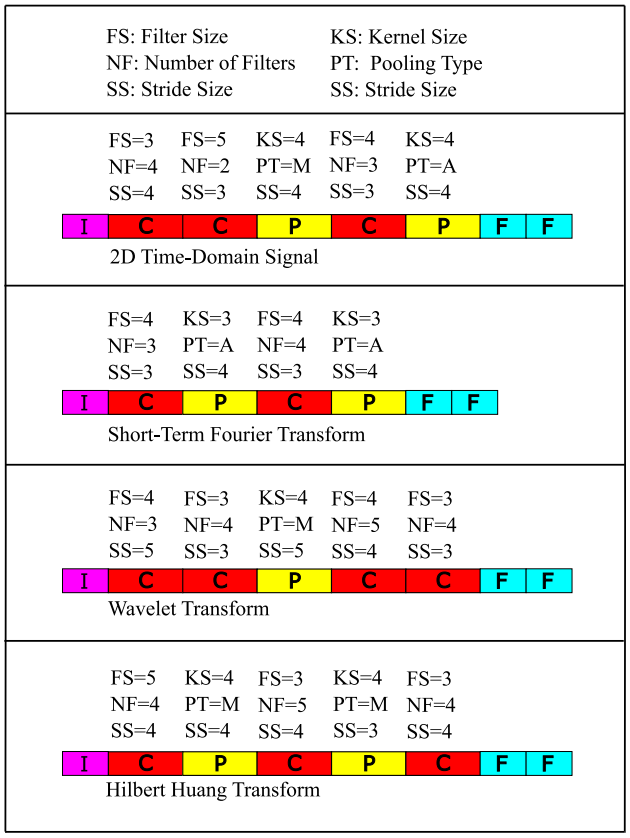


Fig. 6. The best CNN architecture for each of the signal types.

which they belong. By migrating, the individuals transfer knowledge between the sub-population. This way, the sub-populations share the global information they have collected about the search space.

4. Experimental results

In this section, we start the experiments by finding the best CNN architecture for got each type of signal. We use the algorithm 1 to optimize the best architecture for CNN and the best architecture for each of the signal types are presented in Fig. 6. The population size is set to $n = 20$, the maximum number of layers in the CNNs is set to $l = 8$, the mutation rate is set to $m = 0.05$ and the termination condition is set to 1000 number of generations. In all our experiments, the signals in this experiment are converted into 64×64 images. The optimization process is independently performed on all the signal types and the best architecture is reported in this figure. The ordering of the convolutional and the pooling operators along with the filter size, number of filters, stride size, kernel size, and pooling type are presented. For all our experiments in this paper, we use these architectures.

The experiments on the rival algorithms are performed via public toolboxes. For the SVM algorithm the libSVM (Chang & Lin, 2011) and for the BLS algorithm, the BLS (Chen & Liu, 2017) toolboxes are used.

Table 5
The accuracy of different algorithms (Eq. (4)) on the Three-Phase Flow Facility benchmark (case 2). The data are averaged over 30 runs.

Algorithm	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	Avg
SVM	25.80	96.25	99.77	25.87	100.00	69.54
ANN	28.36	97.06	99.04	37.25	99.40	72.94
SAE	26.99	96.98	98.76	36.41	99.25	72.05
DBN	31.27	95.52	98.77	25.29	100.00	70.17
BLS	82.00	63.72	**99.93**	37.29	100.00	76.59
Raw Stat.	19.32	73.98	98.50	30.73	99.21	64.35
Fourier Stat.	70.54	80.46	98.26	24.32	99.54	74.62
Wavelet Stat.	44.94	66.03	98.45	29.60	99.31	67.67
Walsh Stat.	59.92	82.66	99.69	35.56	99.98	75.56
HHT Stat.	56.36	77.15	98.98	27.18	99.69	71.87
1D to 2D CNN	77.45	70.36	98.93	34.85	99.84	76.29
STFT CNN	52.27	80.25	98.28	32.40	100.00	72.64
Wavelet CNN	69.58	81.91	99.13	34.21	99.74	76.91
HHT CNN	72.42	78.71	99.17	36.68	99.51	77.30
EECNN	**85.72**	**97.24**	99.76	**40.69**	**100.00**	**83.77**
1D to 2D CNN_	63.75	69.75	98.70	30.30	99.60	72.42
STFT CNN_	47.35	78.25	98.40	35.25	99.65	71.78
Wavelet CNN_	67.90	78.75	98.95	37.55	99.60	76.55
HHT CNN_	70.70	75.10	99.30	36.85	99.75	76.34
Cruz	83.55	97.03	99.70	38.45	99.95	82.37
Uddamvathanak	75.35	97.08	99.70	33.10	99.90	79.67

Table 6
The true positive rate of different algorithms (Eq. (2)) on the Three-Phase Flow Facility benchmark (case 2). The data are averaged over 30 runs.

Algorithm	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	Avg
SVM	53.13	92.93	100.00	31.05	100.00	75.42
ANN	30.66	**94.28**	100.00	39.97	100.00	72.98
SAE	30.11	93.74	100.00	40.23	100.00	72.82
DBN	36.81	91.27	100.00	30.47	100.00	71.71
BLS	82.89	63.30	100.00	40.25	100.00	77.29
Raw Stat.	47.29	76.51	99.06	32.85	100.00	71.14
Fourier Stat.	68.89	72.54	98.50	33.05	100.00	74.60
Wavelet Stat.	46.01	63.76	98.46	33.92	100.00	68.43
Walsh Stat.	63.91	76.80	98.83	36.92	100.00	75.29
HHT Stat.	70.88	75.92	99.07	34.27	100.00	76.03
1D to 2D CNN	74.71	66.34	98.90	36.35	100.00	75.26
STFT CNN	54.43	83.68	98.79	37.07	100.00	74.79
Wavelet CNN	67.32	80.44	98.95	37.99	100.00	76.94
HHT CNN	71.81	75.20	99.42	37.40	100.00	76.76
EECNN	**86.35**	93.84	**100.00**	**40.32**	**100.00**	**84.10**
1D to 2D CNN_	62.21	65.93	99.00	34.45	100.00	72.32
STFT CNN_	53.02	81.12	98.70	36.37	100.00	73.84
Wavelet CNN_	67.07	75.74	98.71	37.24	100.00	75.75
HHT CNN_	71.01	71.42	99.10	36.65	100.00	75.63
Cruz	82.82	89.87	99.90	39.88	100.00	82.49
Uddamvathanak	80.12	88.63	99.50	37.53	100.00	81.16

Table 7
The positive predictive rate of different algorithms (Eq. (3)) on the Three-Phase Flow Facility benchmark (case 2). The data are averaged over 30 runs.

Algorithm	Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	Avg
SVM	33.16	100.00	99.73	43.03	100.00	75.18
ANN	30.77	100.00	97.65	43.50	99.19	74.22
SAE	30.81	100.00	97.83	43.64	99.13	74.28
DBN	30.63	100.00	97.11	35.16	100.00	72.58
BLS	80.37	68.09	98.85	43.15	100.00	78.09
Raw Stat.	30.83	72.97	97.79	36.01	99.03	67.33
Fourier Stat.	72.12	98.18	98.12	37.30	99.41	81.03
Wavelet Stat.	42.64	77.02	97.81	43.69	99.04	72.04
Walsh Stat.	60.02	92.87	**99.94**	44.10	99.62	79.31
HHT Stat.	55.91	80.18	98.79	44.99	99.90	75.95
1D to 2D CNN	82.29	80.61	98.79	42.71	99.25	80.73
STFT CNN	54.89	77.67	98.25	39.30	99.28	73.88
Wavelet CNN	78.88	86.19	99.75	40.77	98.69	80.85
HHT CNN	74.86	86.60	99.42	41.71	100.00	80.52
EECNN	**84.94**	**100.00**	99.80	**45.10**	**100.00**	**85.97**
1D to 2D CNN_	66.09	79.18	98.41	41.52	99.20	76.88
STFT CNN_	48.03	76.59	98.11	38.51	99.30	72.11
Wavelet CNN_	70.95	85.64	99.20	40.63	99.20	79.13
HHT CNN_	70.16	84.44	99.50	41.44	99.50	79.01
Cruz	84.88	95.09	99.50	44.46	99.90	84.76
Uddamvathanak	73.34	92.67	99.90	43.31	99.80	81.80

In order to implement ANN, SAE, and DBN algorithms, the DeepLearn toolbox (DeepLearn Toolbox, 2015) is used. For CNN, the MATLAB toolbox is used, for SVM, the polynomial kernel is used, for ANN, we use a 5 layer network with 20 neurons at each layer and for other algorithms, the default hyper parameters are used. For BLS, the number of feature nodes is set to 100 and the number of enhancement nodes is set to 2000. All other parameters in the algorithms used the default parameters of the employed software. The statistical features including Fourier, Wavelet, Walsh and HHT features are extracted via functions implemented via Matlab 2018a toolbox.

To implement and run the experiments we used the cluster system at the University of Hertfordshire. The information about its architecture can be found in clu (2022).

In order to compare the proposed algorithm with other ensemble algorithms, in this paper we perform the experiments on the algorithms developed in Cruz et al. (2021) and Uddamvathanak et al. (2018). In the experimental results in the tables, these algorithms are referred to as Cruz and Uddamvathanak.

In order to perform a fair comparison between the proposed ensemble algorithm and the two other ensemble learning algorithms, we tried to give similar time budget to all algorithms.

Table 1 shows the range of the optimization parameters in the proposed algorithm. Theoretically, the connection weights can get any real number, so during the optimization process, we did not put any limit on this. However, in the initialization process, random values are generated between [−10,10]. It is then to the optimization process to find the optimal value. In this paper, we allowed the algorithm to get three types of pooling which include maximum, minimum, and average operators.

Fig. 5 shows the evolutionary process of algorithm 1 in optimizing the architecture of CNNs. This is the accuracy of the resulting CNN at different iterations of the algorithm. The plots are generated for three of the benchmarks and for the four CNN feature extractors. As the data suggest, the performance starts from a random architecture and progresses to reach the best architecture found by the proposed algorithm.

As mentioned in the description of the algorithm 3, in step 5 of the algorithm, if in calculating the fitness, the performance of the individuals is measured based on its accuracy on the training data, the evolutionary algorithm would select the individuals that fit the training data and so the over-fitting problem would occur. To manage this, the data are partitioned into the train and test sets, so the architectures are trained on the training data but their fitness is measured based on unseen test data. We suggested that this would manage the over-fitting problem and improve the performance of the algorithm. In order to compare the proposed algorithm with gradient descent, we also train CNNs via gradient descent and the results are presented in the experiments. The algorithms that end with '_' means that the algorithm uses gradient descent in the training phase. For example 'STFT CNN' means when the CNN with STFT features are trained via the proposed evolutionary training and 'STFT CNN_' means when the CNN is trained via gradient descent.

This paper uses three measures to evaluate the performance of the algorithms, which are true positive rate (known as sensitivity), positive predictive rate (known as precision), and accuracy.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{2}$$

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \tag{3}$$

Table 8
The performance of different algorithms in terms of accuracy, true positive rate, and positive predictive rate on the Motor Bearing Fault Diagnosis, the Self-Priming Centrifugal Pump Fault Diagnosis, and Axial Piston Hydraulic Pump Fault Diagnosis benchmarks. The data are averaged over 30 runs.

Algorithm	Case 3 (TSP)			Case 4 (SPCP)			Case 5 (APHP)		
	ACC	TPR	PPV	ACC	TPR	PPV	ACC	TPR	PPV
SVM	99.77	99.53	99.61	99.46	99.31	99.36	99.65	99.87	99.81
ANN	99.66	99.68	99.59	99.40	99.35	99.39	99.78	99.77	99.87
SAE	99.61	99.51	99.61	99.36	99.28	99.24	99.88	99.86	99.85
DBN	99.65	99.47	99.77	99.06	99.09	99.43	99.93	99.84	100.00
BLS	99.79	99.71	99.66	99.29	99.17	**99.46**	99.98	99.99	99.83
Raw Stat.	99.44	99.60	99.62	99.46	99.39	99.37	99.68	99.70	99.79
Fourier Stat.	99.72	99.45	99.67	99.18	99.32	99.35	99.81	99.72	99.83
Wavelet Stat.	99.48	99.35	99.67	99.13	99.07	99.24	99.83	99.83	99.78
Walsh Stat.	99.75	99.68	99.85	99.44	99.34	99.25	99.74	99.72	100.00
HHT Stat.	99.72	99.76	99.83	99.45	99.34	99.38	99.80	99.90	99.89
1D to 2D CNN	99.67	99.65	99.84	99.39	99.21	99.40	99.73	99.82	99.90
STFT CNN	99.54	99.39	99.77	99.28	99.34	99.27	99.99	99.89	99.83
Wavelet CNN	99.71	99.78	99.74	99.42	99.32	99.32	99.79	99.77	100.00
HHT CNN	99.76	99.68	99.75	99.27	99.35	99.16	99.82	99.93	99.74
EECNN	**99.84**	**99.80**	**99.87**	**99.50**	99.37	99.42	**100.00**	**100.00**	**100.00**
1D to 2D CNN_	99.65	99.60	99.70	99.30	99.20	99.40	99.80	99.80	99.80
STFT CNN_	99.55	99.50	99.60	99.30	99.30	99.30	99.85	99.90	99.80
Wavelet CNN_	99.65	99.60	99.70	99.35	99.30	99.40	99.80	99.80	99.80
HHT CNN_	99.75	99.64	99.70	99.30	99.30	99.30	99.90	99.90	99.90
Cruz	99.82	99.78	99.68	99.38	**99.40**	99.37	99.93	99.84	99.93
Uddamvathanak	99.79	99.71	99.79	99.47	99.38	99.41	99.99	99.91	99.94

Table 9
The time it takes to train and test each of the algorithms in seconds. The data are averaged over 30 runs.

Algorithm	Case 1		Case 2		Case 3		Case 4		Case 5	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
SVM	35.30	3.10e−03	58.30	5.10e−03	65.30	6.10e−03	47.30	4.10e−03	58.30	5.10e−03
ANN	41.05	3.51e−03	64.05	5.51e−03	71.05	6.51e−03	53.05	4.51e−03	64.05	5.51e−03
SAE	38.22	3.23e−03	61.22	5.23e−03	68.22	6.23e−03	50.22	4.23e−03	61.22	5.23e−03
DBN	39.42	3.13e−03	62.42	5.13e−03	69.42	6.13e−03	51.42	4.13e−03	62.42	5.13e−03
BLS	40.21	4.05e−03	63.21	6.05e−03	70.21	7.05e−03	52.21	5.05e−03	63.21	6.05e−03
Raw Stat.	40.29	3.70e−03	63.29	5.70e−03	70.29	6.70e−03	52.29	4.70e−03	63.29	5.70e−03
Fourier Stat.	37.39	3.26e−03	60.39	5.26e−03	67.39	6.26e−03	49.39	4.26e−03	60.39	5.26e−03
Wavelet Stat.	42.91	3.81e−03	65.91	5.81e−03	72.91	6.81e−03	54.91	4.81e−03	65.91	5.81e−03
Walsh Stat.	43.12	3.48e−03	66.12	5.48e−03	73.12	6.48e−03	55.12	4.48e−03	66.12	5.48e−03
HHT Stat.	39.89	3.95e−03	62.89	5.95e−03	69.89	6.95e−03	51.89	4.95e−03	62.89	5.95e−03
1D to 2D CNN	353.63	3.90e−03	583.63	5.90e−03	653.63	6.90e−03	473.63	4.90e−03	583.63	5.90e−03
STFT CNN	358.48	4.16e−03	588.48	6.16e−03	658.48	7.16e−03	478.48	5.16e−03	588.48	6.16e−03
Wavelet CNN	356.99	4.34e−03	586.99	6.34e−03	656.99	7.34e−03	476.99	5.34e−03	586.99	6.34e−03
HHT CNN	363.41	4.27e−03	593.41	6.27e−03	663.41	7.27e−03	483.41	5.27e−03	593.41	6.27e−03
EECNN	1914.62	5.27e−02	3117.59	8.12e−02	3483.71	9.54e−02	2542.26	6.69e−02	3117.59	8.12e−02
1D to 2D CNN_	358.05	4.19e−03	588.05	6.19e−03	658.05	7.19e−03	478.05	5.19e−03	588.05	6.19e−03
STFT CNN_	357.48	4.84e−03	587.48	6.84e−03	657.48	7.84e−03	477.48	5.84e−03	587.48	6.84e−03
Wavelet CNN_	353.67	4.01e−03	583.67	6.01e−03	653.67	7.01e−03	473.67	5.01e−03	583.67	6.01e−03
HHT CNN_	361.96	4.35e−03	591.96	6.35e−03	661.96	7.35e−03	481.96	5.35e−03	591.96	6.35e−03
Cruz	1959.01	5.48e−02	3189.87	8.43e−02	3564.48	9.91e−02	2601.20	6.95e−02	3189.87	8.43e−02
Uddamvathanak	1980.26	5.61e−02	3224.46	8.64e−02	3603.13	1.02e−01	2629.41	7.13e−02	3224.46	8.64e−02

Table 10
The accuracy of different algorithms (Eq. (4)) on the Tennessee Eastman benchmark (case 1) for different evolutionary algorithms. The data are averaged over 30 runs.

Algorithm		Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	Cls 6	Cls 7	Cls 8	Cls 9	Cls 10
A3	1D to 2D CNN	94.99	98.50	33.53	97.11	79.74	70.29	89.69	76.75	31.17	31.35
	STFT CNN	94.76	96.83	30.28	97.29	89.56	70.16	88.92	56.90	34.41	26.23
	Wavelet CNN	**96.52**	**98.51**	32.71	96.18	**91.27**	53.53	**98.84**	**81.50**	**37.66**	**37.71**
	HHT CNN	94.56	98.06	34.48	**98.02**	90.69	**70.64**	97.71	65.25	31.93	23.78
PSO	1D to 2D CNN	95.00	98.33	34.59	97.14	78.79	69.73	89.47	74.70	34.44	31.20
	STFT CNN	94.72	96.73	31.01	96.73	88.29	69.88	88.66	56.46	33.86	25.42
	Wavelet CNN	96.33	97.87	33.09	96.02	89.37	52.91	98.57	80.35	35.94	32.75
	HHT CNN	94.53	96.70	34.13	97.52	89.18	70.04	97.18	65.14	33.18	25.00
GA	1D to 2D CNN	94.99	98.50	33.75	97.20	79.36	69.60	89.64	75.92	34.13	27.03
	STFT CNN	94.72	96.63	30.76	96.94	86.39	68.56	88.53	56.54	32.79	25.51
	Wavelet CNN	96.38	97.95	33.20	96.01	90.23	52.74	97.43	81.21	35.06	35.55
	HHT CNN	94.36	98.06	31.81	97.34	90.25	69.85	97.35	65.12	32.55	25.15
DE	1D to 2D CNN	94.99	98.30	33.75	97.16	79.27	69.11	89.53	71.54	31.83	30.72
	STFT CNN	94.72	96.69	30.87	97.26	86.99	69.00	88.89	56.09	32.64	25.42
	Wavelet CNN	96.40	98.37	33.28	95.87	88.25	52.52	97.58	81.38	36.79	35.22
	HHT CNN	94.44	97.88	31.24	97.90	87.50	69.85	96.93	65.15	33.27	29.09
ES	1D to 2D CNN	95.00	98.26	34.62	97.17	78.41	70.28	89.47	71.17	32.55	31.25
	STFT CNN	94.71	96.59	31.29	96.67	86.26	66.98	88.15	55.96	33.92	24.81
	Wavelet CNN	96.34	98.25	33.08	96.07	89.69	51.55	97.91	81.12	35.75	37.08
	HHT CNN	94.37	97.34	30.61	97.32	86.88	70.45	97.54	65.24	32.53	24.75
FES	1D to 2D CNN	95.00	98.39	34.76	97.19	78.88	69.28	89.56	71.38	34.58	31.31
	STFT CNN	94.75	96.54	31.59	97.00	87.99	67.89	88.09	56.38	32.73	26.00
	Wavelet CNN	96.34	98.43	33.53	95.89	86.92	52.20	98.12	80.84	35.33	35.24
	HHT CNN	94.36	97.64	31.77	97.82	88.95	69.54	97.37	65.19	32.17	27.64
EP	1D to 2D CNN	94.99	98.43	**35.14**	97.22	78.92	69.48	89.52	74.82	34.76	30.69
	STFT CNN	94.70	96.26	30.60	96.57	88.07	69.30	88.42	55.89	32.59	25.28
	Wavelet CNN	96.43	97.62	33.11	96.05	88.36	52.80	98.37	79.87	35.41	37.17
	HHT CNN	94.52	96.95	32.44	97.32	89.84	70.40	97.47	65.22	32.27	26.58
FEP	1D to 2D CNN	95.00	98.49	34.62	97.12	78.29	70.02	89.50	72.60	31.93	29.01
	STFT CNN	94.74	96.49	30.94	96.46	89.47	70.01	88.25	55.69	33.10	25.98
	Wavelet CNN	96.39	97.82	32.86	96.12	90.44	52.25	98.01	80.70	35.19	32.65
	HHT CNN	94.53	96.91	33.19	97.66	86.83	70.52	96.93	65.14	32.59	25.70
MA-SW	1D to 2D CNN	95.00	98.29	33.93	97.18	78.65	69.54	89.60	70.55	34.53	28.65
	STFT CNN	94.74	96.76	31.65	97.16	87.69	70.06	88.06	56.68	32.32	25.02
	Wavelet CNN	96.25	98.27	33.48	96.14	86.72	53.41	97.49	79.58	36.13	36.30
	HHT CNN	94.54	97.84	30.67	97.20	86.53	69.85	97.28	65.16	32.96	29.20

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \quad (4)$$

The following cases have been used in this paper to test the proposed algorithm.

4.1. Case 1: Tennessee Eastman process

Developed by the Eastman chemical company, the Tennessee Eastman (TE) process is used in the literature as a benchmark for evaluating developed algorithms (Downs & Vogel, 1993). The Tennessee Eastman Process is a model which is based on a real-world chemical process. The

Table 11
The accuracy of different algorithms (Eq. (4)) on the Three-Phase Flow Facility benchmark (case 2) for different evolutionary algorithms. The data are averaged over 30 runs.

Algorithm		Cls 1	Cls 2	Cls 3	Cls 4	Cls 5
A3	1D to 2D CNN	**77.45**	70.36	98.93	34.85	99.84
	STFT CNN	52.27	80.25	98.28	32.40	**100.00**
	Wavelet CNN	69.58	**81.91**	99.13	34.21	99.74
	HHT CNN	72.42	78.71	99.17	36.68	99.51
PSO	1D to 2D CNN	70.32	70.35	98.70	31.20	99.63
	STFT CNN	51.32	79.12	98.36	32.90	99.73
	Wavelet CNN	68.86	79.85	99.10	36.69	99.69
	HHT CNN	71.88	78.08	99.19	36.78	99.67
GA	1D to 2D CNN	65.31	70.25	98.77	30.79	99.70
	STFT CNN	52.21	79.61	98.31	35.24	99.90
	Wavelet CNN	68.49	80.80	99.08	34.71	99.67
	HHT CNN	70.87	75.66	99.19	36.78	99.56
DE	1D to 2D CNN	73.30	70.15	98.90	30.41	99.83
	STFT CNN	48.84	78.69	98.29	33.06	99.95
	Wavelet CNN	69.24	79.52	99.13	**37.09**	99.62
	HHT CNN	71.09	75.80	99.18	36.70	99.70
ES	1D to 2D CNN	75.08	70.26	98.80	34.04	99.78
	STFT CNN	50.59	79.72	98.36	33.98	99.85
	Wavelet CNN	68.16	81.11	99.04	36.83	99.67
	HHT CNN	72.10	76.14	99.21	36.74	99.64
FES	1D to 2D CNN	71.10	70.01	98.87	34.15	99.68
	STFT CNN	52.10	79.45	98.34	33.71	99.85
	Wavelet CNN	68.24	80.11	99.07	35.03	99.68
	HHT CNN	72.09	76.23	99.24	36.70	99.72
EP	1D to 2D CNN	70.31	70.20	98.80	30.99	99.75
	STFT CNN	47.42	79.77	98.34	34.14	99.89
	Wavelet CNN	69.26	80.12	98.95	34.68	99.69
	HHT CNN	71.90	78.56	99.22	36.80	99.66
FEP	1D to 2D CNN	68.23	69.88	98.73	32.29	99.66
	STFT CNN	48.92	80.00	98.37	33.99	99.95
	Wavelet CNN	68.80	81.37	99.03	36.12	99.66
	HHT CNN	71.65	76.65	99.19	36.81	99.65
MA-SW	1D to 2D CNN	73.50	70.19	98.86	32.60	99.64
	STFT CNN	47.37	79.29	98.35	33.29	99.98
	Wavelet CNN	68.36	79.89	99.06	35.96	99.69
	HHT CNN	71.18	77.31	**99.28**	36.76	99.71

model consists of a separator, a reactor and a recycle arrangement that involves two simultaneous gas-liquid exothermic reactions. The system is a nonlinear open-loop unstable process that presents a model of a complex system that has found applications in many research as case study for fault detection.

In this paper, the first fifteen faults of the TE process are used, where 640 samples are used for the training phase, and 640 are used to test the algorithms. The 161st and the 960th samples are selected as the injection and the clear time respectively. The aim of this paper is to identify the fault class of the abnormal samples, so it is assumed that the selected faults are detected at the 161st sample.

In order to compare different algorithms and feature extraction methods, the performance of the accuracy of the algorithms is summarized in Table 2. On average, the rival algorithms reach a performance around 65%–70% accuracy. The base learner algorithms reach similar performance. As the data suggest, the proposed ensemble algorithm achieves better performance than the base learners. This is what is expected in ensemble learning, where the voting system improves the performance of the individual learners. The ensemble algorithm achieves around 5% better performance compared to the best base learner algorithm which is Statistical features via Walsh transform. Also, it is clear from the data that the CNN algorithms, on average perform better than traditional feature extraction methods. This could be attributed to the fact that CNNs are better at extracting features that are representative of different anomalies in the fault signals, anomalies that are not reflected in statistical features. The data for true positive rate and positive predictive rate for the same experiments are presented in Tables 3 and 4. While the proposed algorithm performs the best in terms of accuracy, in terms of TPR in Table 3, in some cases, other algorithms perform better. For example for the case of class 1, SAE performs better than the proposed algorithm. However, in terms of PPV in Table 4 the proposed algorithm performs much better.

Although in some cases like Cls 7, the proposed algorithm is outperformed with the existing methods, on average, as presented in the last column of the table, the proposed algorithm reaches the best performance among the algorithms.

The performance of all the algorithms is very low for Cls 3 in Tables 2, 3, and 4. This is because this class is a hard problem to identify. An overview on

4.2. Case 2: Three-phase flow facility

Designed by Cranfield University, the three-phase flow (TPF) facility provides a controlled and measured flow rate of air, oil, or water to a pressurized system (Ruiz-Cárcel, Cao, Mba, Lao, & Samuel, 2015). The problem used in this paper to test the proposed algorithms is defined as identifying five typical faults of the system. In this paper, 2/3 of the data are used as the training, and 1/3 are used to test the algorithms. In all our experiments, the selection is performed without insertion, so there is no shared data record between the train and test data. A comparison between the performance of the algorithms in terms of accuracy is performed in Table 5. The performance of the rival algorithms and the base learner algorithms is around 70%. The proposed ensemble algorithm can boost the performance of the rival algorithms by around 5% and reach the accuracy of 78.63%. The performance of the algorithms in terms of TPR and PPV are summarized in Tables 6 and 7.

The last column in these tables presents the average performance of each of the algorithms. As the data suggest, on average, the proposed algorithm outperform the other algorithms. Note that on average, the other ensemble algorithms also outperform the base-learners which is expected from an ensemble algorithm.

4.3. Case 3: Motor Bearing Fault Diagnosis

The third set of experiments in this paper is applied to the motor bearing fault diagnosis benchmark designed by Case Western Reserve University (Smith & Randall, 2015). There are three types of faults in this benchmark, which are roller fault, outer race fault, and inner race fault. Each of these fault types has three different damage sizes which are 0.18 mm, 0.36 mm, and 0.54 mm. Thus, there are ten possible conditions, nine corresponding to the 9 types of faults and one to the normal condition. To test the performance of the proposed method, the signals in these systems are collected under four load conditions (0, 1, 3, 4 hp). In our experiments, we select randomly 2000 samples per load condition for the training and 400 samples per load condition for the testing phase. The accuracy, true positive rate, and positive predictive rate of the algorithms are summarized in Table 8. The best performance among the rival algorithms is achieved by SVM and the best algorithms among the base learners are HHT CNN. The data suggest that the proposed ensemble learning algorithm improves the performance of the base learners.

4.4. Case 4: Self-priming centrifugal pump fault diagnosis

The fourth benchmark in this paper is the self-priming centrifugal pump problem (Lu, Lin et al., 2019). In this benchmark, acceleration sensors are installed on a pedestal above the motor housing. The rotation speed of the motor is 2900 per minute and the sampling frequency of the vibration signal is 10240 Hz. There are four fault conditions that are bearing roller wearing, inner race wearing, outer race wearing, and impeller wearing. The training set contains 200 and the test set contains 400 samples per condition. Overall, there are 50,000 images in the

Table 12
The performance of different algorithms in terms of accuracy, true positive rate, and positive predictive rate on the Motor Bearing Fault Diagnosis, the Self-Priming Centrifugal Pump Fault Diagnosis, and Axial Piston Hydraulic Pump Fault Diagnosis benchmarks for different evolutionary algorithms. The data are averaged over 30 runs.

Algorithm		Case 3 (TSP)			Case 4 (SPCP)			Case 5 (APHP)		
		ACC	TPR	PPV	ACC	TPR	PPV	ACC	TPR	PPV
A3	1D to 2D CNN	99.67	99.65	99.84	99.39	99.21	99.40	99.73	99.82	99.90
	STFT CNN	99.54	99.39	99.77	99.28	99.34	99.27	99.99	99.89	99.83
	Wavelet CNN	99.71	99.81	99.74	99.42	99.32	99.32	99.79	99.77	100.00
	HHT CNN	99.76	99.68	99.75	99.27	99.35	99.16	99.82	99.93	99.74
PSO	1D to 2D CNN	99.66	99.60	99.73	99.31	99.21	99.40	99.76	99.82	99.87
	STFT CNN	99.55	99.49	99.75	99.29	99.32	99.30	99.89	99.90	99.81
	Wavelet CNN	99.65	99.65	99.71	99.41	99.31	99.34	99.80	99.78	99.86
	HHT CNN	99.75	99.76	99.71	99.29	99.34	99.18	99.88	99.90	99.80
GA	1D to 2D CNN	99.65	99.64	99.82	99.35	99.21	99.40	99.75	99.81	99.90
	STFT CNN	99.55	99.46	99.68	99.29	99.33	99.29	99.86	99.90	99.82
	Wavelet CNN	99.67	99.77	99.72	99.37	99.31	99.36	99.79	99.79	99.92
	HHT CNN	99.75	99.69	99.70	99.30	99.32	99.27	99.82	99.92	99.80
DE	1D to 2D CNN	99.65	99.63	99.82	99.34	99.20	99.40	99.78	99.80	99.82
	STFT CNN	99.55	99.39	99.77	99.30	99.32	99.27	99.89	99.90	99.82
	Wavelet CNN	99.65	99.61	99.73	99.39	99.32	99.36	99.79	99.79	99.89
	HHT CNN	99.76	99.75	99.72	99.29	99.30	99.24	99.87	99.92	99.80
ES	1D to 2D CNN	99.66	99.63	99.73	99.32	99.21	99.40	99.77	99.80	99.87
	STFT CNN	99.55	99.40	99.73	99.30	99.31	99.29	99.91	99.90	99.82
	Wavelet CNN	99.70	99.63	99.73	99.38	99.31	99.34	99.80	99.77	99.86
	HHT CNN	99.75	99.71	99.73	99.28	99.31	99.27	99.86	99.91	99.88
FES	1D to 2D CNN	99.67	99.60	99.84	99.37	99.21	99.40	99.79	99.80	99.84
	STFT CNN	99.55	99.46	99.70	99.29	99.31	99.28	99.96	99.90	99.81
	Wavelet CNN	99.69	99.65	99.72	99.35	99.30	99.39	99.80	99.78	99.82
	HHT CNN	99.76	99.75	99.71	99.30	99.33	99.21	99.83	99.93	99.84
EP	1D to 2D CNN	99.67	99.61	99.82	99.38	99.21	99.40	99.76	99.82	99.82
	STFT CNN	99.55	99.40	99.76	99.30	99.33	99.28	99.91	99.90	99.81
	Wavelet CNN	99.67	99.78	99.73	99.36	99.31	99.37	99.80	99.79	99.93
	HHT CNN	99.76	99.74	99.71	99.30	99.33	99.30	99.87	99.92	99.90
FEP	1D to 2D CNN	99.66	99.65	99.83	99.36	99.21	99.40	99.76	99.82	99.82
	STFT CNN	99.55	99.48	99.72	99.28	99.32	99.27	99.89	99.89	99.81
	Wavelet CNN	99.67	99.79	99.70	99.42	99.31	99.37	99.79	99.79	99.90
	HHT CNN	99.75	99.77	99.72	99.28	99.32	99.28	99.83	99.91	99.88
MA-SW	1D to 2D CNN	99.66	99.63	99.76	99.32	99.21	99.40	99.76	99.81	99.84
	STFT CNN	99.55	99.50	99.64	99.28	99.31	99.28	99.86	99.89	99.81
	Wavelet CNN	99.70	99.76	99.72	99.38	99.30	99.35	99.79	99.78	99.80
	HHT CNN	99.75	99.76	99.71	99.28	99.30	99.24	99.88	99.91	99.75

training and 10,000 images in the test dataset. The performance of the algorithms in terms of ACC, TPR, and PPV is summarized in Table 8. The best performance among the rival algorithms is achieved by SVM and among CNN algorithms by the HHT-CNN algorithm. The proposed ensemble algorithm improves the performance of the base learners by around 0.5%.

4.5. Case 5: Axial piston hydraulic pump fault diagnosis

The Axial Piston Hydraulic Pump Fault Diagnosis (Lu, Lin et al., 2019) is defined as diagnosing two fault conditions of a pump via the signals collected from an accelerograph installed at the end face of a pump with a sampling frequency of 1 kHz. The two fault types are the piston shoes and swashplate wearing (PS wearing) and valve plate wearing (VP wearing). So there are three fault conditions of PS, VP, and normal conditions. The rotation speed of the system is 5280 rpm and the corresponding spindle frequency is 88 Hz. The performance of different algorithms is summarized in Table 8. The best performance among the rival algorithms is achieved by BLS and the best performance among CNN algorithms is reached by short term Fourier transform. The proposed algorithm reaches the accuracy of 100% in all the experiments.

As the data in these tables suggest, the performance of the algorithms for Cls 3 and Cls 9 in case 1 and Cls 4 in case 2 is much lower than other benchmarks. An overview on other research in the literature shows that other works have achieved similar results (for example, the work presented in Yu & Zhao, 2019). This suggest that these cases are hard to solve for learning algorithms and further research is required to improve the performance further.

In order to compare the algorithms in terms of computational complexity, Table 9 presents the time it takes to train and test each of them in seconds. The data in this table are averaged over 30 independent runs. The time it takes for the ensemble algorithms in this table is greater than the other algorithms because ensemble algorithms employ the base-learners and so the time they require should be in the order of the sum of the time it takes for the base-learners to process data. The training time for the proposed algorithm is larger than the base-learners. This is because any ensemble learning algorithm involves the training of the base-learners. It takes around half an hour to train the learning algorithm. However, in these systems, the training is performed only once and it is testing that will perform continuously. Therefore, the testing time is more important and should be taken into account. In terms of testing, the time it takes for the proposed algorithm to generate the results is around the sum of the time it takes for the base-learners to produce the output. All the testing times are in the order of milliseconds.

In order to compare the performance of different evolutionary algorithms, including GA (Holland, 1975), PSO (Kennedy, Eberhart, & Shi, 2001), Evolutionary Programming (EP) (Fogel, Owens, & Walsh, 1966), Fast Evolutionary Programming (FEP) (Yao, Liu, & Lin, 1999), Evolutionary Strategy (ES) (Schwefel, 1995), Fast Evolutionary Strategy (FES) (Yao & Liu, 1997), Ma-ssw-chains (MASSW) (Molina, Lozano,

Table 13
The accuracy of different algorithms (Eq. (4)) on the Tennessee Eastman benchmark (case 1) for different CNN architectures. The data are averaged over 30 runs.

Algorithm		Cls 1	Cls 2	Cls 3	Cls 4	Cls 5	Cls 6	Cls 7	Cls 8	Cls 9	Cls 10
A1	1D to 2D CNN	94.99	98.50	33.53	97.11	79.74	70.29	89.69	76.75	31.17	31.35
	STFT CNN	94.76	96.83	30.28	97.29	89.56	70.16	88.92	56.90	34.41	26.23
	Wavelet CNN	**96.52**	**98.51**	32.71	96.18	**91.27**	53.53	**98.84**	**81.50**	**37.66**	**37.71**
	HHT CNN	94.56	98.06	34.48	**98.02**	90.69	70.64	97.71	65.25	31.93	23.78
LSTM	1D to 2D CNN	94.50	97.65	**36.95**	96.70	79.35	69.20	87.70	71.45	34.25	25.20
	STFT CNN	94.60	96.85	32.50	96.50	85.30	63.65	88.80	55.65	32.65	24.85
	Wavelet CNN	95.80	97.00	32.65	95.75	86.80	52.55	97.20	72.30	32.85	31.85
	HHT CNN	94.30	97.90	34.00	97.20	86.10	68.15	96.70	61.55	33.65	30.35
SAAA-3	1D to 2D CNN	94.90	97.75	33.20	96.90	77.50	66.90	88.40	76.85	33.90	30.45
	STFT CNN	94.75	96.40	32.10	96.35	88.95	65.85	87.10	53.20	33.60	23.15
	Wavelet CNN	96.30	98.20	31.30	96.05	87.90	52.15	98.35	75.65	37.05	35.15
	HHT CNN	94.30	97.15	31.65	96.90	86.40	62.65	95.60	62.15	29.60	33.95
TIRBM	1D to 2D CNN	94.60	97.55	34.30	96.40	77.35	65.35	89.15	73.00	33.05	27.30
	STFT CNN	94.80	96.50	33.75	95.95	85.05	68.60	88.35	55.30	32.70	28.55
	Wavelet CNN	96.15	97.10	32.25	95.60	90.10	52.00	98.10	81.00	34.00	37.15
	HHT CNN	94.50	97.70	34.95	97.30	89.40	70.45	97.45	63.35	34.85	24.65
PGBM+DN-1	1D to 2D CNN	94.75	97.25	34.35	96.85	77.50	63.40	89.50	73.70	34.85	28.40
	STFT CNN	94.75	96.20	33.90	96.85	88.95	66.20	88.70	55.35	32.50	23.55
	Wavelet CNN	95.85	98.15	30.75	96.05	90.70	51.30	95.00	75.45	37.15	30.25
	HHT CNN	94.40	97.40	33.95	97.35	90.20	67.45	95.45	61.65	33.00	31.95
ScatNet-2	1D to 2D CNN	94.60	98.30	35.85	96.95	78.95	69.60	87.75	72.30	36.85	27.50
	STFT CNN	94.70	96.20	31.70	96.05	85.70	66.80	89.25	55.00	32.40	28.40
	Wavelet CNN	95.95	97.05	30.20	96.20	90.95	49.80	94.45	74.90	35.75	28.40
	HHT CNN	94.25	97.40	32.05	97.25	87.75	67.80	96.25	62.80	32.25	29.00
RandNet-2	1D to 2D CNN	94.75	98.20	33.95	96.90	77.60	62.65	87.90	76.25	34.20	25.55
	STFT CNN	94.60	96.20	29.15	96.65	86.10	64.40	87.45	54.85	31.95	25.45
	Wavelet CNN	96.10	96.75	30.50	95.65	90.50	51.00	97.10	74.15	29.55	32.35
	HHT CNN	94.25	97.30	32.90	96.75	87.70	63.35	93.40	59.50	32.65	31.10
LDANet-2	1D to 2D CNN	94.85	97.45	34.25	96.70	79.50	67.70	88.95	74.35	33.30	24.40
	STFT CNN	94.60	95.90	32.45	96.70	87.60	66.75	87.65	54.25	32.20	25.45
	Wavelet CNN	95.80	97.90	35.15	96.00	90.90	50.10	95.25	80.10	35.05	32.30
	HHT CNN	94.25	97.55	31.95	96.50	89.45	70.00	97.25	64.90	32.70	25.20
NNet	1D to 2D CNN	94.60	97.20	34.90	96.20	78.20	67.15	87.55	75.30	35.35	23.30
	STFT CNN	94.80	95.95	29.85	96.35	85.65	66.50	87.90	56.60	32.45	24.70
	Wavelet CNN	95.85	97.95	33.55	95.95	91.25	51.35	96.60	75.00	35.50	30.80
	HHT CNN	94.30	97.60	33.80	96.85	86.65	65.60	93.40	62.60	32.90	30.15
SQNet	1D to 2D CNN	94.85	98.20	30.50	96.35	79.10	62.05	87.25	74.90	35.60	28.05
	STFT CNN	94.65	96.05	30.40	96.25	89.25	68.90	89.30	55.20	30.70	23.70
	Wavelet CNN	95.75	97.00	33.60	95.70	86.20	53.35	98.00	79.25	33.30	35.80
	HHT CNN	94.35	97.55	30.95	97.55	86.10	67.65	96.30	62.10	30.90	28.75
MobileNetV2	1D to 2D CNN	94.95	98.20	35.15	97.15	77.45	62.75	89.25	75.85	33.05	28.15
	STFT CNN	94.50	96.65	35.30	96.80	85.75	68.25	88.80	53.90	33.25	23.05
	Wavelet CNN	96.10	97.35	32.15	95.70	88.45	50.10	97.15	73.30	33.70	30.95
	HHT CNN	94.45	97.15	32.15	96.70	88.95	63.95	96.55	66.50	32.50	26.30
DBN-3	1D to 2D CNN	94.75	98.30	36.45	96.95	78.30	64.75	87.30	71.70	34.65	29.00
	STFT CNN	94.75	96.30	33.30	96.50	85.45	63.75	87.20	55.50	32.45	24.40
	Wavelet CNN	96.05	97.60	32.30	95.55	90.75	49.85	98.60	79.25	33.95	33.55
	HHT CNN	94.25	97.75	31.35	97.35	88.40	**70.80**	95.15	62.45	32.65	28.95

Sánchez, & Herrera, 2011) and Differential Evolution (DE). These algorithms have been compared with algorithm 3 and the experimental results are summarized in Tables 10–12.

Table 10 presents the accuracy of different optimization algorithms when they are used in the training phase of the CNNs. In this table, A3 represents algorithms 3 and the data are averaged over 30 runs. As the data in this table suggest, the best performance is achieved when the scheme in algorithm 3 is employed. The explanation for this could be that the proposed algorithm uses a set of sub-populations which help the algorithm escape from local optima.

Table 11 presents the accuracy of different evolutionary algorithms on the Three-Phase Flow Facility benchmark (case 2), where the data are averaged over 30 runs. As the data suggest, the proposed algorithm achieves the best performance among the rival algorithms.

Table 12 presents the performance of different algorithms in terms of accuracy, true positive rate and positive predictive rate on the motor bearing fault diagnosis, the self-priming centrifugal pump fault diagnosis, and axial piston hydraulic pump fault diagnosis benchmarks. The data in this table are averaged over 30 independent runs. A comparison between the performance of different evolutionary algorithms suggests that the proposed evolutionary algorithm outperforms the existing algorithms.

To compare the proposed algorithm with other CNN architectures we use a set of algorithms in this paper. The algorithms include SAAA-3 (Larochelle et al., 2007), TIRBM (Sohn & Lee, 2012), PGBM+DN-1 (Sohn & Lee, 2012), ScatNet-2 (Bruna & Mallat, 2013), RandNet-2 (Chan et al., 2015), LDANet-2 (Chan et al., 2015), NNet (Larochelle et al., 2007), SqweezNet (SQNet) (Iandola et al., 2016), MobileNetV2 (Sandler et al., 2018), DBN-3 (Larochelle et al., 2007), and LSTM.

Table 13 shows the experimental results for different CNN architectures. A1 in this table represents the performance of CNNs when the proposed architecture optimization algorithm (presented in algorithm 1) is used to find the optimal architecture for the CNNs. The table summarizes the accuracy of the algorithms for different classes in case study 1. As the data suggest in this table, the proposed algorithm achieves the best performance among the existing architectures. This

Table 14
The accuracy of different algorithms (Eq. (4)) on the Three-Phase Flow Facility benchmark (case 2) for different CNN architectures. The data are averaged over 30 runs.

Algorithm		Cls 1	Cls 2	Cls 3	Cls 4	Cls 5
A1	1D to 2D CNN	**77.45**	70.36	98.93	34.85	99.84
	STFT CNN	52.27	80.25	98.28	32.40	**100.00**
	Wavelet CNN	69.58	81.91	99.13	34.21	99.74
	HHT CNN	72.42	78.71	99.17	36.68	99.51
LSTM	1D to 2D CNN	64.95	69.00	98.80	30.25	99.60
	STFT CNN	53.45	77.50	98.55	33.90	99.65
	Wavelet CNN	62.70	79.00	98.95	35.65	99.60
	HHT CNN	65.85	75.95	99.15	35.55	99.75
SAAA-3	1D to 2D CNN	64.25	69.30	98.70	33.05	99.60
	STFT CNN	47.25	76.35	98.55	33.85	99.65
	Wavelet CNN	57.85	77.50	99.15	38.35	99.60
	HHT CNN	68.75	76.00	99.05	31.40	99.80
TIRBM	1D to 2D CNN	63.75	69.15	98.75	32.25	99.60
	STFT CNN	51.05	78.80	98.45	32.40	99.65
	Wavelet CNN	59.00	79.90	99.05	36.60	99.60
	HHT CNN	65.70	77.50	**99.35**	33.75	99.75
PGBM	1D to 2D CNN	67.10	69.95	98.85	34.30	99.60
	STFT CNN	45.55	77.35	98.60	34.20	99.65
	Wavelet CNN	60.05	79.95	98.90	35.70	99.60
	HHT CNN	61.85	78.45	99.15	35.70	99.75
ScatNet-2	1D to 2D CNN	63.35	69.25	98.80	31.10	99.60
	STFT CNN	48.50	78.15	98.55	**39.05**	99.65
	Wavelet CNN	58.85	79.10	99.10	35.25	99.60
	HHT CNN	65.80	76.80	99.30	31.05	99.75
RandN-2	1D to 2D CNN	66.45	70.05	98.65	32.50	99.60
	STFT CNN	49.90	77.25	98.40	33.70	99.65
	Wavelet CNN	64.75	78.40	98.95	35.25	99.60
	HHT CNN	61.70	76.30	99.20	33.90	99.75
LDANet-2	1D to 2D CNN	69.05	69.85	98.60	30.45	99.60
	STFT CNN	50.25	76.30	98.40	37.45	99.60
	Wavelet CNN	60.25	79.70	99.00	37.00	99.60
	HHT CNN	68.60	77.10	99.30	34.80	99.75
NNet	1D to 2D CNN	70.75	68.90	98.60	31.30	99.60
	STFT CNN	50.60	78.65	98.55	38.55	99.65
	Wavelet CNN	70.20	79.55	99.05	38.40	99.60
	HHT CNN	67.15	76.30	99.05	33.00	99.75
SQNet	1D to 2D CNN	63.80	69.95	98.70	31.55	99.60
	STFT CNN	53.40	78.00	98.55	34.95	99.65
	Wavelet CNN	64.15	78.35	99.25	34.35	99.60
	HHT CNN	63.00	74.75	99.15	35.75	99.80
MobileNet	1D to 2D CNN	64.40	69.60	98.80	33.00	99.60
	STFT CNN	45.55	78.55	98.45	35.85	99.65
	Wavelet CNN	66.65	**82.20**	99.00	35.60	99.60
	HHT CNN	67.75	76.45	99.15	35.75	99.80
DBN-3	1D to 2D CNN	75.50	68.70	98.65	34.65	99.60
	STFT CNN	49.15	76.90	98.45	35.75	99.65
	Wavelet CNN	64.75	77.60	99.25	34.25	99.60
	HHT CNN	69.00	77.05	99.15	34.55	99.75

is because the proposed algorithm designs the optimal architecture for the given problem, while the existing architectures are generic and are not particularly tuned for a given problem.

Table 14 summarizes the accuracy of different algorithms (Eq. (4)) on the Three-Phase Flow Facility benchmark (case 2) for different CNN architectures. As the data suggest, in most cases, the proposed algorithm outperforms the existing architectures.

Table 15 presents the performance of different architectures in terms of accuracy, true positive rate, and positive predictive rate on the Motor Bearing Fault Diagnosis, the Self-Priming Centrifugal Pump Fault Diagnosis, and Axial Piston Hydraulic Pump Fault Diagnosis benchmarks for different CNN architectures. The data suggest that the proposed algorithm outperforms the existing architectures in terms of ACC, TPR and PPV in many cases.

5. Conclusion

5.1. Conclusional remarks

There are many feature extraction approaches proposed in the literature for fault diagnosis problem. While each of these sets of features has its own advantages, there is not much research in the literature to develop a method to take advantage of all these approaches. In this paper, we propose an ensemble learning algorithm that combines a number of base learner algorithms, where each of these base learners uses a particular set of features. The features used in this paper are extracted from time-domain, Fourier transforms, Wavelet transforms, Walsh transforms and Hilbert–Huang transforms. Two main approaches are used for feature extraction. One is a set of statistical features, which include mean, Kurtosis, Entropy estimation, etc of the original signal. The other approach is to use a Convolutional Neural Network to automatically extract features from these signals. Using CNNs to extract features from signals (particularly images) is very popular due to its potentials. This paper uses CNNs to extract automatically features from time-domain and frequency-domain signals. Each set of features is used to train a base learner algorithm, the output of which is used in a weighted voting paradigm. Finding the optimal weight of the votes is an optimization process that is performed via an evolutionary algorithm in this paper. The process is not much time consuming as the output of the classifiers for each of the training data can be stored in a lookup table which makes evaluating the fitness of a weight vector very quick.

The structure of the CNNs has a great deal of effect on their performance. This paper proposes an evolutionary algorithm to find the best structure for CNNs in extracting features for fault diagnosis. We could use the generic architectures that are used in the literature for different problems; however, we believe that there is no optimal architecture that suits all the problems. The existing CNN architectures in the literature are tuned for particular or generic problems and are not necessarily the optimal ones for fault diagnosis problem. The proposed approach in this paper has the advantage that it fine-tunes CNNs for not only fault diagnosis problem but also different signals. That is, CNN architectures for wavelet, short-term Fourier transform, etc. signals are optimized independently.

This work shows that employing new features and using ensemble methods can improve the performance of fault diagnosis algorithms. Because fault diagnosis is a very important problem in many companies, adopting this method can increase the rate of fault detection and thus reduce the costs due to system failure. The proposed method requires adequate processing units as it employs sophisticated feature extraction methods in combination with a number of learning algorithms. Using this system requires proper computational systems and cannot be implemented with simple micro-processing units. This should be taken into account, although it would not prove problematic for most industrial companies as today proper computers are easily available. Implementation of this system can be improved by the use of application-specific processing units. For example, applying Fourier, wavelet and other forms of signal processing tasks could be efficiently performed with fast, low cost, and readily available micro-chips. Some of the learning algorithms can also be found readily available in the form of VLSI.

5.2. Future work and discussion

There are still a number of open questions that can be targeted in future work. The learning process in CNNs is an optimization problem which was performed in this paper via a combination of gradient descent and evolutionary operators. It is known in the community of optimization that studying and understanding the fitness landscape of problems can help develop better optimization algorithms (Prugel-Bennett & Tayarani-Najaran, 2012; Tayarani-N. & Prügel-Bennett, 2014; Tayarani-N. & Prügel-Bennett, 2015, 2016). As future work, studying the fitness

Table 15
The performance of different algorithms in terms of accuracy, true positive rate, and positive predictive rate on the Motor Bearing Fault Diagnosis, the Self-Priming Centrifugal Pump Fault Diagnosis, and Axial Piston Hydraulic Pump Fault Diagnosis benchmarks for different CNN architectures. The data are averaged over 30 runs.

Algorithm		Case 3 (TSP)			Case 4 (SPCP)			Case 5 (APHP)		
		ACC	TPR	PPV	ACC	TPR	PPV	ACC	TPR	PPV
A1	1D to 2D CNN	99.67	99.65	99.84	99.39	99.21	99.40	99.73	99.82	99.90
	STFT CNN	99.54	99.39	99.77	99.28	99.34	99.27	99.99	99.89	99.83
	Wavelet CNN	99.71	99.81	99.74	99.42	99.32	99.32	99.79	99.77	100.00
	HHT CNN	99.76	99.68	99.75	99.27	99.35	99.16	99.82	99.93	99.74
LSTM	1D to 2D CNN	99.65	99.59	99.67	99.25	99.18	99.40	99.76	99.74	99.76
	STFT CNN	99.49	99.47	99.60	99.22	99.22	99.30	99.82	99.87	99.78
	Wavelet CNN	99.56	99.51	99.66	99.28	99.23	99.32	99.73	99.75	99.74
	HHT CNN	99.68	99.77	99.62	99.22	99.27	99.22	99.80	99.85	99.87
SAAA-3	1D to 2D CNN	99.59	99.56	99.64	99.24	99.16	99.35	99.75	99.74	99.74
	STFT CNN	99.46	99.45	99.58	99.28	99.22	99.26	99.83	99.80	99.78
	Wavelet CNN	99.66	99.61	99.61	99.29	99.25	99.33	99.80	99.75	99.78
	HHT CNN	99.69	99.64	99.62	99.25	99.30	99.30	99.88	99.83	99.87
TIRBM	1D to 2D CNN	99.64	99.53	99.66	99.26	99.19	99.32	99.72	99.74	99.74
	STFT CNN	99.47	99.44	99.53	99.20	99.28	99.26	99.85	99.86	99.80
	Wavelet CNN	99.63	99.51	99.69	99.27	99.29	99.22	99.72	99.77	99.70
	HHT CNN	99.65	99.64	99.65	99.22	99.21	99.22	99.89	99.80	99.80
PGBM+DN-1	1D to 2D CNN	99.69	99.60	99.79	99.28	99.18	99.32	99.79	99.73	99.78
	STFT CNN	99.48	99.47	99.52	99.30	99.21	99.29	99.83	99.84	99.72
	Wavelet CNN	99.64	99.52	99.67	99.33	99.25	99.34	99.72	99.73	99.71
	HHT CNN	99.68	99.71	99.61	99.24	99.23	99.23	99.81	99.84	99.89
ScatNet-2	1D to 2D CNN	99.64	99.55	99.70	99.30	99.19	99.40	99.78	99.78	99.77
	STFT CNN	99.48	99.48	99.55	99.21	99.27	99.21	99.81	99.86	99.71
	Wavelet CNN	99.61	99.65	99.64	99.33	99.25	99.34	99.77	99.77	99.73
	HHT CNN	99.64	99.61	99.69	99.23	99.29	99.30	99.85	99.82	99.82
RandNet-2	1D to 2D CNN	99.64	99.59	99.63	99.29	99.14	99.34	99.77	99.77	99.78
	STFT CNN	99.50	99.45	99.54	99.28	99.24	99.25	99.79	99.85	99.71
	Wavelet CNN	99.64	99.70	99.70	99.22	99.25	99.29	99.76	99.72	99.77
	HHT CNN	99.73	99.80	99.62	99.21	99.29	99.24	99.88	99.87	99.89
LDANet-2	1D to 2D CNN	99.64	99.51	99.74	99.27	99.16	99.31	99.71	99.74	99.79
	STFT CNN	99.47	99.50	99.55	99.20	99.22	99.30	99.82	99.88	99.75
	Wavelet CNN	99.68	99.65	99.60	99.25	99.30	99.34	99.78	99.78	99.74
	HHT CNN	99.72	99.75	99.65	99.23	99.23	99.28	99.89	99.82	99.83
NNet	1D to 2D CNN	99.69	99.57	99.77	99.22	99.20	99.37	99.72	99.75	99.73
	STFT CNN	99.46	99.43	99.53	99.28	99.21	99.24	99.78	99.89	99.79
	Wavelet CNN	99.64	99.57	99.61	99.32	99.28	99.32	99.74	99.74	99.75
	HHT CNN	99.67	99.60	99.63	99.26	99.27	99.25	99.87	99.88	99.89
SQNet	1D to 2D CNN	99.58	99.54	99.61	99.25	99.17	99.39	99.73	99.79	99.75
	STFT CNN	99.50	99.43	99.55	99.24	99.30	99.25	99.79	99.88	99.71
	Wavelet CNN	99.60	99.57	99.68	99.31	99.26	99.33	99.76	99.78	99.78
	HHT CNN	99.70	99.72	99.68	99.23	99.21	99.27	99.85	99.87	99.81
MobileNetV2	1D to 2D CNN	99.62	99.55	99.61	99.21	99.16	99.33	99.79	99.73	99.73
	STFT CNN	99.52	99.50	99.51	99.25	99.20	99.20	99.85	99.84	99.71
	Wavelet CNN	99.69	99.68	99.64	99.25	99.22	99.34	99.79	99.70	99.76
	HHT CNN	99.68	99.78	99.66	99.27	99.21	99.24	99.84	99.84	99.81
DBN-3	1D to 2D CNN	99.62	99.57	99.62	99.21	99.12	99.38	99.75	99.76	99.76
	STFT CNN	99.52	99.43	99.54	99.26	99.23	99.26	99.75	99.84	99.76
	Wavelet CNN	99.56	99.59	99.63	99.24	99.27	99.25	99.78	99.76	99.78
	HHT CNN	99.68	99.63	99.65	99.21	99.21	99.28	99.88	99.85	99.87

landscape of the learning process in CNNs can be conducted which can provide insight into how to avoid the local optima problem and to have a better chance of finding better optima.

In this paper, we used a wide variety of features including the statistical features from the original signal, automatically extracted featured via CNNs, and transform functions on the signal to classify the data. However, clearly, not all these features, especially the CNN features are not necessarily discriminative enough to improve the classification process. As future work, a feature analysis research to find the best set of features among a wide variety of signal processing approaches can be performed. A feature selection research, a filter or wrapper paradigm, can provide insights into finding the best signal processing and feature extraction methods.

The optimization process in evolutionary architecture design requires the measurement of the fitness of each architectural design (individual). In the proposed algorithm (and in the algorithms proposed in the literature), in order to find the fitness of an individual, a CNN is constructed based on the architecture that the individual suggests. Then the CNN is trained, tested and its performance in classification is used as the fitness. This process involves some uncertainty, stemming from the fact that the training process is the non-deterministic process of gradient descent algorithms. Managing the uncertainty in the evolutionary design of CNN architectures is a line of research for future work.

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.

Data availability

Data will be made available on request.

References

Azamfar, M., Singh, J., Bravo-Imaz, I., & Lee, J. (2020). Multisensor data fusion for gearbox fault diagnosis using 2-D convolutional neural network and motor current signature analysis. *Mechanical Systems and Signal Processing*, *144*, Article 106861.

Baldominos, A., Saez, Y., & Isasi, P. (2018). Evolutionary convolutional neural networks: An application to handwriting recognition. *Neurocomputing*, *283*, 38–52.

Ben Ali, J., Fnaiech, N., Saidi, L., Chebel-Morello, B., & Fnaiech, F. (2015). Application of empirical mode decomposition and artificial neural network for automatic bearing fault diagnosis based on vibration signals. *Applied Acoustics*, *89*, 16–27.

Benediktsson, J. A., Sveinsson, J. R., Ersoy, O. K., & Swain, P. H. (1997). Parallel consensual neural networks. *IEEE Transactions on Neural Networks*, *8*(1), 54–64.

Bingham, G., Macke, W., & Miikkulainen, R. (2020). Evolutionary optimization of deep learning activation functions. In *Proceedings of the 2020 genetic and evolutionary computation conference* (pp. 289–296).

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (1996b). Stacked regressions. *Machine Learning*, *24*(1), 49–64.

Breiman, L. (1999). Pasting small votes for classification in large databases and on-line. *Machine Learning*, *36*(1–2), 85–103.

Bruna, J., & Mallat, S. (2013). Invariant scattering convolution networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, *35*(8), 1872–1886.

Bryll, R., Gutierrez-Osuna, R., & Quek, F. (2003). Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, *36*(6), 1291–1302.

Bühlmann, P., Yu, B., et al. (2002). Analyzing bagging. *The Annals of Statistics*, *30*(4), 927–961.

Buja, A., & Stuetzle, W. (2006). Observations on bagging. *Statistica Sinica*, 323–351.

Burriel-Valencia, J., Puche-Panadero, R., Martinez-Roman, J., Sapena-Bano, A., & Pineda-Sanchez, M. (2017). Short-frequency Fourier transform for fault diagnosis of induction machines working in transient regime. *IEEE Transactions on Instrumentation and Measurement*, *66*(3), 432–440.

Cai, Q.-T., Peng, C.-Y., & Zhang, C.-S. (2008). A weighted subspace approach for improving bagging performance. In *Acoustics, speech and signal processing, 2008. ICASSP 2008. IEEE international conference on* (pp. 3341–3344). IEEE.

Cao, P., Zhang, S., & Tang, J. (2018). Preprocessing-free gear fault diagnosis using small datasets with deep convolutional neural network-based transfer learning. *Ieee Access*, *6*, 26241–26253.

Chan, T., Jia, K., Gao, S., Lu, J., Zeng, Z., & Ma, Y. (2015). PCANet: A simple deep learning baseline for image classification?. *IEEE Trans. Image Process.*, *24*(12), 5017–5032. http://dx.doi.org/10.1109/TIP.2015.2475625.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 1–27.

Chen, C. P., & Liu, Z. (2017). Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(1), 10–24.

clu (2022). University of Hertfordshire computing cluster. https://uhhpc.herts.ac.uk/wiki/index.php/Architecture.

Cruz, Y. J., Rivas, M., Quiza, R., Villalonga, A., Haber, R. E., & Beruvides, G. (2021). Ensemble of convolutional neural networks based on an evolutionary algorithm applied to an industrial welding process. *Computers in Industry*, *133*, Article 103530.

DeepLearn toolbox. (2015). https://github.com/rasmusbergpalm/DeepLearnToolbox.

Downs, J., & Vogel, E. (1993). A plant-wide industrial process control problem. *Computers & Chemical Engineering*, *17*(3), 245–255, Industrial challenge problems in process control.

Fernandes Junior, F. E., & Yen, G. G. (2019b). Particle swarm optimization of deep neural networks architectures for image classification. *Swarm and Evolutionary Computation*, *49*, 62–74. http://dx.doi.org/10.1016/j.swevo.2019.05.010, URL: https://www.sciencedirect.com/science/article/pii/S2210650218309246.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. New York, USA: John Wiley.

Gawali, N. U., Hasabe, R., & Vaidya, A. (2015). A comparison of different mother wavelet for fault detection & classification of series compensated transmission line. *International Journal of the Innovations Research Science and Technology*, *1*(9), 57–63.

Gottapu, R. D., & Dagli, C. H. (2020). Efficient architecture search for deep neural networks. *Proc. Comput. Sci.*, *168*, 19–25.

Guo, X., Chen, L., & Shen, C. (2016). Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis. *Measurement*, *93*, 490–502.

Guo, M., Yang, N., & Chen, W. (2019). Deep-learning-based fault classification using Hilbert–Huang transform and convolutional neural network in power distribution systems. *IEEE Sensors Journal*, *19*(16), 6905–6913.

Haidong, S., Junsheng, C., Hongkai, J., Yu, Y., & Zhantao, W. (2020). Enhanced deep gated recurrent unit and complex wavelet packet energy moment entropy for early fault prognosis of bearing. *Knowledge-Based Systems*, *188*, Article 105022.

Han, Y., Tang, B., & Deng, L. (2018). Multi-level wavelet packet fusion in dynamic ensemble convolutional neural network for fault diagnosis. *Measurement*, *127*, 246–255.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(10), 993–1001.

Hashem, S. (1997). Optimal linear combinations of neural networks. *Neural Networks*, *10*(4), 599–614.

He, Z., Shao, H., Zhang, X., Cheng, J., & Yang, Y. (2019). Improved deep transfer auto-encoder for fault diagnosis of gearbox under variable working conditions with small training samples. *IEEE Access*, *7*, 115368–115377.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844.

Hoang, D.-T., & Kang, H.-J. (2019). Rolling element bearing fault diagnosis using convolutional neural network and vibration image. *Cognitive Systems Research*, *53*, 42–50.

Holland, J. H. (1975). *Adaption in Natural and Artificial Systems* (first ed.). Ann Arbor MI.

Hu, Q., He, Z., Zhang, Z., & Zi, Y. (2007). Fault diagnosis of rotating machinery based on improved wavelet package transform and SVMs ensemble. *Mechanical Systems and Signal Processing*, *21*(2), 688–705.

Hu, Q., Qin, A., Zhang, Q., He, J., & Sun, G. (2018). Fault diagnosis based on weighted extreme learning machine with wavelet packet decomposition and KPCA. *IEEE Sensors Journal*, *18*(20), 8472–8483.

Huang, W., Cheng, J., Yang, Y., & Guo, G. (2019). An improved deep convolutional neural network with multi-scale information for bearing fault diagnosis. *Neurocomputing*, *359*, 77–92.

Huang, R., Liao, Y., Zhang, S., & Li, W. (2018). Deep decoupling convolutional neural network for intelligent compound fault diagnosis. *IEEE Access*, *7*, 1848–1858.

Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 1MB model size. CoRR.

Jiao, J., Zhao, M., Lin, J., & Zhao, J. (2018). A multivariate encoder information based convolutional neural network for intelligent fault diagnosis of planetary gearboxes. *Knowledge-Based Systems*, *160*, 237–250.

Jing, L., Zhao, M., Li, P., & Xu, X. (2017). A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox. *Measurement*, *111*, 1–10.

Johner, F. M., & Wassner, J. (2019). Efficient evolutionary architecture search for CNN optimization on GTSRB. In *2019 18th IEEE international conference on machine learning and applications (ICMLA)* (pp. 56–61). http://dx.doi.org/10.1109/ICMLA.2019.00018.

Junior, F. E. F., & Yen, G. G. (2019a). Particle swarm optimization of deep neural networks architectures for image classification. *Swarm Evol. Comput.*, *49*, 62–74.

Kennedy, J., Eberhart, R. C., & Shi, Y. H. (2001). *The Morgan Kaufmann Series in Evolutionary Computation, Swarm Intelligence* (first ed.). Morgan Kaufmann.

Konar, P., & Chattopadhyay, P. (2015). Multi-class fault diagnosis of induction motor using Hilbert and wavelet transform. *Applied Soft Computing*, *30*, 341–352.

Kuncheva, L. I. (2005). *Diversity in multiple classifier systems*. Elsevier.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning* (pp. 473–480).

Li, C., & Liang, M. (2012). Time–frequency signal analysis for gearbox fault diagnosis using a generalized synchrosqueezing transform. *Mechanical Systems and Signal Processing*, *26*, 205–217.

Liang, P., Deng, C., Wu, J., Yang, Z., Zhu, J., & Zhang, Z. (2019). Compound fault diagnosis of gearboxes via multi-label convolutional neural network and wavelet transform. *Computers in Industry*, *113*, Article 103132.

Lindu, Z., & Zhaohan, S. (1996). Combination of discrete cosine transform with neural network in fault diagnosis for rotating machinery. In *Proceedings of the IEEE international conference on industrial technology (ICIT'96)* (pp. 450–454). IEEE.

Liu, Y., Mu, Y., Chen, K., Li, Y., & Guo, J. (2020). Daily activity feature selection in smart homes based on pearson correlation coefficient. *Neural Processing Letters*, 1–17.

Liu, H., Zhou, J., Xu, Y., Zheng, Y., Peng, X., & Jiang, W. (2018). Unsupervised fault diagnosis of rolling bearings using a deep neural network based on generative adversarial networks. *Neurocomputing*, *315*, 412–424.

Lou, X., & Loparo, K. A. (2004). Bearing fault diagnosis based on wavelet transform and fuzzy inference. *Mechanical Systems and Signal Processing*, *18*(5), 1077–1095.

Lu, X., Lin, P., Cheng, S., Lin, Y., Chen, Z., Wu, L., et al. (2019). Fault diagnosis for photovoltaic array based on convolutional neural network and electrical time series graph. *Energy Conversion and Management*, *196*, 950–965.

Lu, S., Yan, R., Liu, Y., & Wang, Q. (2019). Tacholess speed estimation in order tracking: A review with application to rotating machine fault diagnosis. *IEEE Transactions on Instrumentation and Measurement*, *68*(7), 2315–2332.

Ma, B., Li, X., Xia, Y., & Zhang, Y. (2020). Autonomous deep learning: a genetic DCNN designer for image classification. *Neurocomputing*, *379*, 152–161.

Mo, H., Custode, L. L., & Iacca, G. (2021). Evolutionary neural architecture search for remaining useful life prediction. *Applied Soft Computing*, *108*, Article 107474. http://dx.doi.org/10.1016/j.asoc.2021.107474, URL: https://www.sciencedirect.com/science/article/pii/S1568494621003975.

Molina, D., Lozano, M., Sánchez, A. M., & Herrera, F. (2011). Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SSW-chains. *Soft Comput.*, *15*, 2201–2220.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, *6*(3), 21–45.

Prugel-Bennett, A., & Tayarani-Najaran, M. (2012). Maximum satisfiability: Anatomy of the fitness landscape for a hard combinatorial optimization problem. *IEEE Transactions on Evolutionary Computation*, *16*(3), 319–338. http://dx.doi.org/10.1109/TEVC.2011.2163638.

Rai, V., & Mohanty, A. (2007). Bearing fault diagnosis using FFT of intrinsic mode functions in Hilbert–huang transform. *Mechanical Systems and Signal Processing*, *21*(6), 2607–2615.

Ranawana, R., & Palade, V. (2006). Multi-classifier systems: Review and a roadmap for developers. *International Journal of Hybrid Intelligent Systems*, *3*(1), 35–61.

Rifai, S., Vincent, P., Muller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders:Explicit invariance during feature extraction. In *Icml*.

Ruiz-Cárcel, C., Cao, Y., Mba, D., Lao, L., & Samuel, R. (2015). Statistical process monitoring of a multiphase flow facility. *Control Engineering Practice*, *42*, 74–88.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Wiley.

Shirai, S., Kudo, M., & Nakamura, A. (2008). Bagging, random subspace method and biding. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)* (pp. 801–810). Springer.

Smith, W. A., & Randall, R. B. (2015). Rolling element bearing diagnostics using the case western reserve university data: A benchmark study. *Mechanical Systems and Signal Processing*, *64*, 100–131.

Sohn, K., & Lee, H. (2012). Learning invariant representations with local transformations. arXiv:1206.6418.

Stefanowski, J. (2007). Combining answers of sub-classifiers in the bagging-feature ensembles. In *International conference on rough sets and intelligent systems paradigms* (pp. 574–583). Springer.

Sun, Y., Xue, B., Zhang, M., & Yen, G. G. (2018). A particle swarm optimization-based flexible convolutional autoencoder for image classification. *IEEE Trans. Neural Netw. Learn. Syst.*, *30*(8), 2295–2309.

Sun, Y., Xue, B., Zhang, M., & Yen, G. G. (2020). Completely automated CNN architecture design based on blocks. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(4), 1242–1254. http://dx.doi.org/10.1109/TNNLS.2019.2919608.

Tayarani-N., M., & Prügel-Bennett, A. (2014). On the landscape of combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, *18*(3), 420–434.

Tayarani-N., M.-H., & Prügel-Bennett, A. (2015). Anatomy of the fitness landscape for dense graph-colouring problem. *Swarm and Evolutionary Computation*, *22*, 47–65.

Tayarani-N., M.-H., & Prügel-Bennett, A. (2016). An analysis of the fitness landscape of travelling salesman problem. *Evolutionary Computation*, *24*(2), 347–384, PMID: 26066806.

Tran, V. T., AlThobiani, F., Ball, A., & Choi, B.-K. (2013). An application to transient current signal based induction motor fault diagnosis of Fourier–bessel expansion and simplified fuzzy ARTMAP. *Expert Systems with Applications*, *40*(13), 5372–5384.

Uddamvathanak, R., Yang, F., Yang, X., Das, A. K., Shen, Y., Salahuddin, M., et al. (2018). Two-stage ensemble of deep convolutional neural networks for object recognition. In *2018 international conference on intelligent rail transportation (ICIRT)* (pp. 1–5). IEEE.

Wang, Y., Ma, Q., Zhu, Q., Liu, X., & Zhao, L. (2014). An intelligent approach for engine fault diagnosis based on Hilbert–Huang transform and support vector machine. *Applied Acoustics*, *75*, 1–9.

Wen, L., Li, X., Gao, L., & Zhang, Y. (2017). A new convolutional neural network-based data-driven fault diagnosis method. *IEEE Transactions on Industrial Electronics*, *65*(7), 5990–5998.

Wu, C., Jiang, P., Ding, C., Feng, F., & Chen, T. (2019). Intelligent fault diagnosis of rotating machinery based on one-dimensional convolutional neural network. *Computers in Industry*, *108*, 53–61.

Xiang, X., Zhou, J., Li, C., Li, Q., & Luo, Z. (2009). Fault diagnosis based on walsh transform and rough sets. *Mechanical Systems and Signal Processing*, *23*(4), 1313–1326.

Xie, Y., Chen, H., Ma, Y., & Xu, Y. (2022). Automated design of CNN architecture based on efficient evolutionary search. *Neurocomputing*, *491*, 160–171. http://dx.doi.org/10.1016/j.neucom.2022.03.046, URL: https://www.sciencedirect.com/science/article/pii/S092523122200340X.

Yan, R., Gao, R. X., & Chen, X. (2014). Wavelets for fault diagnosis of rotary machines: A review with applications. *Signal Processing*, *96*, 1–15, Time-frequency methods for condition based maintenance and modal analysis.

Yao, X., & Liu, Y. (1997). Fast evolution strategies. *Control Cybern.*, *26*, 467–496.

Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Trans. Evol. Comput.*, *3*(2), 82–102.

Yu, W., & Zhao, C. (2019). Broad convolutional neural network based industrial process fault diagnosis with incremental learning capability. *IEEE Transactions on Industrial Electronics*, *67*(6), 5081–5091.

Zhang, Z., Wang, Y., & Wang, K. (2013). Fault diagnosis and prognosis using wavelet packet decomposition, Fourier transform and artificial neural network. *Journal of Intelligent Manufacturing*, *24*(6), 1213–1227.

Zhang, Y., Xing, K., Bai, R., Sun, D., & Meng, Z. (2020). An enhanced convolutional neural network for bearing fault diagnosis based on time–frequency image. *Measurement*, *157*, Article 107667.

Zhao, H., Liu, H., Xu, J., & Deng, W. (2019). Performance prediction using high-order differential mathematical morphology gradient spectrum entropy and extreme learning machine. *IEEE Transactions on Instrumentation and Measurement*.

Zhao, H., Sun, S., & Jin, B. (2018). Sequential fault diagnosis based on LSTM neural network. *IEEE Access*, *6*, 12929–12939.

Zhu, Z., Peng, G., Chen, Y., & Gao, H. (2019). A convolutional neural network based on a capsule network with strong generalization for bearing fault diagnosis. *Neurocomputing*, *323*, 62–75.

Mohammad- H. Tayarani- N. received his Ph.D. degree from the University of Southampton, Southampton, U.K, in 2013. Then he worked as research fellow at the University of Birmingham, Birmingham, UK and University of Glasgow, Glasgow, UK. He currently holds a fellowship at the university of Hertfordshire, Hatfield, UK. His main research interests include evolutionary algorithms, machine learning, and fractal image compression.