www.czasopisma.pan.pl PAN www.journals.pan.pl

Metrol. Meas. Syst., Vol. 24 (2017), No. 4, pp. 701–720.

METROLOGY AND MEASUREMENT SYSTEMS

Index 330930, ISSN 0860-8229

www.metrology.pg.gda.pl

A FAST CLASSIFICATION METHOD OF FAULTS IN POWER ELECTRONIC CIRCUITS BASED ON SUPPORT VECTOR MACHINES

Jiang Cui, Ge Shi, Chunying Gong

Nanjing University of Aeronautics and Astronautics, College of Automation Engineering, 29 Jiangjun Road, Jiangning, Nanjing, Jiangsu, China (✉ pushriver@sina.com, +86 153 6600 6770, *SG1503018@126.com, zjnjgcy@nuaa.edu.pl)*

Abstract

Fault detection and location are important and front-end tasks in assuring the reliability of power electronic circuits. In essence, both tasks can be considered as the classification problem. This paper presents a fast fault classification method for power electronic circuits by using the *support vector machine* (SVM) as a classifier and the wavelet transform as a feature extraction technique. Using one-against-rest SVM and one-against-one SVM are two general approaches to fault classification in power electronic circuits. However, these methods have a high computational complexity, therefore in this design we employ a *directed acyclic graph* (DAG) SVM to implement the fault classification. The DAG SVM is close to the one-against-one SVM regarding its classification performance, but it is much faster. Moreover, in the presented approach, the DAG SVM is improved by introducing the method of *K*-nearest neighbours to reduce some computations, so that the classification time can be further reduced. A rectifier and an inverter are demonstrated to prove effectiveness of the presented design.

Keywords: power electronics, fault diagnosis, wavelet transforms, support vector machines, directed acyclic graph, nearest neighbours.

1. Introduction

Power electronic circuits (PECs) can be found extensively in industrial, military and residential applications [1]. High thermal and frequent mechanical stresses during the operations can accelerate the failure process of PECs. Once a fault inside a PEC occurs, unplanned electrical device breakdown may be triggered, in some cases associated with a very high cost or even casualties. For reasons of safety, reliability and maintenance, a fault has to be detected and diagnosed as soon as possible after its occurrence [2]. In some PECs with the fault tolerance capability, fault detection and diagnosis are necessary steps [3].

In essence, fault detection and fault diagnosis fall into the category of fault classification. Fault classification methods of PECs can be classified into three groups: model-based, expert system-based and *artificial intelligence* (AI)-based ones [4]. Among these methods, the AI-based ones seem to be attractive and interesting, because they have some advantages in comparison with other methods [1]. An AI-based method considers the whole system as a black box, whose inside details are being not relevant. This can avoid the problem of circuit system modelling. Classifiers based on the AI technique, such as the fuzzy inference method [5–7], have been proved to be effective. Compared with methods based on pure hardware circuits [8], AI-based methods usually employ algorithms, which can be easy and flexible to transplant and upgrade while keeping unchanged the corresponding hardware .

In the AI applications, the *Neural Network* (NN) is a good classifier, which has good performance in fault classification of PEC. In [9], the *radial-basis-function* (RBF) NN is adopted to perform fault detection of an induction motor drive circuit. A *back-propagation NN* (BPNN) is presented in [10] to diagnose faults of a three-phase inverter. In this example,

Article history: received on Nov. 12, 2016; accepted on May 21, 2017; available online on Dec. 01, 2017; DOI: 10.1515/mms-2017-0056.

the classification accuracy of over 95% is reported. In [11], a multi-layered perceptron network is employed to classify open faults of a simulated *voltage source inverter* (VSI). Another study of BPNN application to a multi-level inverter fault classification is described in [2]. In some studies, two or more NNs are integrated to perform the classification task, which can improve the classification performance of a diagnostic system [12−14]. The NN-based method has also some drawbacks. For example, different NN trainings may lead to different classification results. In addition, high-dimensional data can result in a long training process, or even a convergence failure. Focusing on these drawbacks, the NN classifier can be improved with other methods. For example, in [15], before being input to the classifier, the fault samples are pre-processed with *Principal Component Analysis* (PCA) and *Genetic Algorithm* (GA), which can reduce dimensions of the training samples. In [16], the BPNN structure is optimized to improve its classification performance.

Recently, the applications of *Support Vector Machine* (SVM) to fault classification of PECs have been reported. The SVM has some excellent characteristics, *e.g*. it needs less adjustable parameters and can find the global solution easily during training, thus leading to stable classification results. The conventional SVM can create binary classes, and such a classifier is called a *binary SVM* (BSVM). In [17], one BSVM is used to detect whether the inverter is faulty, and the other BSVM can localize the faulty power switch (upper or bottom half-bridge). Another application of BSVM to the fault detection of an induction motor drive is described in [18]. Generally, diagnosing a PEC involves a multi-class classification. In the domain of machine learning, a multi-class classifier design for SVM involves two methods [19]. The first method is meant to create a multi-class classification in one step, whereas the second one – to combine several BSVMs to form a multi-class classifier, which has three basic forms: one-against-rest SVM, one-against-one SVM and *Directed Acyclic Graph* (DAG) SVM. In diagnosing a PEC, the one-against-rest SVM and one-against-one SVM have been used. For instance, in [20] and [21], the one-against-rest SVM classifiers are adopted to perform fault diagnosis of simulated rectifiers. Two examples of diagnosing inverters with one-against-rest SVMs are described in [22] and [23]. The application of one-against-one SVM to the diagnosis of an induction motor drive can be found in [24]. The use of DAG SVM, however, is seldom reported in fault classification of PECs.

According to the experiment results obtained in [19], both DAG SVM and one-against-one SVM are suitable for practical use, because they can always achieve high accuracy in applications. A one-against-one SVM creates classification with a greater number of computations, so that fault classification is a time-consuming task for this classifier. In this research, we apply a DAG SVM to PEC fault classification and, moreover, we attempt to improve the DAG SVM by employing an additional method. Compared with the conventional DAG SVM and other SVM classifiers, the new method needs less BSVMs to perform fault classification and, accordingly, the classification time can be shortened. Also, the classification accuracy of the presented method is very close to that of the conventional DAG SVM. Hence, the presented method can be considered as an alternative classifier for the DAG SVM. Experiments on a rectifier and an inverter were performed to prove effectiveness of the presented method. For the purpose of comparison, five classifiers were designed and examined regarding their classification accuracy and testing time.

2. Basic theories cocerning SVM classifier

2.1. Support vector machines for binary classification

A standard support vector machine classifier, invented by Vapnik and his colleagues [25], has a theoretical background of statistical learning theory and executes *Structural Risk*

Minimization (SRM) [26]. It can create a binary classification with excellent performance. The standard binary classifier can create both linear and nonlinear classifications. In the domain of fault detection and diagnosis of PECs, the nonlinear classification seems to be more practical. The nonlinear BSVM adopts a mapping function $\psi(\cdot)$, which can map data samples from the measurement space to a high-dimensional space. The binary classes can become linearly separable in the high-dimensional space. This principle is expressed in Fig. 1, where ○ and ● represent **class I** and **class II**, respectively.

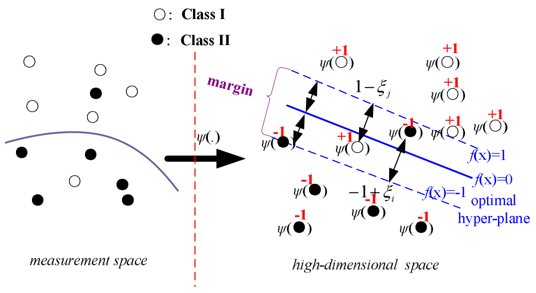


Fig. 1. A basic model of nonlinear BSVM.

In order to implement the BSVM, a margin between samples in the high-dimensional space should be maximized. Assume a data set to be $\{x_i\}$ (i = 1, 2, …, Q, where Q is the number of data samples; $\boldsymbol{x}_i$ is an ith data sample in the measurement space), $\boldsymbol{x}_i \in R^d$ (R being the d-dimensional measurement space). After being mapped to the high-dimensional space with an inexplicit mapping function $\psi(\cdot)$, $\{\boldsymbol{x}_i\}$ turns into $\{\psi(\boldsymbol{x}_i)\}$. Then, each sample $\psi(\boldsymbol{x}_i)$ is assigned to a label y_i ($y_i = +1$ for **Class I**, and y_i=-1 for **Class II**). The optimal hyper-plane can be represented with:

$$f(\boldsymbol{x}_i) = \psi(\boldsymbol{x}_i)\cdot \boldsymbol{w} + b = 0\,, \tag{1}$$

where: $\boldsymbol{w}$ is a weight vector of optimal hyper-plane; b is a bias.

In order to allow some samples to be misclassified to reduce the effect on the decision boundary position, the slack variables $\xi_i \geq 0$ are necessary. Hence, by considering the misclassified samples, the samples try to be classified correctly beyond the margin:

$$y_i(\psi(\boldsymbol{x}_i)\cdot \boldsymbol{w} + b) - 1 + \xi_i \geq 0\,. \tag{2}$$

Maximizing the margin means minimizing the following quadratic optimization problem:

$$\begin{aligned} &\varphi(\boldsymbol{w}) = \|\boldsymbol{w}\|^2 / 2 + C\sum_{i=1}^{Q}\xi_i \\ &s.t. \qquad y_i(\psi(\boldsymbol{x}_i)\cdot \boldsymbol{w} + b) - 1 + \xi_i \geq 0 \end{aligned}\,, \tag{3}$$

where C is a penalty parameter for balancing the classification accuracy and complexity of the decision boundary.

Solving this optimization equation needs the *Lagrange multipliers* (LMs) $\lambda_i \geq 0$:

$$L = \varphi(\boldsymbol{w}) - \sum_{i=1}^{Q}\lambda_i(y_i(\psi(\boldsymbol{x}_i)\cdot \boldsymbol{w} + b) - 1 + \xi_i)\,. \tag{4}$$

By removing primal variables, the partial derivatives of L in respect to $\boldsymbol{w}$ and b are used to yield the dual formulation L^*:

$$L^* = \sum_{i=1}^{Q}\lambda_i - \frac{1}{2}\sum_{i=1}^{Q}\sum_{j=1}^{Q}\lambda_i\lambda_j y_i y_j(\psi(\boldsymbol{x}_i)\cdot\psi(\boldsymbol{x}_j)^T), \tag{5}$$

where: λ_i, λ_j are LMs of *i*th and *j*th data samples, respectively; $(\cdot)$ is an inner product; *T* is the transpose of vector.

The solution of this optimization problem will generate *support vectors* (SVs), whose corresponding LMs are $\lambda_i > 0$. Let the number of SVs be n_{sv} and considering a kernel function $k(\boldsymbol{t},\boldsymbol{x}_k) = (\psi(\boldsymbol{t})\cdot\psi(\boldsymbol{x}_k)^T)$, the calculation function of BSVM becomes:

$$f(\boldsymbol{t}) = \sum_{k=1}^{n_{sv}} y_k \hat{\lambda}_k k(\boldsymbol{t},\boldsymbol{x}_k) + b, \tag{6}$$

where: $\boldsymbol{t}$ is a data sample to be classified; $\hat{\lambda}_k > 0$ is an LM of *k*th SV $\boldsymbol{x}_k$.

The kernel function has several forms [27]. In our experiments, the RBF kernel function $(k(\boldsymbol{t},\boldsymbol{x}_i) = \exp(-|\boldsymbol{t}-\boldsymbol{x}_i|^2/\sigma^2)$, where $\sigma > 0$ is a kernel parameter; $\boldsymbol{x}_i$ is an *i*th SV) is considered, because this nonlinear kernel function can always lead to good classification performance.

2.2. Two conventional multi-class SVMs for PEC fault classification

A conventional one-against-rest SVM employs the *Winner-Takes-All* (WTA) rule to implement a pattern classification [28]. This classifier is simple to use in the fault classification of PEC. For *N* fault classes, *N* BSVMs are needed. For each training, an i^{th} class (labelled with "–1") is separated from the rest (*N*–1) classes (labelled with "+1"). Finally, a sample $\boldsymbol{x}$ should be assigned to the fault class whose corresponding decision function has the minimum value:

$$\arg\min_{i=1,2,\ldots,N}[f_i(\boldsymbol{t})], \tag{7}$$

where $f_i(\boldsymbol{t})$ is a decision function of *i*th BSVM for a sample $\boldsymbol{t}$.

For the one-against-one SVM, altogether *N*(*N*–1)/2 BSVMs are constructed. The decision function for the BSVM, which is formed by classes *i* and *j* ($i \neq j$), can be expressed in the form of:

$$f_{ij}(\boldsymbol{t}) = \sum_{k=1}^{n_{sv}^{i,j}} y_k^{i,j}\lambda_k^{i,j}K(\boldsymbol{t},\boldsymbol{x}_k^{i,j}) + b_{i,j}^*, \tag{8}$$

where: $n_{sv}^{i,j}$ is the number of SVs; $\lambda_k^{i,j}$ is an LM of *k*th SV; $y_k^{i,j}$ is a label of *k*th SV; $b_{i,j}^*$ is a bias of this BSVM.

In the final stage, all decision functions of BSVMs need to vote for the appropriate class. The max-wins strategy is adopted to find a class which wins the maximum votes. However, with the increase of *N*, the number of computations for this classifier will increase drastically, and thus this method will probably become unsuitable for fast fault classification of PECs.

3. Presented SVM classifier

3.1. Typical Structure of DAG SVM

The classifier employed in our research is a typical DAG SVM [29], whose training phase is the same as in the one-against-one SVM by constructing (*N*–1)*N*/2 BSVMs for *N* fault classes. In the classification phase, the BSVMs are arranged to form a directed acyclic graph. Each node of the DAG SVM is represented with B*ij*, indicating a BSVM classifier corresponding to class *i* and class *j* ($i \neq j$). Except for the root node, each node has one input and two outputs which stand for the possible decision values (left and right branch) of the

BSVM. Fig. 2 shows a typical model of DAG SVM with N = 5 (for simplicity, assume five fault classes to be marked: ‘0’, ‘1’, ‘2’, ‘3’ and ‘4’, respectively).

The DAG SVM classifier structure is similar to a pyramid, which can be partitioned into N layers for N fault classes. For instance, in Fig. 2, the root node is the first layer (containing B04), and the second layer contains two nodes (containing B03 and B14), … , the kth layer ($k < N$) contains k BSVMs, …, and so on. The final layer contains only leaves, which represent the five separated classes.

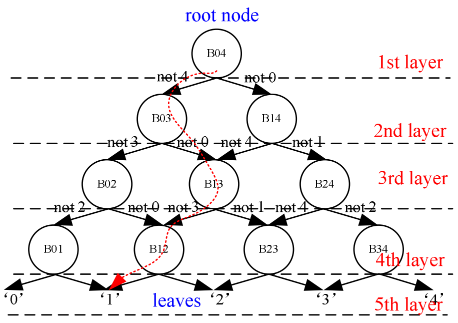


Fig. 2. DAG SVM Structure for N = 5.

The classification task is initiated from the first layer and is stopped in the final layer. In each layer (except for the final layer), one BSVM is evaluated to generate the output result, which becomes the input of a BSVM in the next layer. Fig. 2 gives an illustration of such a flow path (in red dashed curve), in which B04→B03→B13→B12→’1’ are evaluated one after the other. Hence, for N fault classes, (N–1) BSVMs need to be evaluated for each classification task.

3.2. Improvement of DAG SVM

Generally, the DAG SVM is a fast classifier, but it still needs to compute (N–1) decision functions for the BSVMs. A typical DAG SVM always starts from the root node, however in this study we consider changing the starting node of DAG SVM, which will probably reduce the evaluation time of this SVM classifier. For example, in Fig. 2, if the starting node is initiated from B03, not from the root node, the evaluation flow path will become B03→B13→B12→’1’. In this case, the computation of BSVM decision function at B04 node can be bypassed. The key problem is, how to know it is the node B04 that should be avoided. In other words, how to find a limited set of nodes which participate in the computation.

In the paper there is adopted the method of *K-Nearest Neighbours* (K-NN) as an auxiliary classifier to find the limited set of nodes. The K-NN classifier is an easy to use, non-parametric method and. It was applied to the fault diagnosis of a generator rotor [30].

Assume N fault classes, each fault class containing L data samples. The centroid for class j is defined in the measurement space:

$$\boldsymbol{C}_j = \frac{1}{L}\sum_{i=1}^{L}\boldsymbol{x}_{ij}\,, \tag{9}$$

where $\boldsymbol{x}_{ij}$ is an ith training sample of class j (j = 1, 2, …, N).

The K-NN method selects K closest neighbours basing on Euclidean distances between an unknown sample $\boldsymbol{x}$ and the centroids. K fault classes corresponding to the K closest centroids fall into a limited set, from which the starting node can be chosen. Fig. 3 illustrates the way of finding the starting node (N = 5) for K = 3.

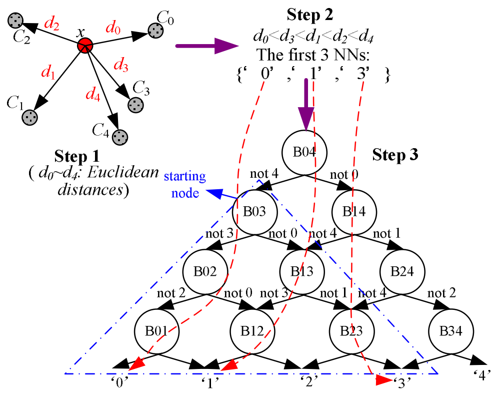


Fig. 3. An illustration of finding the starting node ($N = 5$, $K = 3$).

In this figure, three steps are implemented. Step 1 computes the Euclidean distances between $\boldsymbol{x}$ and the centroids, and five distances are obtained. These distances are sorted in the ascending order in Step 2 and the first 3 nearest neighbours (assumed to be '0', '1' and '3') are selected. In Step 3, we can observe that three selected classes, whose corresponding leaves are indicated in the final layer of DAG SVM in Fig. 3, are derived from the subsidiary DAG, enclosed in a triangle marked by dashed lines. The starting node (*i.e.* B03) is obtained as the root node of the sub graph.

Another way of obtaining the starting node is the use of the information included in indexes. In this case, the index for each class should be predefined and arranged according to the DAG SVM structure. For example, the index for class '0' is 0; for class '1' – 1; …; and so on. The indices i and j for a starting node Bij correspond to the minimal and maximal values of selected K numbers, respectively. Therefore, in a simple way, the starting node B03 can be obtained.

3.3. Classification system design based on SVM

The design of a classification system based on an SVM classifier is similar to that based on a neural network, and the steps of the presented *improved DAG SVM* (*i*DAG SVM) system design for PEC are as follows:

1) Feature extraction. The original current or voltage signals are sampled from the available sensors. These signals contain noise or redundant information, so they need to be pre-processed by signal processing techniques, such as PCA [15, 20, 31], FFT [2, 32], *wavelet transformation* (WT) [6, 7, 17], S-transformation [21], or Concordia transformation [9, 18]. This step generates feature samples, which can be used in the offline training.
2) Offline training of the SVM. Prior to the training, the feature samples need to be assigned with labels (+1 or −1). In our design, in the BSVM training of one-against-rest SVM, the feature samples corresponding to one class are labelled with −1 and the other faults' samples are labelled with +1. For the BSVM training of one-against-one SVM, features for the ith class are labelled with −1, and the features for the jth class are labelled with +1 ($i < j$). After training, for each BSVM, the generated SVM parameters are saved. Also, considering the *i*DAG SVM, the centroid of each class needs to be calculated and saved.
3) Fault classification. Given a new sample, the diagnostic flow is implemented according to Fig. 3.

4. Case studies: rectifier and inverter

4.1. Simulated rectifier

The first circuit is a three-phase full-bridge rectifier with six uncontrolled diodes and a load resistor $R_{load,}$ is shown in Fig. 4. This topology can be used in aerospace power systems and many industrial power electronic converter design applications.

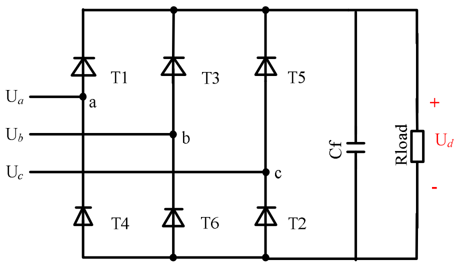


Fig. 4. The simulated three-phase full-bridge rectifier.

This circuit is modelled and simulated with Matlab R2010b-Simulink. In this simulation, the R_{load} value is set to 550 Ω with 10% tolerance (to simulate the load fluctuation) and the filter capacitor Cf value is set to 10 μF: the nominal phase frequency and voltage of the input source (U_a, U_b and U_c) are 400 Hz and 23 V rms, respectively. The output voltage U_d on the load is selected as an accessible signal.

In this circuit, open-circuit faults for the diodes are examined. Faulty diodes and their fault codes are listed in Table 1. $\{T_i, T_j\}$ (i, j = 1,…,6 and $i \neq j$) means that two diodes T_i and T_j are faulty simultaneously. In this simulation, f0, indicating a sound circuit, is regarded as a special class. Hence, twenty two classes are considered. For each fault, this circuit model is simulated 50 times and each time the load value is varied and the corresponding signal U_d is sampled (a sample rate for the simulation is 20 kHz). In this way, altogether 50 samples for each fault can be collected. A randomly selected segment of sample for each fault is shown in Fig. 5.

Table 1. Faults for the rectifier.

Code	Faulty diode(s)	Code	Faulty diode(s)	Code	Faulty diode(s)
f0	–	f8	{T3, T6}	f16	{T1, T2}
f1	T1	f9	{T2, T5}	f17	{T2, T3}
f2	T2	f10	{T1, T3}	f18	{T3, T4}
f3	T3	f11	{T1, T5}	f19	{T4, T5}
f4	T4	f12	{T2, T4}	f20	{T5, T6}
f5	T5	f13	{T2, T6}	f21	{T6, T1}
f6	T6	f14	{T3, T5}		
f7	{T1, T4}	f15	{T4, T6}		

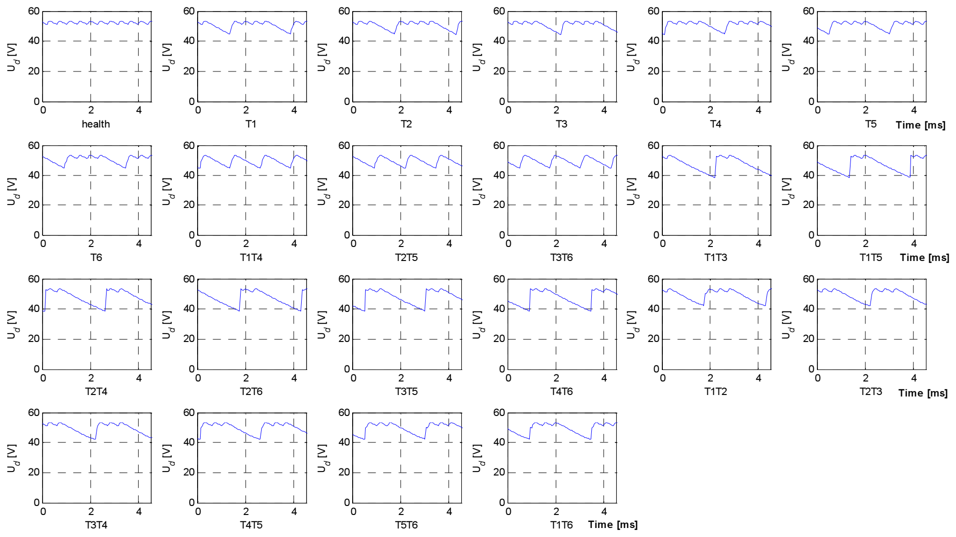


Fig. 5. Waveforms of faults for the simulated rectifier.

Feature extraction. In this example, the WT method is applied to U_d waveforms of each fault class to extract features. The WT is a useful technique [33] that can be used to decompose the collected data into the time-frequency domain, in which coarse coefficients and detail coefficients can be obtained. A simplified diagram of WT decomposition tree is shown in Fig. 6. The coarse coefficients in the low frequency band can indicate the outline of waveform. Generally, different waveforms, indicating different fault classes, can have different coarse coefficients. Hence, the coarse coefficients are selected as the fault features in our research. The detail coefficients in the high frequency band, however, are not considered as features, because these coefficients can be easily corrupted by noise.

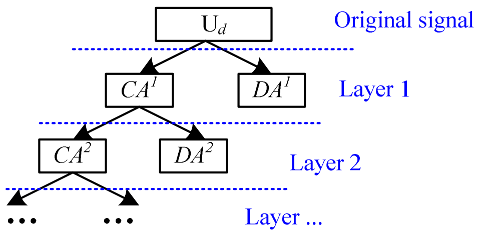


Fig. 6. A simplified diagram of WT decomposition tree.

The steps of extracting fault features with WT are as follows:

a) Apply WT to U_d, assumed to have N data points, and in each layer I ($I = 1, 2, \ldots$) the coarse coefficients $CA^I = \{ CA^I(k) \}$ ($k = 1, 2, \ldots, N/2I$) can be obtained after wavelet decomposition. Also, in WT applications, discussion of the number of decomposition layers and the mother function is inevitable. In our research, we determined these parameters by comparing different experiment results. Finally, *'Haar'* was selected as the mother function of WT for 5-layered decomposition, and these parameters can lead to good experiment results.

b) In each decomposition layer I, calculate the mean value of CA^I with the following equation:

$$Me^I = \frac{2^I}{N} \cdot \sum_{k=1}^{N/2^I} [CA^I(k)]. \tag{10}$$

Find the maximum value Mx^I from $(CA^I \text{-} Me^I)$, according to the following equation:

$$Mx^I = \max[\text{abs}\{CA^I(k) - Me^I\}]. \tag{11}$$

c) Normalize the coarse coefficients with 2-norm. In this case, normalization of coefficients can reduce the effect of load fluctuation. This step can be accomplished as follows:

$$\text{nCA}^I = \left\| \{\frac{CA^I(k) - Me^I}{Mx^I}\} \right\|_2, \tag{12}$$

$$\left(k = 1,\ 2,\ \ldots,\ N/2I\ \right).$$

d) Calculate the feature nCA^I in layer I. In all, a fault feature vector $\mathbf{E}\text{=}[\text{nCA}^1, \text{nCA}^2, \text{nCA}^3, \text{nCA}^4, \text{nCA}^5]$ can be extracted. Finally, the feature vector **E** should be normalized to have zero mean and unity variance. In the machine learning domain, this is a commonly used means which can avoid a large data range.

Result. The Matlab codes for all classifiers were run on a P4 personal computer with 2.6 GHz dual CPUs and 2 GB RAM. Our operating system is Windows XP.

The feature set is split into two parts: a training set and a testing set. The training set contains 10 samples of each class, whereas 40 samples of each class are used for testing. A penalty parameter for SVM is set to 100, and σ is varied across a range {1, 2, 4, 8, 16, 32}. With these parameters, each SVM classifier can generate six results; the best result for each SVM is recorded. In this experiment, all SVM classifiers can achieve 100% testing accuracy when $\sigma = 1$. The *i*DAG SVM needs to confirm the value of *K*. In this research, we performed an exhaustive offline searching for *K*, ranging from 2 to 22, to find a suitable inflection point. In each search, the *K* value was changed, the *i*DAG SVM was used to evaluate the testing set, and accuracy was recorded. The testing accuracy, as well as the testing time, as functions of *K*, are shown in Figs. 7a–7b.

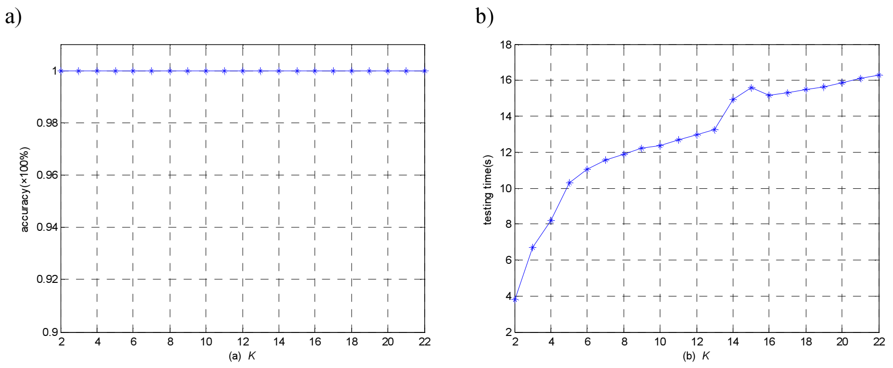


Fig. 7. Accuracy (a) and testing time (b), as functions of *K* for the simulated rectifier.

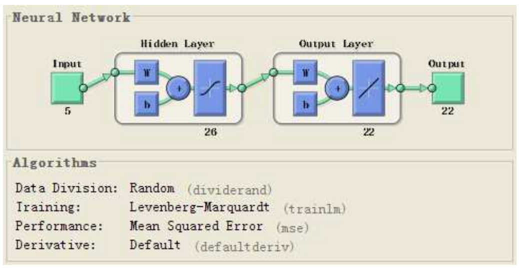
In Fig. 7a, we can observe that the accuracy curve is simple and any value of *K* can lead to 100% classification accuracy. Hence, we choose *K* = 2 as a suitable value, which gives the shortest testing time.

Five classifiers are compared in terms of the *classification Accuracy* (Acc), *training time* (TrT) and *testing time* (TeT) for the testing set. Comparison of SVM classifiers' performance is based on their best results ($\sigma = 1$) and shown in Table 2.

The BPNN used in this study is a forward-feed neural network with three-layered structure, whose training parameters are shown in Fig. 8. Training of this neural classifier was performed with Matlab toolbox for neural networks. Note that the number of hidden neurons is 26, with which a good result can be obtained. Also, the neural classifier was trained for 3 times and the best result was added to Table 2. The Matlab function to validate the testing set is *'sim'*.

Table 2. Comparison of the classifiers' performance for the simulated rectifier.

Classifier	Acc	TrT [s]	TeT [s]
One-against-rest SVM	100%	3.2	19.1
One-against-one SVM	100%	5.2	174.3
DAG SVM	100%	5.2	15.6
*i*DAG SVM (*K* = 2)	100%	5.2	3.8
BPNN	100%	188.6	20.9



```
Algorithms
Data Division:  Random  (dividerand)
Training:       Levenberg-Marquardt  (trainlm)
Performance:    Mean Squared Error  (mse)
Derivative:     Default  (defaultderiv)
```

Fig. 8. The neural network used in our experiment.

4.2. Actual rectifier

An actual rectifier is mainly designed with six discrete power diodes (type: 6A10). A photo of the circuit fault diagnosis system (including a fault setup and a fault data acquisition) is shown in Fig. 9.

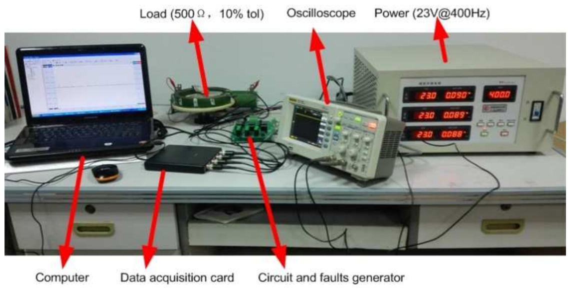


Fig. 9. A photo of the rectifier fault diagnosis platform.

In this system, each diode is connected in series with a relay and an action of the relay, manually controlled with a button, can generate an open fault of diode. A single fault can be generated by pushing one button, whereas double faults need pushing two buttons simultaneously. We used this system to collect 50 samples for each fault. Collecting a fault sample, each time we changed the load randomly within a 10% tolerance. The fault signals were collected with Handyscope HS4 (12-bit ADC inside), and a sample rate for this data collection device was set to 20 kHz, which was consistent with the setup during the simulation.

For each fault, a segment of one sample was randomly selected, as shown in Fig. 10. Subsequently, the WT was applied to these signals for feature extraction.

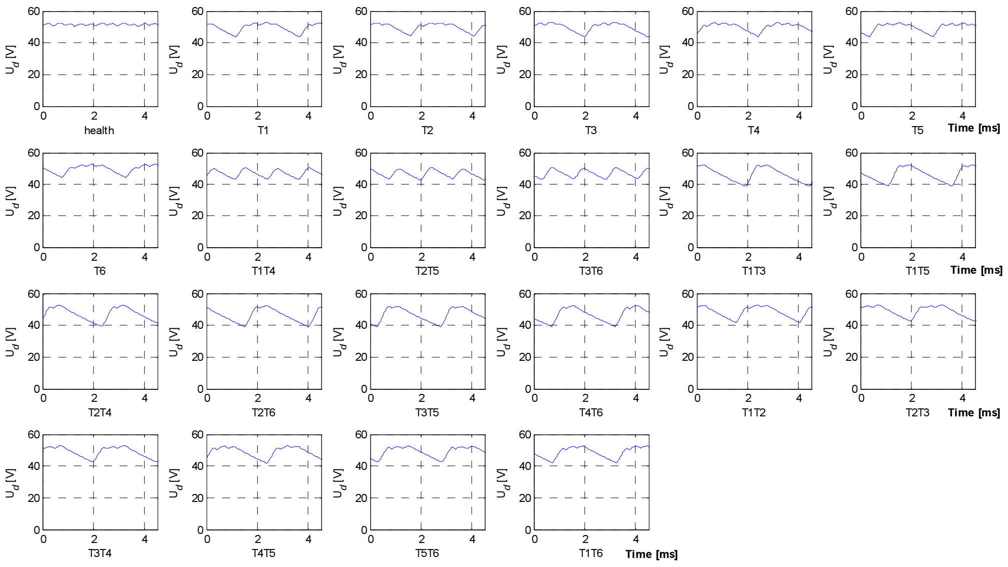


Fig. 10. Waveforms of faults for the actual rectifier.

The starting point (closely related to phase information) of the signal is important for WT, and we found the first starting point of the signal by zero-crossing detection of the input power source waveform U_{ab}. The basic principle of finding a starting point of U_d is shown in Fig. 11, which presents the waveforms of U_d and U_{ab} in a sound circuit condition.

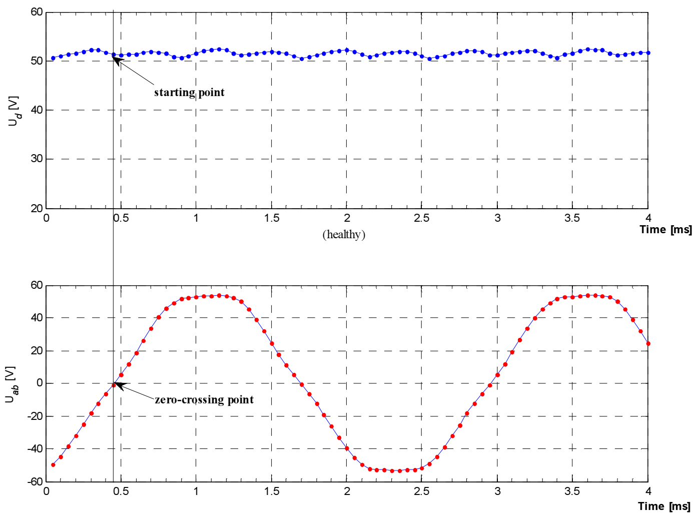


Fig. 11. Finding a starting point of U_d by using the zero-crossing detection method with U_{ab}.

In this research, the steps of feature extraction and classifiers' design were identical to those of the simulated rectifier. Fig. 12 presents the curves of accuracy and testing time for *i*DAG SVM. Comparison of the classifiers' performance with their best results is shown in Table 3 for $\sigma = 4$.

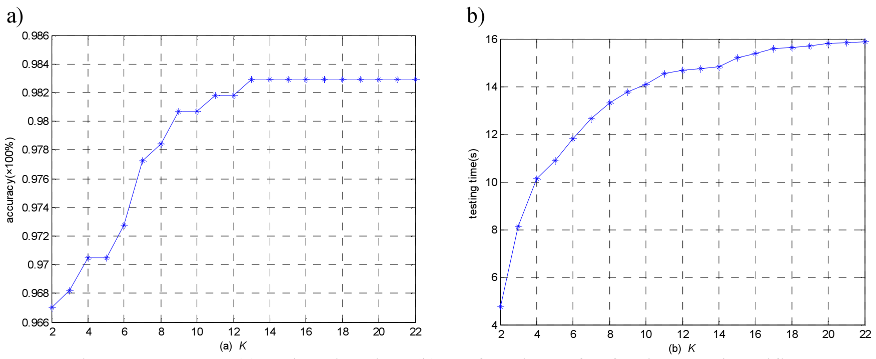


Fig. 12. Accuracy (a) and testing time (b), as functions of *K* for the actual rectifier with fluctuation of load (10%).

Table 3. Comparison of the classifiers' performance for the actual rectifier with fluctuation of load (10%).

Classifier	Acc	TrT [s]	TeT [s]
One-against-rest SVM	97.8%	2.6	18.6
One-against-one SVM	98.3%	5.1	184.3
DAG SVM	98.3%	5.1	17.0
*i*DAG SVM (*K* = 13)	98.3%	5.1	15.2
BPNN	96.8%	35.8	23.9

For the actual rectifier circuit we also performed another experiment, in which both the input power (including amplitude and frequency) and load were randomly fluctuated. The tolerances of input power amplitude, input frequency and load were set to 5%, 5% and 10%, respectively. This experiment aimed to examine the classifier performance in a complicated operation environment. The curves of finding a good result in the *i*DAG SVM classifier design are shown in Fig. 13 and, for $\sigma = 1$, several classifiers can achieve the best results, which are listed in Table 4.

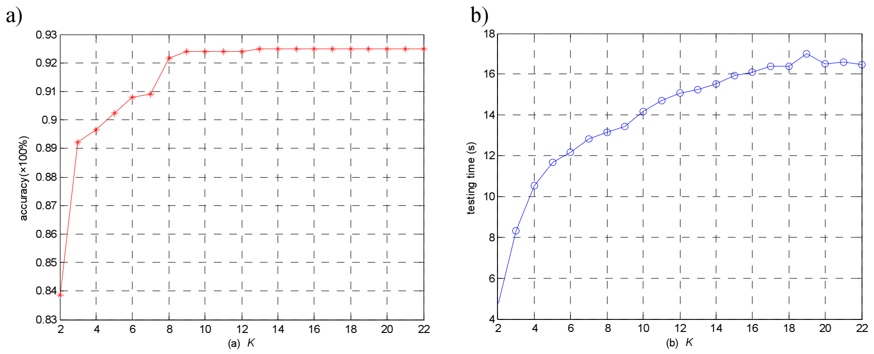


Fig. 13. Accuracy (a) and testing time (b), as functions of *K* for the actual rectifier with fluctuations of load (10%), input power amplitude (5%) and frequency (5%).

Table 4. Comparison of the classifiers' performance for the actual rectifier with fluctuations of load (10%), input power amplitude (5%) and frequency (5%).

Classifier	Acc	TrT [s]	TeT
One-against-rest SVM	91.1%	2.3	18.4
One-against-one SVM	93.2%	4.6	173.2
DAG SVM	92.5%	4.6	15.9
*i*DAG SVM (*K* = 12)	92.5%	4.6	14.9
BPNN	82.5%	13.1	20.7

4.3. Inverter

The third PEC, with its structure shown in Fig. 14, is an actual three-phase inverter which drives a motor with a symmetric structure. The motor is a 50 W *brushless DC motor* (BLDCM) with 3-phase and 5-pair poles, and the BLDCM shaft is coupled with an electric fan, running at a rate of 800 rpm (~5% fluctuation). This system is used for cooling in industry applications. Although the used inverter has a low driving power, the considered fault classification algorithms can be extended to inverters with a higher power in a straightforward way.

In this inverter, the MOSFETs are driven with a square wave *pulse width modulation* (PWM), and the voltage value is Vdc = 48 V. We consider a single switch open fault for this drive circuit, a total of 7 faults need to be classified. In this case, the fault code for switch T*i* is f*i* (*i* = 1, …, 6), and f0 denotes the sound condition of this circuit.

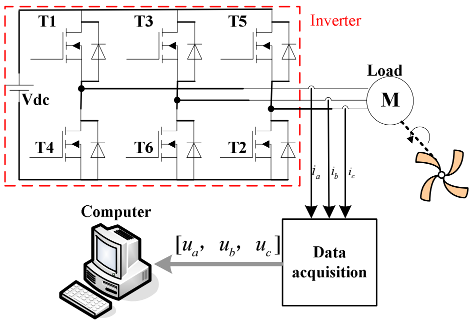


Fig. 14. The inverter system structure used in the experiment.

In this example, three phase currents (i_a, i_b and i_c) need to be collected synchronously. In this experiment, forty samples for each fault pattern were collected based on the experiment platform, in which an open fault of a power switch could be set by an emulator of the drive circuit controller.

Feature extraction. The presented feature extraction algorithm needs two steps.

Step 1: The WT is adopted to reduce the effect of noise. *'Haar'* wavelet is selected as the mother function to decompose the currents' signals into coarse coefficients ($i_a^w(k), i_b^w(k), i_c^w(k)$) and detail coefficients in layer 3. The detail coefficients were discarded in this design.

In order to reduce the effect of load, with reference to [17], normalization of wavelet coefficients can be considered:

$$\begin{aligned} \hat{i}_a(k) &= i_a^w(k)/\max(\mathrm{abs}(i_a^w)) \\ \hat{i}_b(k) &= i_b^w(k)/\max(\mathrm{abs}(i_b^w)) \\ \hat{i}_c(k) &= i_c^w(k)/\max(\mathrm{abs}(i_c^w)) \\ k &= 1,2,\ldots,M/8, \end{aligned} \tag{13}$$

where M is the number of collected data points.

The segments of collected phase currents for each fault before and after *'Haar'* WT in layer 3 are shown in Figs. 15a–15g, respectively.

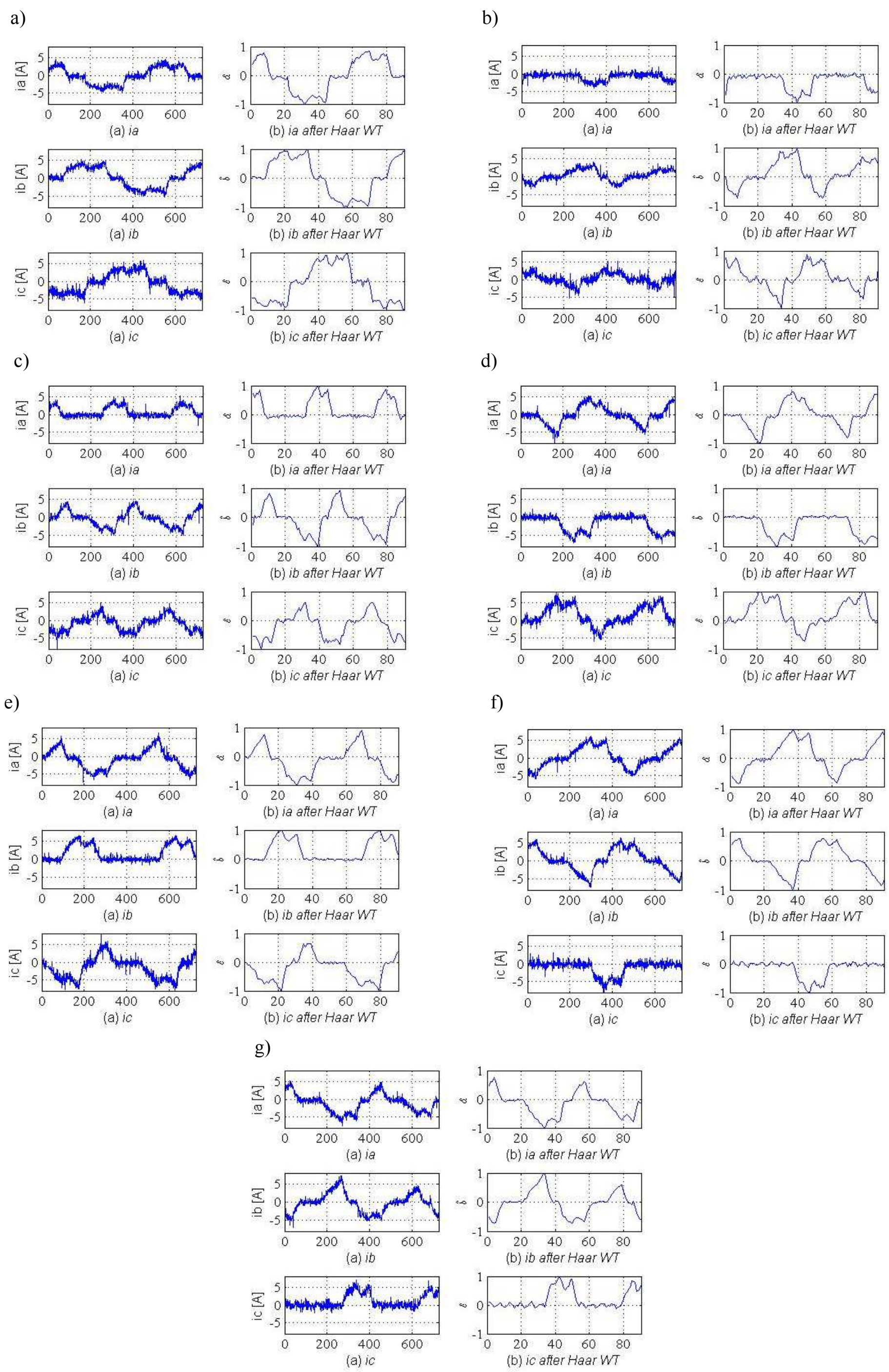


Fig. 15. Current waveforms before and after adapting WT for seven faults, respectively. Waveforms under no fault (a); waveforms under T1 fault (b); waveforms under T2 fault (c); waveforms under T3 fault (d); waveforms under T4 fault (e); waveforms under T5 fault (f); waveforms under T6 fault (g).

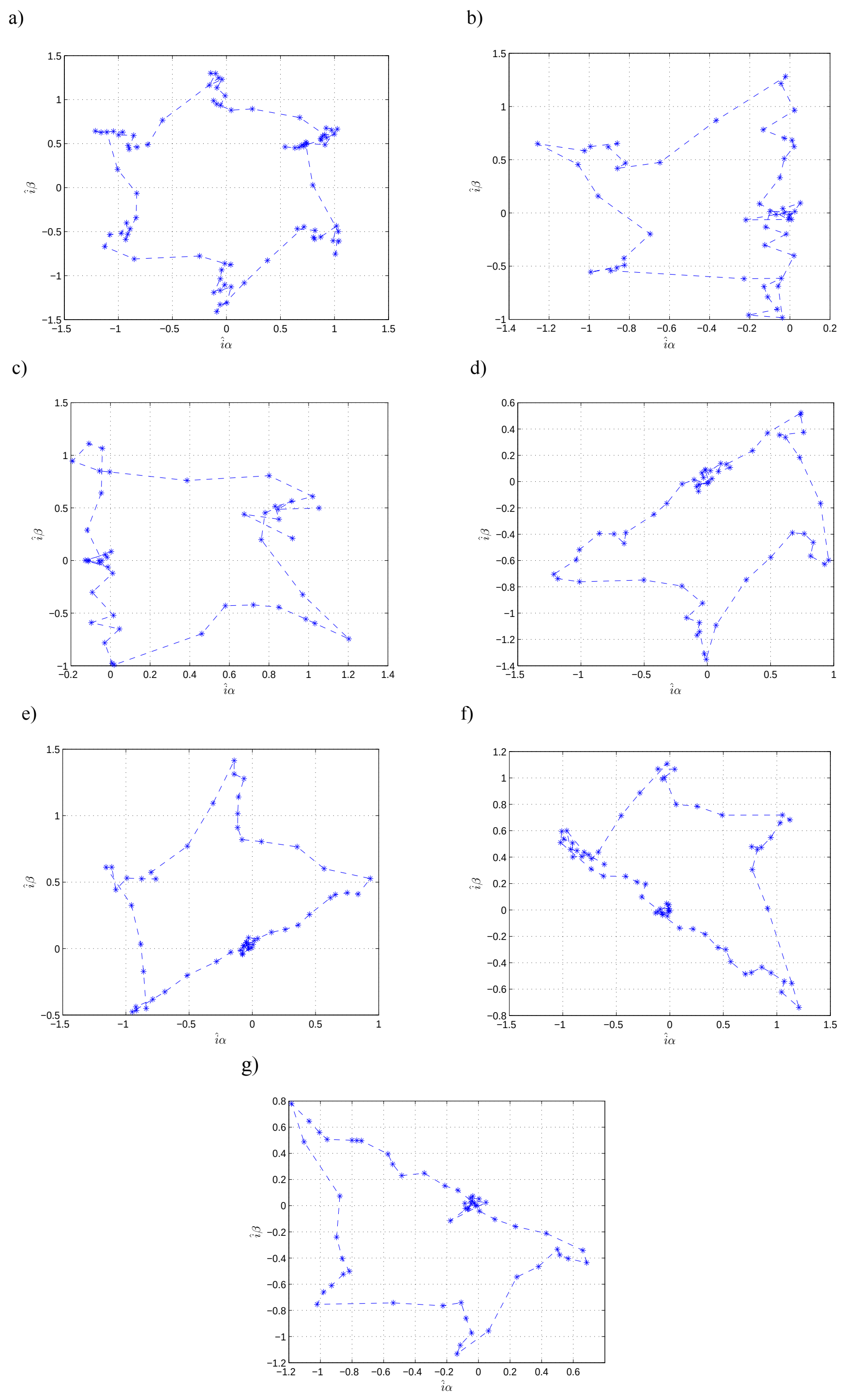


Fig. 16. Current trajectories for seven faults, respectively. A current trajectory under no fault (a); a current trajectory under T1 fault (b); a current trajectory under T2 fault (c); a current trajectoryunder T3 fault (d); a current trajectory under T4 fault (e); a current trajectory under T5 fault (f); a current trajectory under T6 fault (g).

In the Fig. 15, we can observe that, after decomposition of 3 layers, the outlines of current waveforms are clear. Also, we can find that the waveforms for each fault are different from others, and hence they can be used for subsequent fault classification.

Step 2: For the AC motor system with a symmetric structure, the Concordia transform can be used to calculate a 2-dimensional current trajectory $(\hat{i}_\alpha, \hat{i}_\beta)$ in the *α*-*β* frame:

$$\begin{cases} \hat{i}_\alpha = (2\hat{i}_a - \hat{i}_b - \hat{i}_c)/\sqrt{6} \\ \hat{i}_\beta = (\hat{i}_b - \hat{i}_c)/\sqrt{2} \end{cases}. \tag{14}$$

Figure 16 illustrates selected trajectories for $(\hat{i}_\alpha, \hat{i}_\beta)$ under faults. Note that the numbers on the axes have no units because they are ratios from the formulae (13) and (14). We can observe that these trajectories are different from each other mainly in terms of geometry. Hence, we can consider extracting simple centroid features from these trajectories.

The centroid features can be extracted from a closed trajectory [34, 35]:

$$\begin{cases} C_\alpha = \dfrac{8}{M}\sum_{k=1}^{M/8} \hat{i}_\alpha(k) \\ C_\beta = \dfrac{8}{M}\sum_{k=1}^{M/8} \hat{i}_\beta(k) \\ r = \dfrac{8}{M}\sum_{k=1}^{M/8} \sqrt{(\hat{i}_\alpha(k) - C_\alpha)^2 + (\hat{i}_\beta(k) - C_\beta)^2} \\ \hat{C}_\alpha = C_\alpha / r \\ \hat{C}_\beta = C_\beta / r \end{cases} \tag{15}$$

In order to obtain a completely closed trajectory, *M* should be the number of data points from at least one cycle of waveform. A 2-dimensional feature vector $\mathbf{E}=[\hat{C}_\alpha, \hat{C}_\beta]$ can describe the centroid of trajectory regarding its radius size. The 2-dimensional centroid features are sufficient to discern seven faults and can lead to good classification results.

Results for inverter. Fourteen feature samples were used as the training set, whereas the other 26 samples were used as the validation set. By confining *C* to 100, the one-against-rest SVM can achieve the best result for $\sigma=1$, whereas the one-against-one SVM, – for $\sigma=2$.

For the designed *i*DAG SVM, the validation set is also used to search for a proper value of *K*. The searching curves are given in Fig. 17.

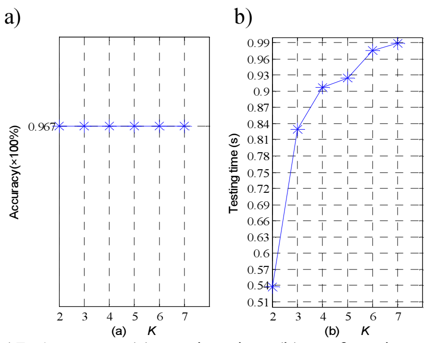


Fig. 17. Accuracy (a); testing time (b), as functions of *K*.

In this experiment, different values of *K* can lead to the same accuracy (96.7%). Hence, *K* = 2 was directly selected for this *i*DAG SVM. In the neural classifier design, the number of hidden neurons is set to 7 and the BPNN is trained for several times. The best result of 97.8% is recorded. Good results for these classifiers are shown in Table 5.

Table 5. Comparison of the classifiers' performance for the inverter.

Classifier	Acc	TrT [s]	TeT [s]
One-against-rest SVM	96.1%	1.0	1.12
One-against-one SVM	96.7%	0.6	3.66
DAG SVM	96.7%	0.6	0.99
*i*DAG SVM (K = 2)	96.7%	0.6	0.54
BPNN	97.8%	0.3	4.65

4.4. Analysis

1) According to the above examples, we believe that the SVM and neural classifiers are applicable to power electronic circuit faults' classification, if only the parameter values are properly chosen.
2) Through several experiments, we found that the neural networks can achieve good results when classifying a small number of faults. However, it is not a good choice to classify a large number of faults with conventional neural networks, because these classifiers will need much more hidden neurons to implement a complex training process and so to increase the calculation complexity. Changing a big network into some small-scale networks can solve this problem [32], but both the structure and parameters of these small networks need to be determined deliberately. Moreover, the neural network classifier exhibits different performance for different trainings.
   The SVM classifiers can be regarded as alternative solutions for the neural classifiers in the applications to power electronic system diagnosis, because an SVM classifier can achieve very close or even better performance to that of a neural classifier. In addition, a standard SVM classifier can exhibit stable performance for different trainings. Moreover, it needs to tune relatively fewer parameters.
3) The SVM classifier has many forms [36, 37]. In the research, we examined two typical forms in the domain of power electronic circuit fault diagnosis. As a result, the one-against-one SVM is usually used for classification, because it can achieve a high classification accuracy. However, this classifier needs a high computational complexity to implement classification and this drawback limits its usage in some applications, *e.g.* in power electronic circuit on-line fault diagnosis, surveillance or even fault-tolerant systems. Hence, the structure of this classifier needs to be improved or rearranged to adapt to fast fault classification. We adopt the DAG SVM as an alternative for this classifier.
   In our research, the DAG SVM and the one-against-one SVM were compared in terms of classification accuracy and testing time, and we found that the DAG SVM's performance was very close to its counterpart's, but with a significantly lower computational complexity needed. Hence, we believe that the DAG SVM can be used to replace the one-against-one SVM for fault classification of PECs. The *i*DAG SVM, with the help of nearest neighbours, can further reduce the testing time, whilst maintaining almost unchanged performance.
4) The WT is a good tool for fault feature extraction of power electronic circuits. We achieved good results by using this tool in the experiments. The proper selection of a good mother function and decomposition layers is a difficult problem, and in our research we solved this problem by comparing and evaluating the experiment results with different parameters.

5. Conclusions

In industrial applications, a fast method of fault diagnosis in power electronic circuits is important because of the requirements of high reliability and fault-tolerance. This paper presents a data-driven method of fault classification in power electronic circuits, and this method is based on the DAG SVM. This classifier can be improved by combining it with the

K-NN method. Compared with the conventional one-against-rest SVM and one-against-one SVM, the presented method has a very high implementation speed, because this method is based on the DAG SVM, which needs to compute (N−1) BSVMs for N faults. After the improvement, the number of BSVMs needed by *i*DAG SVM is less than or equal to (N−2). Hence, among the SVM classifiers, the *i*DAG SVM has the lowest computational complexity. Also, the *i*DAG SVM has the classification performance comparable with that of the DAG SVM, if only the parameter K is properly selected. In our research, K was determined on the basis of experimental searching results. In another way, K can be selected with an empirical formula as follows:

$$K = \lceil N/2 \rceil . \tag{16}$$

With this formula, in many experiments, we found that the *i*DAG SVM can achieve satisfactory results. Note that the proposed classifier can also be considered as a general method, and this classifier can be easily extended to other fast fault classification applications. The presented method also has some limitations, because it is based on the conventional DAG SVM, which needs to be prearranged. In arranging the DAG SVM structure, the selection of a root node for the DAG SVM is a problem. Different root nodes will probably lead to different performance of the classifier. Hence, the proper selection of a root node should be further studied. In the future, we can consider some available methods in designing pattern classifiers to solve this problem [38–40].

Finally, the feature extraction is important in the design of a successive classifier In our research we need to select fault features manually, so in the future work, automatic and efficient feature selection methods will be examined.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant # 51377079).

References

[1] Mohagheghi, S., Harley, R.G., Habetler, T.G., Divan, D. (2009). Condition monitoring of power electronic circuits using artificial neural networks. *IEEE Trans. Power Electron.*, 24(10), 2363–2367.

[2] Khomfoi, S., Tolbert, L.M. (2007). Fault diagnostic system for a multilevel inverter using a neural network. *IEEE Trans. Power Electron.*, 22(03), 1062–1069.

[3] Mirafzal, B. (2014). Survey of fault-tolerance techniques for three-phase voltage source inverters. *IEEE Trans. Ind. Electron.*, 61(10), 5192–5202.

[4] Filippetti, F., Franceschini, G., Tassoni, C., Vas, P. (2000). Recent developments of induction motor drives fault diagnosis using AI techniques. *IEEE Trans. Ind. Electron.*, 47(05), 994–1004.

[5] Zidani, F., Diallo, D., Benbouzid, M.E.H., Naït-Saïd, R. (2008). A fuzzy-based approach for the diagnosis of fault modes in a voltage-fed PWM inverter induction motor drive. *IEEE Trans. Ind. Electron.*, 55(02), 586–593.

[6] Khanniche, M.S., Mamat-Ibrahim, M.R. (2004). Wavelet-fuzzy-based algorithm for condition monitoring of voltage source inverter. *Electron. Lett.*, 40(04), 267–268.

[7] Potamianos, P.G., Mitronikas, E.D., Safacas, A.N. (2014). Open-circuit fault diagnosis for matrix converter drives and remedial operation using carrier-based modulation methods. *IEEE Trans. Ind. Electron.*, 61(1), 531–545.

[8] An, Q.T., Sun, L.Z., Sun, L., Jahns, T.M. (2010). Low-cost diagnostic method for open-switch faults in inverters. *Electron. Lett.*, 46(14), 1021–1022.

[9] Diallo, D., Benbouzid, M.E.H., Hamad, D., Pierre, X. (2005). Fault detection and diagnosis in an induction machine drive: a pattern recognition approach based on Concordia stator mean current vector. *IEEE Trans. Energy Conver.*, 20(03), 512–519.

[10] Charfi, F., Sellami, F., Al-Haddad, K. (2006). Fault diagnostic in power system using wavelet transforms and neural networks. *Proc. ISIE*, 1143–1148.

[11] Kadri, F., Drid, S., Djeffal, F.Y., Chrifi-Alaoui, L. (2013). Neural classification method in fault detection and diagnosis for voltage source inverter in variable speed drive with induction motor. *Proc. EVER*, 1–5.

[12] Masrur, M.A., Chen, Z., Murphey, Y. (2010). Intelligent diagnosis of open and short circuit faults in electric drive inverters for real-time applications. *IET Power Electron.*, 3(02), 279–291.

[13] Ma, C., Gu, X., Wang, Y. (2009). Fault diagnosis of power electronic system based on fault gradation and neural network group. *Neurocomputing*, 72(13–15), 2909–2914.

[14] Lu, B., Sharma, S.K. (2009). A literature review of IGBT fault diagnostic and protection methods for power inverters. *IEEE Trans. Ind. Appl.*, 45(5), 1770–777.

[15] Khomfoi, S., Tolbert, L.M. (2007). Fault diagnosis and reconfiguration for multilevel inverter drive using AI-based techniques. *IEEE Trans. Ind. Electron.*, 54(6), 2954–2968.

[16] Fan, B., Dong, M., Zhao, J., Zhang, Q. (2010). Three-phase inverter fault diagnosis based on optimized neural networks. *Proc. ICCASM*, 4, 482–485.

[17] Kim, D.E., Lee, D.C. (2008). Fault diagnosis of three-phase PWM inverters using wavelet and SVM. *Proc ISIE*, 329–334.

[18] Delpha, C., Chen, H., Diallo, D. (2012). SVM based diagnosis of inverter fed induction machine drive: a new challenge. *Proc. IECON*, 3931–3936.

[19] Hsu, C.W., Lin, C.J. (2002). A comparison of methods for multi-class support vector machines. *IEEE Trans. Neural Networ.*, 13(2), 415–425.

[20] Wang, R., Zhan, Y., Zhou, H., Cui, B. (2013). A fault diagnosis method for three-phase rectifiers. *Int. J. Elec. Power*, 52, 266–269.

[21] Wang, R., Zhan, Y., Zhou, H. (2012). Application of S transform in fault diagnosis of power electronics circuits. Scientia Iranica, 19(3), 721–726.

[22] Xu, H., Zhang, J., Qi, J., Wang, T., Han, J. (2014). RPCA-SVM fault diagnosis strategy of cascaded H-bridge multilevel inverters. *Proc. ICGE*, 164–169.

[23] Hu, Z., Gui, W., Yang, C., Deng, P., Ding, S. X. (2011). Fault classification method for inverter based on hybrid support vector machines and wavelet analysis. *Int. J. Control Autom. Syst.*, 9(4), 797–804.

[24] Huang, C., Zhao, J., Wu, C. (2013). Data-based inverter IGBT open-circuit fault diagnosis in vector control induction motor drives. *Proc. IEEE ICIEA*, 1039–1044.

[25] Cortes, C., Vapnik, V.N. (1995). Support-vector networks. *J. Mach. Learn.*, 20(3), 273–297.

[26] Vapnik, V. (1999). An overview of statistical learning theory. *IEEE Trans. Neural Networ.*, 10(5), 988–999.

[27] Burges, C.J.C. (1998). A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Disc.*, 2(2), 121–167.

[28] Chapelle, O., Haffner, P., Vapnik, V.N. (1999). Support Vector Machines for histogram-based image classification. *IEEE Trans. Neural Networ.*, 10(5), 1055–1064.

[29] Platt, J.C., Cristianini, N., Shawe Taylor, J. (2000). Large margin DAGs for multi-class Classification. *Advances in Neural Information Processing Systems*, 12, 547–553.

[30] Biet, M. (2013). Rotor faults diagnosis using feature selection and nearest neighbors rule: application to a turbogenerator. *IEEE Trans. Ind. Electron.*, 60(9), 4063–4073.

[31] Martins, J.F., Pires, V.F., Lima, C., Pires A.J. (2012). Fault detection and diagnosis of grid-connected power inverters using PCA and current mean value. *Proc. IECON*, 5185–5190.

[32] Murphey, Y.L., Masrur, M.A., Chen, Z., Zha, B. (2006). Model-based fault diagnosis in electric drives using machine learning. *IEEE-ASME Trans. Mech.*, 11(3), 290–303.

[33] Mallat, S.G. (1989). A theory for multi-resolution signal decomposition: the wavelet representation. *IEEE Trans. Pattern Anal*., 11(7), 674–693.

[34] Fernao, V.P., Amaral, T.G., Martins, J.F. (2012). Fault detection and diagnosis of voltage source inverter using the 3D current trajectory mass center. *Proc. IEEE ICIT*, 737–742.

[35] Cui, J. (2015). Faults classification of power electronic circuits based on a support vector data description method. *Metrol. Meas. Syst*., 22(2), 205–222.

[36] Cui, J., Wang, Y. (2011). A novel approach of analog circuit fault diagnosis using support vector machines classifier. *Measurement*, 44(1), 281–289.

[37] Cui, J., Wang, Y. (2011). Analog circuit fault classification using improved one-against-one support vector machines. *Metrol. Meas. Syst*., 18(4), 569–582.

[38] Gu, B., Sheng, Victor S., Li, S. (2015). Bi-parameter space partition for cost-sensitive SVM. *Proc. IJCAI*, 3532–3539.

[39] Wen, X., Shao, L., Xue, Y., Fang, W. (2015). A rapid learning algorithm for vehicle classification. *Inform. Sciences*, 295(1), 395–406.

[40] Gu, B., Sheng, Victor S. (2016). A Robust Regularization Path Algorithm for ν-Support Vector Classification. *IEEE Trans. Neural Netw. Learn. Syst*., DOI: 10.1109/TNNLS.2016.2527796.