Chinese Journal of Aeronautics, (2019), 32(2): 296–307

Chinese Society of Aeronautics and Astronautics
& Beihang University

Chinese Journal of Aeronautics

cja@buaa.edu.cn
www.sciencedirect.com

Aircraft engine fault detection based on grouped convolutional denoising autoencoders

Xuyun FU [a], Hui LUO [b], Shisheng ZHONG [a,b,*], Lin LIN [b]

[a] Department of Mechanical Engineering, Harbin Institute of Technology at Weihai, Weihai 264209, China
[b] School of Mechatronics Engineering, Harbin Institute of Technology, Harbin 150001, China

Received 10 December 2017; revised 7 February 2018; accepted 4 July 2018
Available online 5 January 2019

KEYWORDS

Aircraft engines;
Anomaly detection;
Convolutional Neural Network (CNN);
Denoising autoencoder;
Engine health management;
Fault detection

Abstract Many existing aircraft engine fault detection methods are highly dependent on performance deviation data that are provided by the original equipment manufacturer. To improve the independent engine fault detection ability, Aircraft Communications Addressing and Reporting System (ACARS) data can be used. However, owing to the characteristics of high dimension, complex correlations between parameters, and large noise content, it is difficult for existing methods to detect faults effectively by using ACARS data. To solve this problem, a novel engine fault detection method based on original ACARS data is proposed. First, inspired by computer vision methods, all variables were divided into separated groups according to their correlations. Then, an improved convolutional denoising autoencoder was used to extract the features of each group. Finally, all of the extracted features were fused to form feature vectors. Thereby, fault samples could be identified based on these feature vectors. Experiments were conducted to validate the effectiveness and efficiency of our method and other competing methods by considering real ACARS data as the data source. The results reveal the good performance of our method with regard to comprehensive fault detection and robustness. Additionally, the computational and time costs of our method are shown to be relatively low.

1. Introduction

With the development of civil aviation, the safety, reliability, and efficiency of the aircraft engine have received considerable attention. Engine fault detection is an important process of improving the abovementioned metrics, and can help managers allocate monitoring resources more reasonably, improve the safety and reliability of every flight, establish a scientifically based maintenance scheme, and achieve the maximum reduction of operation and maintenance costs.[1]

* Corresponding author at: Department of Mechanical Engineering, Harbin Institute of Technology at Weihai, Weihai 264209, China.
E-mail address: zhongss@hit.edu.cn (S. ZHONG).

Peer review under responsibility of Editorial Committee of CJA.

https://doi.org/10.1016/j.cja.2018.12.011

Generally, engine fault detection methods can be divided into model-based methods and data-driven methods. Model-based methods can establish a mathematical engine model on the basis of the engine's operation mechanisms. These methods have the advantage of a good fault detection model, which can detect abrupt faults and gradual performance degradation. Moreover, the fault detection results are interpretable. However, these methods have obvious shortcomings. First, the complex nonlinearity and uncertainties of the engine pose tremendous modelling difficulties.[2] Second, the establishment of an engine model requires a large amount of expertise and unavailable design parameters.

With the development of Machine Learning (ML), big data, and computational power, great attention has been paid to data-driven methods, which can overcome the various shortcomings of model-based methods. First, ML methods, and deep learning methods in particular, are good at modeling complex nonlinear dynamic systems. Additionally, such methods do not rely much on expertise and unavailable design parameters. However, data-driven methods also have their limitations because they depend heavily on large amounts of historical data. Typically, the established model is a black box, which makes the explanation of the fault mechanisms difficult. Based on the above analysis and the current research situation, a data-driven method is investigated in this study.

Many existing data-driven engine fault detection methods depend on performance deviation data.[3–6] However, such data are provided by the Original Equipment Manufacturer (OEM), and airlines have to pay high fees to acquire them. If the collaboration between the airlines and the OEM ends owing to various unpredictable reasons, it will become difficult for the airlines to acquire the performance deviation data and detect the engine faults effectively. Thereby, this may lead to unsafe flight incidents and enormous financial losses.

To improve the ability of independent engine fault detection, a type of condition monitoring data that can replace the performance deviation data is required. In this regard, the Aircraft Communications Addressing and Reporting System (ACARS) data are a good choice. However, ACARS data are different to the performance deviation data in several aspects. Fig. 1 shows a comparison between smoothed performance deviation parameters and original ACARS parameters during the cruise phase. Firstly, as shown in Fig. 1, the ACARS data often have a higher dimension than the performance deviation data. Secondly, the performance deviation data can eliminate the influences of working conditions and ambient environment, and thus the status of an engine can be indicated more clearly in such data. In contrast, ACARS data are largely affected by the working conditions and ambient environment. Therefore, the fault patterns often hide in the complex variation of multiple parameters. Finally, ACARS data typically contain a lot of noise, which can influence the analysis results.

ACARS data typically contain more information about the engine's status in comparison with performance deviation data. If the ACARS data can be processed by appropriate methods, the obtained fault detection results may be better than the results obtained by methods based on performance deviation data. However, the characteristics of high dimension, complex correlations between parameters, and excessive noise content, also present challenges with regard to fault detection in the ACARS data. In previous studies,[3–5] fault detection methods based on performance deviation data were proposed. Although these methods achieved good fault detection results with the performance deviation data, they cannot be applied to fault detection in the more complicated ACARS data. Various studies have validated the methods with data generated by engine simulation models.[7–9] However, real ACARS data depend on the working conditions and ambient environment, and thus real ACARS data always have lower quality and higher complexity in comparison with simulated data. In some studies,[10,11] the number of used parameters was small. Therefore, their methods cannot be applied to the high-dimensional ACARS data. In summary, most existing engine fault detection methods cannot process ACARS data effectively.

The Convolutional AutoEncoder (CAE), which has received extensive attention in recent years, has achieved good results in many high-dimensional and complex pattern recognition problems. Some studies have used the CAE to extract

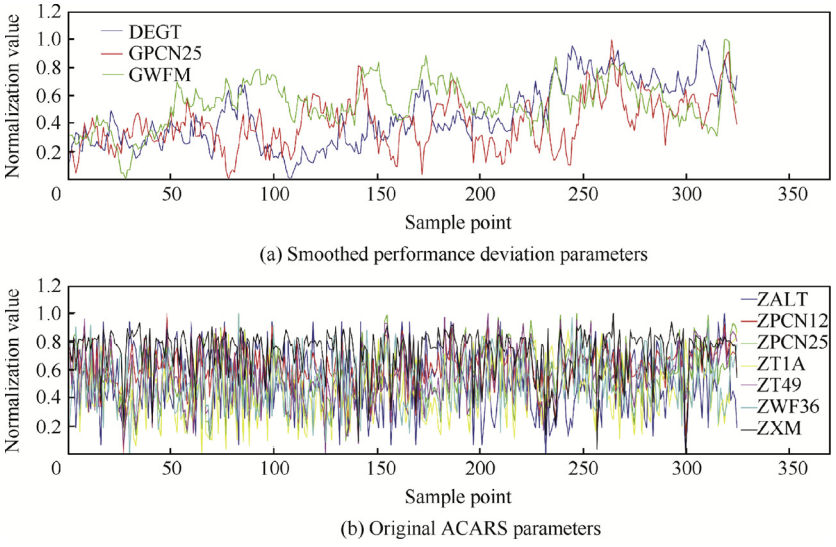

Fig. 1 Comparison between smoothed performance deviation parameters and original ACARS parameters.

the hierarchical representations of images and applied it to image classification problems.[12,13] One study took the spatial and temporal attributes of video sequences into consideration and proposed a spatial–temporal CAE to classify the video sequences.[14] Another study used the CAE to learn motion manifolds and applied it to animation problems.[15] In another study,[16] page segmentation was defined as a pixel labeling problem, and the CAE was used to learn the features from the pixel intensities. Finally, a study has used the CAE to detect and classify the electricity transmission line faults.[17] However, the considered dataset only had a dimension of six.

Inspired by the abovementioned studies, a novel and effective engine fault detection method based on the ACARS data is proposed. This method can achieve good fault detection results without requiring a large amount of labeled data, while keeping the computational and time costs relatively low. The proposed method does not require much expertise and data preprocessing work. First, instead of extracting features from all of the variables in the ACARS data directly, the variables are divided into several groups according to their correlations. Then, unsupervised feature learning is carried out independently for each variable group, and the feature weights are reshaped as convolution kernels. Subsequently, a convolutional feature mapping and pooling operation is performed on each variable group to extract features. Finally, all of the variable groups' features are fused to form the feature vectors. The Support Vector Machine (SVM) method is used to identify faults based on the feature vectors. Experiments with real ACARS data of a certain type of civil aircraft engine were conducted to validate the effectiveness of our method. The experimental results reveal that our method outperformed competing fault detection methods. Additionally, our method requires less parameters and has faster computational speed. In summary, the experiments reveal the superiority of our method. With these advantages, our method can be applied to real-world scenarios.

This paper is organized as follows. In Section 2, the technical framework and main technical solutions of our method are described. In Section 3, the experiments conducted with real ACARS data are discussed and the experimental results are presented for the purpose of comparing the effectiveness of our method with that of competing methods. In Section 4, several conclusions are drawn and future research directions are discussed.

2. Methodology

CAE has been used to simultaneously process all variables in a multivariate time series.[17] However, if CAE is used to process ACARS data, it may face several problems such as local features being covered up, decreased feature representation ability, high model complexity, and enormous computational and time costs.

To resolve these issues, a novel engine fault detection method based on original ACARS data is proposed. The method introduces a variable grouping operation, namely, grouping multiple variables according to their correlations. After the variable grouping, the input dimension is reduced and the correlations between the input variables are simplified. The weak correlations between the parameters are ignored, while the strong correlations are strengthened. More robust and representative features can be learned from each variable group through an unsupervised approach. Additionally, after the variable grouping, the number of parameters and calculation complexity reduce to a great extent.

The technical framework of our method is shown in Fig. 2. In Fig. 2, ZALT is the altitude, ZXM is the Mach number, ZPOIL is the oil pressure, ZTOIL is the oil temperature, t and T is the time index and the length of the parameter series, respectively. W_{ij} and W'_{ij} is the weight in the ith row and the jth column of the convolutional kernel. The width of convolutional kernel is d. C_i and C'_i is the ith element in the vector after convolution, P_i and P'_i is the ith element in the vector after pooling. The length of the vector after convolution is f, and the length of the vector after pooling is g. The main technical solutions are as follows: (A) grouping the ACARS variables according to their correlations; (B) acquiring the convolutional kernels of every variable group through unsupervised feature learning; (C) implementing the convolutional feature mapping and pooling operation for every variable group individually; (D) fusing the extracted features to form feature vectors and using SVM to identify faults based on the feature vectors.

2.1. Grouping of ACARS variables

In the computer vision domain, because the pixels in a patch are closely related, the unsupervised feature learning of convolutional kernels is usually implemented in patches.[12,13] Inspired by this, the ACARS variables were grouped according to their correlations. Closely related variables were placed in the same group, while weakly related variables were placed in separate groups. K-means clustering, independent variable group analysis, and Agglomerative Independent Variable Group Analysis (AIVGA) are common variable grouping methods. In our study, the AIVGA method was chosen as the method to group the ACARS variables because it can determine the optimal number of groups automatically and run relatively fast.

It is assumed that the ACARS dataset is represented as $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m]$, and $\boldsymbol{x}_i = [x_i(1), x_i(2), \cdots, x_i(T)]^{\mathrm{T}}$ is the time series for one of the variables. The length of the time series is T. The objective of the AIVGA algorithm is to find a partition of m variables to n disjoint subsets $G = \{G_i | i = 1, 2, \ldots, n\}$, and that partition should maximize the sum of the marginal log-likelihoods of the models H_i for the different groups.[18] $\boldsymbol{\theta}_i$ are the parameters of the models H_i. To approximate the marginal log-likelihoods, the variation Bayes approximations $q_i(\boldsymbol{\theta}_i)$ were fitted to the posteriors $p(\boldsymbol{\theta}_i|\boldsymbol{X}_{G_i}, H_i)$ of different groups' models. During the model fitting process, the cost function C was minimized as follows:

$$\begin{aligned} C(G) &= \sum_i C(\boldsymbol{X}_{G_i}|H_i) = \sum_i \int \ln \frac{q_i(\boldsymbol{\theta}_i)}{p(\boldsymbol{X}_{G_i}, \boldsymbol{\theta}_i|H_i)} q_i(\boldsymbol{\theta}_i) \mathrm{d}\boldsymbol{\theta}_i \\ &= \sum_i [D_{\mathrm{KL}}(q_i(\boldsymbol{\theta}_i)||p(\boldsymbol{\theta}_i|\boldsymbol{X}_{G_i}, H_i)) - \ln p(\boldsymbol{X}_{G_i}|H_i)] \\ &\geqslant -\sum_i \ln p(\boldsymbol{X}_{G_i}|\boldsymbol{H}_i) \end{aligned} \tag{1}$$

where $D_{\mathrm{KL}}(q||p)$ is the Kullback-Leibler divergence between q and p.

The relationship between the cost function and the mutual information can be expressed as

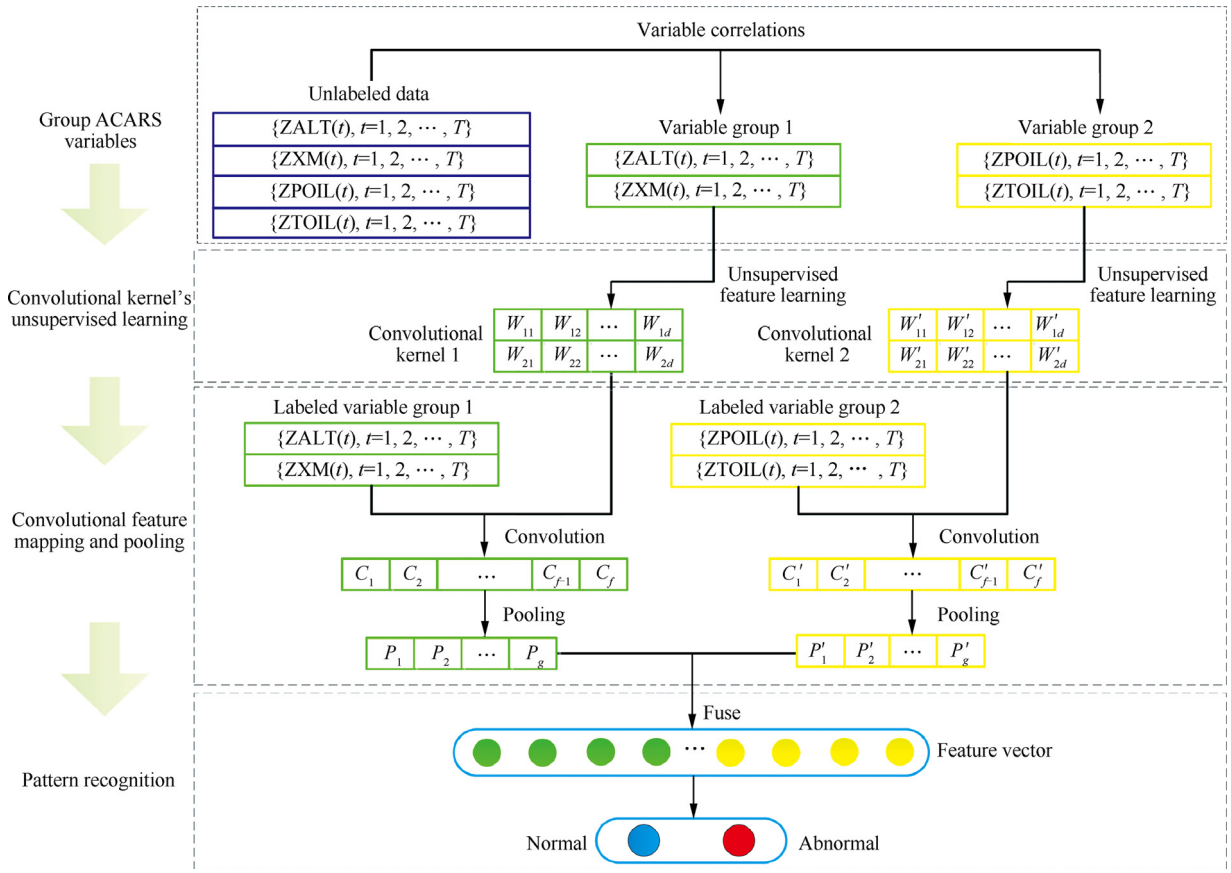

Fig. 2 Schematic diagram of proposed technical framework.

$$C(G) \geqslant -\sum_i \ln p(\boldsymbol{X}_{G_i}|H_i) \approx TI_G(\boldsymbol{x}) + TH(\boldsymbol{x})$$
$$I_G(\boldsymbol{x}) = \sum_i H(\{\boldsymbol{x}_j | j \in G_i\}) - H(\boldsymbol{x}) \tag{2}$$

where $H(\boldsymbol{x})$ represents the entropy of the random vector $\boldsymbol{x}$, and $I_G(\boldsymbol{x})$ represents the mutual information related to the G grouping.

2.2. Unsupervised learning of convolutional kernels

2.2.1. Advantages of training denoising autoencoders in groups

The sparse coding, Denoising AutoEncoder (DAE), and restricted Boltzmann machine, are three commonly used unsupervised feature learning models. The DAE model is powerful in extracting robust and representative features, and is also easy to construct. The dropout technique can be used to regularize the DAE model. Therefore, in this study, the DAE model with dropout was used to extract the features of the ACARS data.

The extracted features were a set of functions of all the variables considered by previous methods. Thus, the extracted features of the ACARS data can be expressed as

$$\text{Features} = \{f(\text{ZALT}, \text{ZXM}, \text{ZPOIL}, \text{ZTOIL}, \dots)\} \tag{3}$$

where f represents the functions of the extracted features.

In our method, every DAE is individually trained on a certain variable group. Therefore, the number of DAEs is equal to the number of variable groups. The extracted features are a set of functions of the variables in the same group. Assuming that all variables in the ACARS data can be divided into the variable groups G_1 and G_2, the extracted features of the ACARS data can be expressed as

$$\begin{cases} G_1 = \{\ \text{ZALT}, \text{ZXM}, \dots\} \\ G_2 = \{\ \text{ZPOIL}, \text{ZTOIL}, \dots\} \\ \text{Features} = \{f_1(G_1), f_2(G_2)\} \end{cases} \tag{4}$$

where f_1 and f_2 are the functions of the extracted features in variable groups G_1 and G_2, respectively.

The convolutional kernels are trained on each variable group instead of being trained on all variables. This operation has at least two advantages. Most importantly, the extracted features can better represent the distribution space of the ACARS data because the transformation of large-scale problems into sub-problems is typically effective in increasing the compactness of the models. In some studies,[18,19] all variables were divided into several subsets according to their correlations, and the model of each subset was developed independently. The experiments described below reveal that a more compact and efficient model can be developed in this way.

Furthermore, the number of network parameters and time complexity can be reduced to a great extent. Assuming that the length and dimension of the ACARS data segment are l and m, respectively, the input dimension of the DAE model is ml. To prevent the extracted features from losing an excessive amount of original information, the number of hidden layer nodes is typically slightly less than the input dimension. Thus, the num-

ber of the hidden layer nodes was $ml - s_1$ ($s_1 \ll ml$). If all of the variables in the ACARS data were divided equally into n groups, the dimension of the input in every DAE model would be ml/n, and the number of hidden layer nodes would be $ml/n - s_2$ ($n \ll ml$, $s_2 \ll ml/n$). A comparison between Chen's method[16] and our method is presented in Table 1.

The ratio of the total number of parameters in the two methods is as follows:

$$\frac{n(ml/n+1)(ml/n-s_2)}{(ml+1)(ml-s_1)} = \frac{ml+n}{ml+1}\cdot\frac{ml/n-s_2}{ml-s_1} \approx 1\cdot\frac{ml/n}{ml} = \frac{1}{n} < 1 \tag{5}$$

The total number of parameters and the time complexity of our method were approximately equal to $1/n$ of Chen's method. Therefore, our method can not only increase the computational speed, but also assist in the implementation of parallel network training. It has been proven that parallel training can break the performance limits of single equipment, train a larger-scale network, and achieve higher accuracy in a classification task.[20]

2.2.2. Learning convolutional kernels through DAE

The DAE model can only take one-dimensional vectors as input. Therefore, every ACARS segment is first transformed into a one-dimensional vector $\boldsymbol{A}$ = [ZALT(t), ZXM(t), ZPOIL(t), ZTOIL(t), ...] (t = 1, 2, ⋯, T). The schematic diagram of the convolutional kernel's unsupervised feature learning is shown in Fig. 3.

In Fig. 3, $\boldsymbol{A}$ represents the original input data, while $\tilde{\boldsymbol{A}}$ represents the corrupted input data. The masking corruption method, which is used in our study, forces a λ fraction of the randomly chosen samples to be zero. Through a stochastic function $s_f(\cdot)$, the corruption of $\boldsymbol{A}$ to $\tilde{\boldsymbol{A}}$ is expressed as

$$\tilde{\boldsymbol{A}} = s_f(\boldsymbol{A}, \lambda) \tag{6}$$

Then, $\tilde{\boldsymbol{A}}$ is mapped to the hidden representation $\boldsymbol{h}$, which is expressed as

$$\boldsymbol{h} = \sigma(\boldsymbol{W}\tilde{\boldsymbol{A}} + \boldsymbol{b}) \tag{7}$$

Table 1 Comparison between Chen's method and our method.

Index	Chen's method	Our method
Number of DAEs	1	n
Number of input/output layer nodes	ml	ml/n
Number of hidden layer nodes	$ml - s_1$	$ml/n - s_2$
Number of parameters	$(ml + 1)(ml - s_1)$	$(ml/n + 1)(ml/n - s_2)$
Total number of parameters	$(ml + 1)(ml - s_1)$	$n(ml/n + 1)(ml/n - s_2)$
Time complexity	$O((ml + 1)(ml - s_1))$	$O(n(ml/n + 1)(ml/n - s_2))$

Note: Rows two to four refer to each DAE.

where σ is the nonlinear sigmoid activation function, $\boldsymbol{W}$ is the weight matrix, and $\boldsymbol{b}$ is the bias vector between the input layer and the hidden layer.

Dropout is an effective model regularization technique. During the training process, dropout generated a 0–1 random number vector $\boldsymbol{b}_v$, which followed a Bernoulli distribution with a probability d. Moreover, $\tilde{\boldsymbol{h}}$ is a scalar product of $\boldsymbol{b}_v$ and $\boldsymbol{h}$, and is defined as

$$\begin{cases} \boldsymbol{b}_v \sim \text{Bernoulli}(1-d) \\ \tilde{\boldsymbol{h}} = \boldsymbol{b}_v \cdot \boldsymbol{h} \end{cases} \tag{8}$$

$\tilde{\boldsymbol{h}}$ is mapped to $\boldsymbol{Z}$ and is expressed as

$$\boldsymbol{Z} = \sigma(\boldsymbol{W}'\tilde{\boldsymbol{h}} + \boldsymbol{b}') \tag{9}$$

where $\boldsymbol{Z}$ is a reconstruction version of $\boldsymbol{A}$, $\boldsymbol{W}'$ is the weight matrix, and $\boldsymbol{b}'$ is the bias vector between the hidden layer and the output layer.

The training objective of the DAE is to bring $\boldsymbol{Z}$ as close to $\boldsymbol{A}$ as possible. The objective function J of the DAE is defined as

$$J = \frac{1}{2v}\sum_{i=1}^{v}\| \boldsymbol{Z}^{(i)} - \boldsymbol{A}^{(i)} \|^2 + \frac{\alpha}{2}\left(\| \boldsymbol{W} \|_F^2 + \| \boldsymbol{W}' \|_F^2\right) + \beta\sum_{i=1}^{n_h} D_{KL}(\rho||\tilde{\rho}_j)$$
$$D_{KL}(\rho||\tilde{\rho}_j) = \rho\ln\frac{\rho}{\tilde{\rho}_j} + (1-\rho)\ln\frac{1-\rho}{1-\tilde{\rho}_j} \tag{10}$$

where v is the number of training samples, $\|\cdot\|_F^2$ is squared Frobenius norm, n_h is the number of hidden layer nodes, ρ is the sparsity parameter, $\tilde{\rho}_j$ is the average activation of the jth hidden layer node with regard to all input nodes, and α and β are the weights of the corresponding terms.

The optimal parameters $\boldsymbol{W}_{opt}$, $\boldsymbol{b}_{opt}$, $\boldsymbol{W}'_{opt}$, $\boldsymbol{b}'_{opt}$ can be achieved by minimizing the objective function J, which is expressed as

$$\boldsymbol{W}_{opt}, \boldsymbol{b}_{opt}, \boldsymbol{W}'_{opt}, \boldsymbol{b}'_{opt} = \arg\min_{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{W}', \boldsymbol{b}'} J \tag{11}$$

Then, $\boldsymbol{W}_{opt}$ is reshaped as the convolutional kernel $\boldsymbol{k}_{opt}$, and $\boldsymbol{b}_{opt}$ is used as a bias vector in the convolutional feature mapping.

2.3. Convolutional feature mapping and pooling

Most ML models can only take one-dimensional vectors as input. Thus, the input data have to be transformed into vectors before being imported into the models. However, the vectorization process may lose information regarding the correlations between the input variables. Convolutional Neural Networks (CNN) can process the ACARS data without vectorization and accept both the information about the variables and the information about the variable correlations. A comparison between most ML models and CNN is shown in Fig. 4.

Sparse interaction is a unique property of the CNN. In a CNN, each hidden layer node is only connected to a portion of the input layer nodes, as shown in Fig. 5. In comparison with a fully connected network, a CNN can be more sensitive to local variations. Therefore, the CNN can extract local features more effectively in comparison with fully connected networks. This property makes CNN very suitable for fault

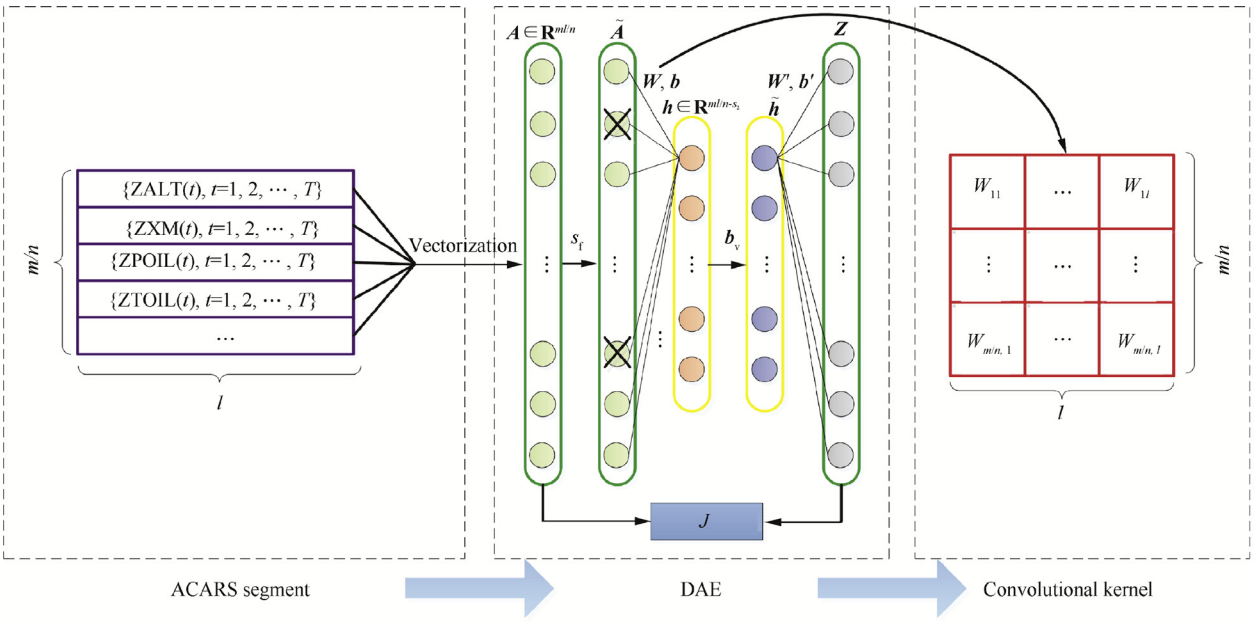

Fig. 3 Schematic diagram of convolutional kernel's unsupervised feature learning.

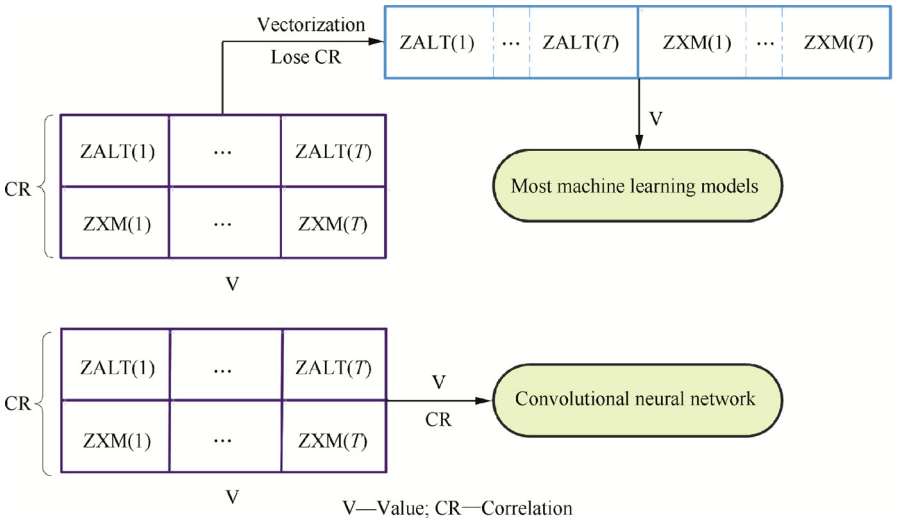

Fig. 4 Comparison between most ML models and CNN.

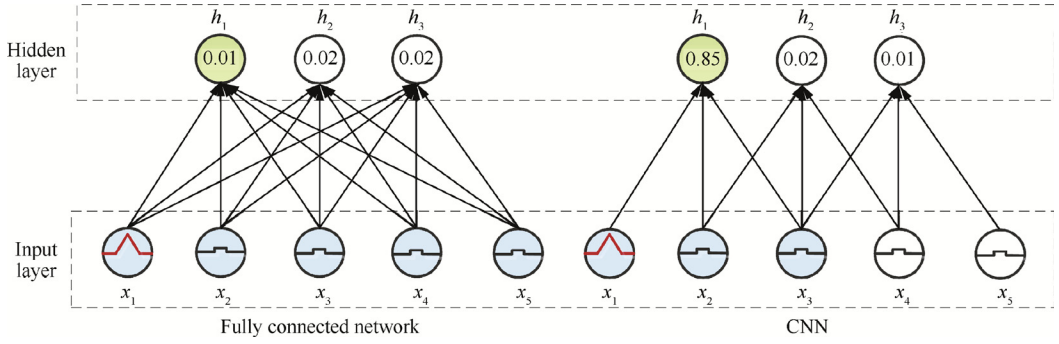

Fig. 5 Comparison between fully connected network and CNN.

detection in the ACARS data segment. The input layer shown in Fig. 5 contains the fault node x_1(red), and the four normal nodes x_2, x_3, x_4, and x_5. In a fully connected network, the hidden layer node h_1 is connected to x_1, x_2, x_3, x_4, and x_5; the activation degree of the hidden layer nodes h_1, h_2, and h_3 is low, and the fault pattern indicated in x_1 is covered up. In a CNN, h_1 is only connected to x_1, x_2, and x_3; h_1 is largely activated, and the fault is detected.

In addition to capturing local features, the CNN can also greatly reduce the number of parameters and time complexity. Assuming that the input layer and the hidden layer have r and s nodes, respectively, a fully connected network will have $(r + 1)s$ parameters and a time complexity of $O(rs + s)$. In a CNN, every node in a hidden layer is only connected to the c ($c < < r$) nodes in the input layer, the number of parameters is $(c + 1)s$, and the time complexity is $O(cs + s)$. The ratio of the number of parameters in the two types of networks is $(r + 1)/(c + 1)$. In Fig. 5, the number of parameters in a fully connected network and in a CNN is 18 and 12, respectively.

In addition to the sparsity interaction, parameter sharing is another important property of the CNN, and means that every node in the same hidden layer has the same weights and biases, as shown in Fig. 6. This not only reduces the number of parameters, but also replaces the matrix multiplication with a convolution operation, which improves the computational speed.

In this study, the sliding window method was used to generate the ACARS data segments. The length and dimension of the sliding window were l_w and u, respectively. When the window reached the ith column of the ACARS time series, an ACARS data segment was generated. The start and end points of the segment were p_i and p_{i+l_w-1}, respectively. When the window moved forward by one step, the start and end point of the new segment became p_{i+1} and p_{i+l_w}, respectively. The window kept moving forward in the ACARS time series, as shown in Fig. 7.

The convolution operation was implemented independently for each variable group. Note that the ACARS data required normalization prior to the convolution operation, because the convolutional kernels were trained on normalized ACARS data. Assuming that the length of the convolutional kernel is l_k, the convolutional feature map $\boldsymbol{f}_m \in \mathbf{R}^{1\times(l_w-l_k+1)}$ was achieved by convolving the ACARS segment $\boldsymbol{s}_{eg} \in \mathbf{R}^{u\times l_w}$ with the convolutional kernel $\boldsymbol{k}_{opt} \in \mathbf{R}^{u\times l_k}$. The calculation formula is expressed as [21]

$$\boldsymbol{f}_m = \sigma(\boldsymbol{s}_{eg} * \boldsymbol{k}_{opt} + \boldsymbol{b}_{opt}) \tag{12}$$

where * is the convolution operation symbol.

The pooling operation can make the extracted features local translation invariant, which is very useful for fault detection in the ACARS segment because we are not considering the position of faults, but rather whether a fault has occurred in the segment. Additionally, the pooling operation can select significant features, reduce the feature dimension, and increase the computational efficiency. Several pooling methods have been proposed, e.g., max pooling, average pooling, spatial pyramid pooling, stochastic pooling, and L^2 pooling.[22–24]

Similar to the convolution operation, the pooling operation is implemented individually for every variable group's convolutional feature maps. After the pooling operation, every variable group's pooling features are transformed into vectors. Then, the vectors are combined to form the feature vectors of each sample. The SVM is a powerful method for solving classification problems under the conditions of small sample size, nonlinearity, and high dimensionality. Therefore, the SVM was used to identify the fault segments in the ACARS data.

3. Results and discussion

Experiments with real ACARS data were conducted to compare the fault detection performance of our method against the performance of several competing methods. Moreover, an analysis of and discussion based on the experimental results are presented.

3.1. Dataset preparation

Different types of engines may exhibit discrepant operation characteristics. To eliminate the differences as much as possible, all data were collected from the ACARS of the CFM56-7B26 engines. Typically, the OEM sends Customer Notification Reports (CNRs) to the airlines to help them detect engine faults. The CNRs record the abnormal shifts of engine performance deviation data and infer the types of faults. Thus, the CNRs are used to label the ACARS data. The recorded ACARS parameters of the CFM56-7B26 engines are listed in Table 2.

According to the CNRs, all of the selected engines experienced at least one fault event. The fault events could be categorized into several types: Exhaust Gas Temperature (EGT) indication fault, Fuel Flow (FF) indication fault, Total Air Temperature (TAT) sensor indication fault, core deterioration fault, and fan imbalance fault. During the period of shifts, the ACARS data were treated as fault samples, while the ACARS data before the shifts were treated as normal samples. A total

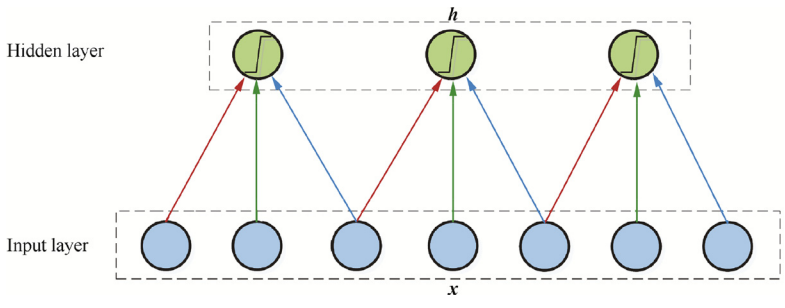

Fig. 6 Schematic diagram of parameter sharing.

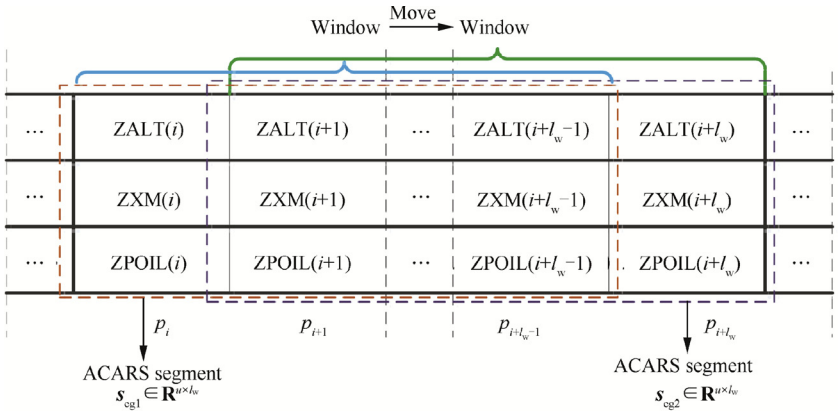

Fig. 7 Sliding window generated ACARS data segment.

Table 2 Recorded ACARS parameters of CFM56-7B26 engines.

No.	Abbreviation	Parameter	No.	Abbreviation	Parameter
1	ZALT	Altitude	11	ZTLA_D	Throttle lever angle divergence
2	ZPCN12	Indicated rotational speed of fan	12	ZTOIL	Oil temperature
3	ZPCN25	Indicated rotational speed of core	13	ZVB1F	Fan forward vibration
4	ZPCN25_D	Indicated rotational speed divergence of core	14	ZVB1R	Fan rear vibration
5	ZPHSF	Fan vibration forward phase	15	ZVB2F	Core forward vibration
6	ZPHSR	Fan vibration rear phase	16	ZVB2R	Core rear vibration
7	ZPOIL	Oil pressure	17	ZWF36	Fuel flow
8	ZT1A	Total air temperature	18	ZWF36_D	Fuel flow divergence
9	ZT49	Exhaust gas temperature	19	ZXM	Mach number
10	ZT49_D	Exhaust gas temperature divergence			

of 18,530 unlabeled samples were used to group the variables, and for the unsupervised feature learning of the convolutional kernels. Additionally, 5749 normal samples and 857 fault samples were used to train and test the SVM.

Each flight cycle is divided into five phases: takeoff, climb, cruise, descend, and landing. The engine had five types of working conditions according to the above flight phases. The acquired ACARS data only included the data recorded during the takeoff, climb, and cruise phases. Because the ambient environment and engine status were relatively stable during the cruise phase, the cruise data were smooth and contained less noise. Additionally, faults occurring under other working conditions are usually evident in the cruise data. Thus, only the cruise data were used in the experiments.

In every flight, one data point was recorded during each phase. In our study, because only the cruise data were used, the data sampling period was one time per flight. The airplanes usually operated for three to five flight cycles per day. Thus, the sampling period in the experiment was three to five times per day.

3.2. Experimental procedures

The experiments were conducted according to the steps described below.

(1) Grouping of ACARS variables

The AIVGA algorithm accepted normalized unlabeled data as input. The variable grouping results are shown in Fig. 8.

The graph shown in the left part of Fig. 8 is the dendrogram of the variable grouping results. The graph shows the group results in each step, and the dashed line represents the optimal cutting position. The right graph is the cost with regard to different group numbers. Obviously, the cost is minimal when the number of variable groups is equal to four. Table 3 presents the optimal group results with regard to the ACARS parameters.

(2) Building DAE model

One DAE was trained for each variable group. A toolbox for deep learning was used to develop the DAE model.[25] The determination of the DAE model's hyperparameters was mainly based on basic principles and many experiments. The hyperparameters of each DAE are listed in Table 4.

(3) Convolutional feature mapping and pooling

The length of each ACARS segment was 30. The max pooling method was chosen after a large number of experiments. The best results were obtained when the pooling dimension was equal to five.

(4) Support Vector Machine

The powerful LIBSVM toolbox was adopted in the experiment.[26] The SVM parameters are shown in Table 5.

The given data were divided into training and test sets. Owing to the limited number of samples, a five-fold cross-validation approach was adopted to select the best model.

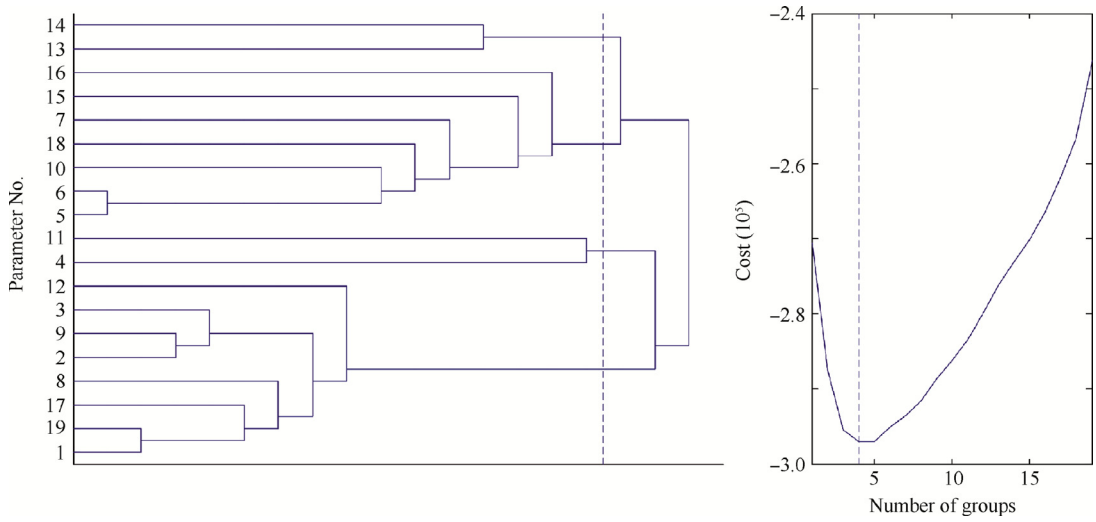

Fig. 8 Variable grouping results.

Table 3 Optimal group results for ACARS parameters.

Parameter	Group	Parameter	Group
ZALT	1	ZPHSF	3
ZPCN12	1	ZPHSR	3
ZPCN25	1	ZPOIL	3
ZT1A	1	ZT49_D	3
ZT49	1	ZVB2F	3
ZTOIL	1	ZVB2R	3
ZWF36	1	ZWF36_D	3
ZXM	1	ZVB1F	4
ZPCN25_D	2	ZVB1R	4
ZTLA_D	2		

Therefore, the number of training samples and test samples for the SVM was 5284 and 1322, respectively.

(5) Selection of evaluation indices

The dataset that was used in our experiment was severely unbalanced. In this situation, the precision, recall, and F1 scores were considered to be good indicators of the fault detection performance. The equations used to calculate them can be found in the Ref. [27].

All algorithms were implemented in MATLAB R2010b, with a computer system that comprised two Intel Core i5 (2.3 GHz) CPUs with 8 GB RAM and Windows 10 Professional.

Table 5 SVM parameters.

Parameter	Value
SVM type	C-SVC
Kernel function type	Radial basis function
Cost	1
Gamma	2

3.3. Experimental results and discussion

Table 6 shows the average of a five-fold cross-validation of the experimental results. In Table 6, CDAE is the abbreviation of Convolutional Denoising AutoEncoder.

As shown in Table 6, the SVM achieved the best fault detection results on training data, but the worst results on test data. This occurred because the dimension of the samples was too high, and a large redundancy existed between the parameters. These two factors led to overfitting when the SVM method was used. By adding the DAE before the SVM, the dimension of the features was reduced. Moreover, the redundancy and noise in the data were partially eliminated. Thus, the second method achieved better results with the test set, in comparison with the results obtained by the first method.

Table 4 Hyperparameters of each DAE.

Hyperparameter	DAE 1	DAE 2	DAE 3	DAE 4
Number of input layer nodes	49	4	36	4
Number of hidden layer nodes	30	3	25	3
Number of output layer nodes	49	4	36	4
Activation function	Sigmoid	Sigmoid	Sigmoid	Sigmoid
Learning rate	1	1	1	1
Noise rate	0.05	0.05	0.05	0.05
Epochs	10	10	10	10
Batch size	100	100	100	100

Because sparsity interaction networks are good at extracting the local features of data, the latter three methods performed better in fault detection, in comparison with the former two methods. The boxplots of the latter three methods' precision, recall, and F1 scores with the test set are shown in Fig. 9(a)–(c), respectively.

In Fig. 9(a)–(c), it can be seen that even though our method has slightly inferior precision, it outperforms the other two methods with regard to both the recall and F1 scores. In comparison with the precision and recall scores, the F1 score is a better indicator of the comprehensive fault detection performance. Therefore, our method has the best comprehensive fault detection performance among the latter three methods. Moreover, in the engineering practice of engine health management, the recall is always more important than the precision. Even though slightly lower precision may increase the workload, slightly lower recall will decrease the flight safety. Undoubtedly, in most cases, safety is always more important. Therefore, in comparison with the other two methods, the proposed method is more suitable for the engineering practice of engine health management.

Some methods described in the literature are good at extracting image representations.[12,13] However, the structure of the ACARS time series and the images are discriminative. Generally, the shorter the distance is between the pixels in an image, the stronger the correlations are. However, the above rule can only apply to the time axis of an ACARS time series, and the correlations between the variables have nothing to do with their distance. Thus, this method did not perform well in our experiments.

In comparison with Chen's method,[17] our method introduces a variable grouping operation, and the process of feature extraction is performed independently on each variable group. Thereby, the features extracted from the variable groups can better represent the distribution spaces of the ACARS data. Therefore, our method has better comprehensive fault detection performance, in addition to being more robust.

A comparison between our method and Chen's method with regard to the computational time and number of parameters is presented in Table 7. The number of parameters in our method (four variable groups) was approximately equal to one quarter of the number of parameters in Chen's method, as presented in Table 2. In Table 7, the total computational time means the time spent on conducting the five-fold cross-validation experiments. Obviously, the time cost of our method is less than that of Chen's method. Moreover, the results agree with the theoretical analysis in Section 2. As we can see, the single test time of our method was 2.92 s (with 1322 samples). Thus, our method can meet the real time pro-

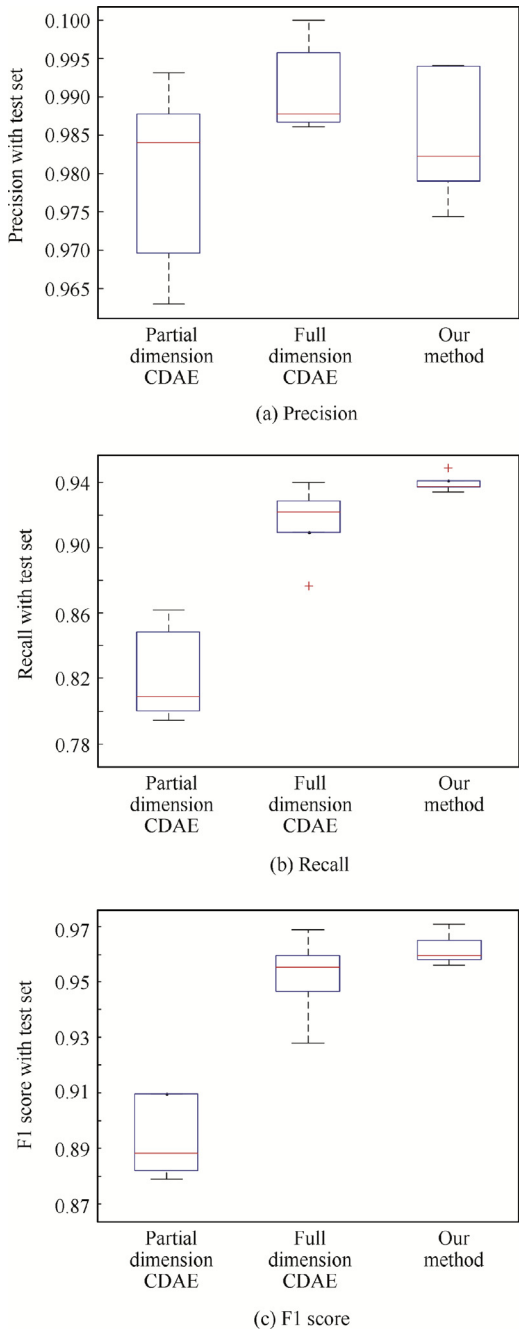

Fig. 9 Boxplots of three methods' precision, recall and F1 score with test set.

Table 6 Average of five-fold cross-validation of experimental results.

Method	Train precision	Train recall	Train F1 score	Test precision	Test recall	Test F1 score
SVM	**1**	**0.9974**	**0.9987**	Error	0	Error
DAE + SVM	0.9688	0.7840	0.8666	0.9173	0.6626	0.7686
Partial dimension CDAE	0.9961	0.8997	0.9455	0.9744	0.8334	0.8982
Full dimension CDAE	0.9988	0.9933	0.9960	**0.9901**	0.9167	0.9518
Our method	0.9979	0.9951	0.9965	0.9851	**0.9393**	**0.9616**

Note: An error occurred because the algorithm identified all samples as negative. The highest scores in each column are printed in bold fonts.

Table 7 Computational time and number of parameters.

Method	Total computational time (s)	Single training time (s)	Single test time (s)	Number of parameters
Full dimension CDAE	269.81	65.80	3.32	9200
Our Method	202.16	58.05	2.92	2356

cess requirements and can potentially be used in engineering practice.

4. Conclusions

(1) Most of the existing aircraft engine fault detection methods are based on performance deviation data provided by the OEM. To improve independent engine fault detection, this study proposed a novel aircraft engine fault detection method based on original ACARS data. The proposed method achieved good fault detection results without depending on the OEM.

(2) Previous convolutional autoencoder models cannot be adapted to high-dimensional ACARS data. Thus, a grouped convolutional denoising autoencoder model was proposed in this study. This model could extract more representative and robust features from ACARS data, and was also able to reduce the computational and time costs. Finally, our method does not require much expertise and data preprocessing. With these advantages, our method can be applied to real world scenarios. In engineering practice, an SVM and several CNN models can be trained offline with historical data. After the SVM and CNN models have been established, the engine fault detection can be conducted online to detect certain fault types which occurred in historical data.

(3) Experiments with real ACARS data of CFM56-7B26 engines were conducted to validate the effectiveness of our method. The results reveal that our method has superior comprehensive fault detection performance and robustness, in comparison with competing methods. Therefore, it is more suitable for practical engine health management applications. Additionally, the number of parameters and the computational time in our method are much less than those of the existing convolutional autoencoder method.[17]

(4) Fault detection is just the first step in our research. The ACARS data provide us with abundant engine status information that can be used for research purposes. Therefore, in future work, we plan to collect more comprehensive fault cases to locate the fault source and estimate the fault situation. Moreover, we are interested in modifying our method to make it adapt to Quick Access Recorder (QAR) data. In comparison with the ACARS data, QAR data have a much shorter sampling interval and much larger number of parameters. These two characteristics make the QAR data more suitable for the detection of aircraft engine faults. However, fault detection based on QAR data remains a challenging research subject.

Acknowledgements

This work was co-supported by the Key Program of National Natural Science Foundation of China (No. U1533202), the Civil Aviation Administration of China (No. MHRD20150104) and Shandong Independent Innovation and Achievements Transformation Fund (No. 2014CGZH1101).

References

1. Zhong SS, Luo H, Lin L, Fu XY. An improved correlation-based anomaly detection approach for condition monitoring data of industrial equipment. *ICPHM 2016: 2016 IEEE international conference on prognostics and health management*; 2016 Jun 20–22; Ottawa, Canada. Piscataway: IEEE Press; 2016. p. 1–5.
2. Tahan M, Tsoutsanis E, Muhammad M, Karim ZAA. Performance-based health monitoring, diagnostic and prognostics for condition-based maintenance of gas turbines: A review. *Appl Energy* 2017;**198**:122–44.
3. Yan WZ, Yu LJ. On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. *Proceedings of the annual conference of the prognostics and health management society*;2015. p. 1–8.
4. Li Z, Zhong SS, Lin L. Novel gas turbine fault diagnosis method based on performance deviation model. *J Propuls Power* 2016;**33**(3):1–10.
5. Yan WZ. One-class extreme learning machines for gas turbine combustor anomaly detection. *IJCNN 2016: 2016 international joint conference on neural networks*; 2016 Jul 24–29; Vancouver, Canada. Piscataway: IEEE Press; 2016. p. 2909–14.
6. Zhao NB, Wen XY, Li SY. A review on gas turbine anomaly detection for implementing health management. New York: ASME; 2016. Report No.: GT2016-58135.
7. Cory J, Beachkofski B, Cross C. Autoregression-based turbine engine anomaly detection. Reston: AIAA; 2007. Report No.: AIAA-2007-5107.
8. Chakraborty S, Sarkar S, Ray A, Phoha S. Symbolic identification for anomaly detection in aircraft gas turbine engines. *Proceedings of the 2010 American control conference*; 2010 Jun 30–Jul 2; Baltimore, USA. Piscataway: IEEE Press; 2010. p. 5954–9.
9. Eklund NHW, Hu X. Intermediate feature space approach for anomaly detection in aircraft engine data. *11th international conference on information fusion*; 2008 Jun 30–Jul 3; Cologne, Germany. Piscataway: IEEE Press; 2008. p. 1–7.
10. Kumar A, Banerjee A, Srivastava A, Goel N, Goel A. Gas turbine engine operational data analysis for anomaly detection: Statistical vs. neural network approach*CCECE 2013: 26th IEEE Canadian conference on electrical and computer engineering*; 2013 May 5–8; Regina, Canada. Piscataway: IEEE Press; 2013. p. 1–5.
11. Zaher A, McArthur SDJ, Infield DG, Patel Y. Online wind turbine fault detection through automated SCADA data analysis. *Wind Energy* 2010;**12**(6):574–93.
12. Masci J, Meier U, Ciresan D, Schmidhuber J. Stacked convolutional auto-encoders for hierarchical feature extraction. *ICANN 2011: International conference on artificial neural networks*; 2011 Jun 14–17; Espoo, Finland. Berlin: Springer; 2011. p. 52–9.

13. Du B, Xiong W, Wu J, Zhang LF, Zhang LP, Tao DC. Stacked convolutional denoising auto-encoders for feature representation. *IEEE Trans Cybern* 2017;**47**(4):1017–26.
14. Baccouche M, Mamalet F, Wolf C, Garcia C, Baskurt A. Spatio-temporal convolutional sparse auto-encoder for sequence classification. *BMVC 2012: 23rd British machine vision conference*; 2012 Sep 3–7; Guildford, England. Berlin: Springer; 2012. p. 1–12.
15. Holden D, Saito J, Komura T, Joyce T. Learning motion manifolds with convolutional autoencoders. *SIGGRAPH 2015 Asia technical briefs*; 2015 Nov 2–6; Kobe, Japan. New York: Association for Computing Machinery; 2015. p. 18–22.
16. Chen K, Seuret M, Liwicki M, Hennebert J, Ingold R. Page segmentation of historical document images with convolutional autoencoders. *ICDAR 2015: 13th international conference on document analysis and recognition*; 2015 Aug 23–26; Tunis, Tunisia. Piscataway: IEEE Press; 2015. p. 1011–5.
17. Chen KJ, Hu J, He JL. Detection and classification of transmission line faults based on unsupervised feature learning and convolutional sparse autoencoder. *IEEE Trans Smart Grid* 2018;**9**(3):1748–58.
18. Honkela A, Seppa J, Alhoniemi E. Agglomerative independent variable group analysis. *Neurocomputing* 2007;**71**(7):1311–20.
19. Yi S, Ju J, Yoon MK, Choi J. Grouped convolutional neural networks for multivariate time series [Internet]. [cited 2017 Dec 9]. Available from: https://arxiv.org/abs/1703.09938.
20. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. *Commun ACM* 2017;**60**(6):84–90.
21. Wang XF, Dong XM, Kong XW, Li JM, Zhang B. Drogue detection for autonomous aerial refueling based on convolutional neural networks. *Chin J Aeronaut* 2017;**30**(1):380–90.
22. He KM, Zhang XY, Ren SQ, Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans Pattern Anal Mach Intell* 2015;**37**(9):1904–16.
23. Zeiler MD, Fergus R. Stochastic pooling for regularization of deep convolutional neural networks. *ICLR-13: proceedings of the international conference on learning representations*; Scottsdale, USA. 2013.
24. Boureau YL, Ponce J, LeCun Y. A theoretical analysis of feature pooling in visual recognition. *ICML-10: Proceedings of the 27th international conference on machine learning*; Haifa, Israel. 2010. p. 111–8.
25. Palm RB. Prediction as a candidate for learning deep hierarchical models of data [dissertation]. Kongens Lyngby: Technical University of Denmark; 2012.
26. Chang CC, Lin CJ. LIBSVM: A library for support vector machines. *ACM Trans Intell Syst Technol* 2011;**2**(3):1–27.
27. Luo H, Zhong SS. Gas turbine engine gas path anomaly detection using deep learning with Gaussian distribution. *PHM-Harbin 2017: 2017 prognostics and system health management conference*; 2017 Jul 9–12; Harbin, China. Piscataway: IEEE Press; 2017. p. 1–6.