Mechanical Systems and Signal Processing 100 (2018) 439–453

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

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

A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load

Wei Zhang, Chuanhao Li, Gaoliang Peng *, Yuanhang Chen, Zhujun Zhang

State Key Laboratory of Robotics and System, Harbin Institute of Technology, No. 92 Xidazhi Street, Harbin 150001, Heilongjiang Province, China

ARTICLE INFO

Article history:
Received 25 January 2017
Received in revised form 16 June 2017
Accepted 19 June 2017
Available online 4 August 2017

Keywords:
Intelligent fault diagnosis
Convolutional neural networks
Load domain adaptation
Anti-noise
End-to-end

ABSTRACT

In recent years, intelligent fault diagnosis algorithms using machine learning technique have achieved much success. However, due to the fact that in real world industrial applications, the working load is changing all the time and noise from the working environment is inevitable, degradation of the performance of intelligent fault diagnosis methods is very serious. In this paper, a new model based on deep learning is proposed to address the problem. Our contributions of include: First, we proposed an end-to-end method that takes raw temporal signals as inputs and thus doesn't need any time consuming denoising preprocessing. The model can achieve pretty high accuracy under noisy environment. Second, the model does not rely on any domain adaptation algorithm or require information of the target domain. It can achieve high accuracy when working load is changed. To understand the proposed model, we will visualize the learned features, and try to analyze the reasons behind the high performance of the model.

1. Introduction

Machine health monitoring is of great importance in modern industry. Failure of these machines could cause great economical loss, and sometimes poses threats to the people who work with the machines. Therefore, in order to keep the industrial machines working properly and reliably, demand for better and more intelligent machine health monitoring technique has never ceased [1–3]. Rolling element bearings are the core components in rotating mechanism, whose health conditions, for example, the fault diameters in different places under different loads could have enormous impact on the performance, stability and life span of the mechanism. The most common way to prevent possible damage is to implement a real-time monitoring of vibration when the rotating mechanism is in operation [4,5].

In recent years, Deep Learning techniques have achieved huge success in Computer Vision [6,7] and Speech Recognition [8,9]. Some deep learning techniques have already found their way into machine health monitoring systems. Lu et al. presented a detailed empirical study of stacked denoising autoencoders with three hidden layers for fault diagnosis of rotary machinery components [10]. In their work, they evaluated the effects of receptive input size, deep structure, sparsity constraint and denoising operation on the performance. Some researchers focused on AE models using frequency domain features as inputs instead of AE models directly fed with raw signals. Jia et al. fed the frequency spectra of time-series data

* Corresponding author.
E-mail addresses: zw1993@hit.edu.cn (W. Zhang), li_chuanhao@126.com (C. Li), pgl7782@hit.edu.cn (G. Peng), cyh.wne@gmail.com (Y. Chen), zhangzhujun36@126.com (Z. Zhang).

http://dx.doi.org/10.1016/j.ymssp.2017.06.022
0888-3270/

into SAE for rotating machinery diagnosis [11], because frequency spectra may better discriminate the health conditions of rotating machinery. Zhu et al. proposed a SAE based DNN for hydraulic pump fault diagnosis that uses frequency features generated by Fourier transform [12]. Liu et al. uses normalized spectrum generated by STFT of sound signal as inputs of a 2-layer SAE based DNN. Some researchers [13,14] feed multi-domain statistical features including time domain features, frequency domain features and time-frequency domain features into SAE as a way of feature fusion.

Convolutional neural networks or CNNs are first proposed by LeCun [15], which aims at processing data with known grid-like shape, such as 2D image data, or 1D time-series data. In recent years, CNNs have been successfully applied to image recognition. Many CNNs architectures such as VGG-net [16], Res-net [17] and inception v4 [18] are proposed. However, these CNNs are not compatible with 1-D vibration signals. For example, all those models use two successive $3 * 3$ convolutional layers, which can gain a receptive region of the same size with that of a $5 * 5$ convolutional kernel while using only $2 * 3 * 3$ weights. However, for 1-D signal, two $3 * 1$ convolutional layers have a $5 * 1$ receptive region at the price of $2 * 3$ weights. Besides, for inception v4, successive $1 * 7$ and $7 * 1$ convolutional layers are used, but for 1-D vibration signal, $7 * 1$ convolutional operation can't be performed. The comparisons above show that classical 2-D CNN model may lose its advantages on 1-D signals. Therefore, a new CNN model is necessary for 1-D vibration signal. Since 2015, CNN has been used for fault diagnosis. Janssens et al. used a shallow structure with one convolutional layer ($2 * 64 * 32$) and one fully-connected layer (200 neurons) to diagnose bearing health conditions [19], whose input is the DFT of normalized vibration signal collected by two accelerators placed perpendicular to one another, and the outputs are four classification categories representing Healthy bearing (HB), Mildly inadequately lubricated bearing (MILB), Extremely inadequately lubricated bearing (EILB), and Outer raceway fault (ORF), respectively. Abdeljaber et al. proposed a method to detect and localize structural damage using 1D CNN [20]. The experiment was conducted using QU grandstand simulator, with 30 accelerometers installed on the main girders at the 30 joints. In this paper, an adaptive 1D CNN architecture is devised in order to fuse feature extraction and learning (damage detection) phases of the raw accelerometer data. In [21], we proposed a 5-layer WDCNN model with large first-layer kernels and small following kernels to diagnose the fault of bearings with huge number of training data, which can achieve high accuracy even in noisy environment.

However, currently in the field of fault diagnosis, the study of Deep Learning technique usually focused on how to use less training samples to learn more information, so that it can achieve higher performance than traditional Machine Learning algorithms [11,22,23]. However, this is not consistent with real world situation where in industry, it's easy to acquire large amount of data, and also with the help of data augmentation technique [24], we can further increase the size of training samples. Since the amount of data is no longer a problem, no matter we use traditional algorithms or deep learning algorithm, almost all of them can reach nearly 100% accuracy. Therefore, more attention should be paid on the adaptation ability of the algorithm in real world industrial environment. In the field of fault diagnosis, algorithm's adaptability can be evaluated in two aspects: First, when the working load of a machine is changed, can the model trained under one working load still achieve high accuracy if it is tested on samples from another working load [25,26]? Second, since noise is an evitable problem, then can the model trained with samples that has no noise achieve high accuracy when testing on noisy samples [27,28]? In the experiments in following sections, it is shown that even the state-of-the-art DNN [11] fails to diagnose properly in noisy environment, and the performance of our former model WDCNN decreases rapidly without the help of AdaBN algorithm which requires statistical knowledge of the whole test data. Therefore, we can see that currently, Deep Learning models have not solved this problem yet.

In order to address the problems above, in this paper, we proposed a method named Convolution Neural Networks with Training Interference (TICNN). The contributions of this paper are summarized below.

(1) We proposed a novel and simple learning framework with tricks that are easy to implement, and this model works directly on raw temporal signals.
(2) This algorithm performs pretty well under the noisy environment, and works directly on raw noisy signals without any pre-denoisng methods.
(3) This proposed algorithm has strong domain adaptation capacity, and therefore can achieve high accuracy under different working load.
(4) We try to explore the inner mechanism of proposed TICNN model in mechanical feature learning and classification by visualizing the feature maps learned by TICNN.

The remainder of this paper is organized as follows. A brief introduction of CNN is provided in Section 2. The intelligent diagnosis method based on TICNN is introduced in Section 3. Some experiments are conducted to evaluate our method against some other methods. After this, visualization about the proposed model is presented in Section 4. We draw the conclusions and the future work in Section 5.

2. Convolutional neural network with batch normalization

Convolutional neural network is a multi-stage neural network which consists of some filter stages and one classification stage [26]. The filter stage is used to extract features from the inputs, which contains four kinds of layers, the convolutional

layer, batch normalization [30] layer, activation layer and the pooling layer. The classification stage is a multi-layer perceptron, which is composed of several fully-connected layers.

2.1. Convolutional operation

The convolutional layer convolves the input local regions with filter kernels and then followed by the activation unit to generate the output features. Each filter uses the same kernel to extract the local feature of the input local region, which is usually referred to as weight sharing in literature. One filter corresponds to one frame in the next layer, and the number of frames is called the depth of this layer. We use $\mathbf{K}_i^l$ to denote the weights of the i-th filter kernel in layer l, and use $\mathbf{x}^{l(\mathbf{r}^j)}$ to denote the j-th local region in convolutional layer l. Many neural network libraries implement a related function called the cross-correlation, which is the same as convolution but without flipping the kernel, the convolution process is described as follows:

$$\mathrm{y}^{l(i,j)} = \mathbf{K}_i^l * \mathbf{x}^{l(\mathbf{r}^j)} = \sum_{j'=0}^{W} \mathbf{K}_i^l(j')\mathbf{x}^{l(j+j')} \tag{1}$$

where the notation $*$ denotes the dot product of the kernel and the local regions, W is the width of the kernel and $\mathbf{K}_i^l(j')$ denotes the j'-th weight or weights (if the depth of kernel is larger than 1) of the kernel.

2.2. Batch normalization

Batch normalization (BN) [30] layer is designed to reduce the shift of internal covariance and accelerate the training process of deep neural network. BN layer is usually added right after the convolutional layer or fully-connected layer and before the activation unit. Given the q-dimension input to the l-th BN layer $\mathbf{y}^l = (\mathbf{y}^{l(1)}, \ldots, \mathbf{y}^{l(q)})$, if BN layer is added right after the convolutional layer, $\mathbf{y}^{l(i)} = (y^{l(i,1)}, \ldots, y^{l(i,p)})$, and if BN layer is added right after the fully-connected layer, $\mathbf{y}^{l(i)} = y^{l(i)} = y^{l(i,1)}$. As shown above, in BN operation, fully-connected layer can be treated as a special kind of convolutional layer, where the number of neurons q in each feature map equals 1. The transformation of BN layer is described as follows:

$$\begin{aligned} \hat{y}^{l(i,j)} &= \frac{y^{l(i,j)} - \mu_{\mathcal{B}}}{\sqrt{(\sigma_{\mathcal{B}}^2 + \epsilon)}} \\ z^{l(i,j)} &= \gamma^{l(i)}\hat{y}^{l(i,j)} + \beta^{l(i)} \end{aligned} \tag{2}$$

where $z^{l(i,j)}$ is the output of one neuron response, $\mu_{\mathcal{B}} = \mathrm{E}[y^{l(i,j)}]$, $\sigma_{\mathcal{B}}^2 = \mathrm{Var}[y^{l(i,j)}]$, ϵ is a small constant added for numerical stability, $\gamma^{l(i)}$ and $\beta^{l(i)}$ are the scale and shift parameters to be learned, respectively.

2.3. Activation operation

After the convolution operation, activation function is essential. It enables the network to acquire a nonlinear expression of the input signal to enhance the representation ability and make the learned features more dividable. In recent years, Rectified Linear Unit (ReLU) is widely used as activation unit to accelerate the convergence of the CNNs. ReLU makes the weights in the shallow layer more trainable when using back-propagation learning method to adjust the parameters. The formula of ReLU is described as follows.

$$a^{l(i,j)} = f(z^{l(i,j)}) = \max\{0, z^{l(i,j)}\} \tag{3}$$

where $z^{l(i,j)}$ is the output value of BN operation and $a^{l(i,j)}$ is the activation of $z^{l(i,j)}$.

2.4. Pooling layer

It is common to add a pooling layer after a convolutional layer in the CNN architecture. It functions as a downsampling operation which reduces the spatial size of the features and the parameters of the network. The most commonly used pooling layer is max-pooling layer, which performs the local max operation over the input features, to reduce the parameters and obtain location-invariant features. The max-pooling transformation is described as follows:

$$p^{l(i,j)} = \max_{(j-1)W+1 \leqslant t \leqslant jW}\{a^{l(i,t)}\} \tag{4}$$

where $a^{l(i,t)}$ denotes the value of t-th neuron in the i-th frame of layer l, $t \epsilon [(j-1)W+1, jW]$, W is the width of the pooling region, and $p^{l(i,l)}$ denotes the corresponding value of the neuron in layer l of the pooling operation.

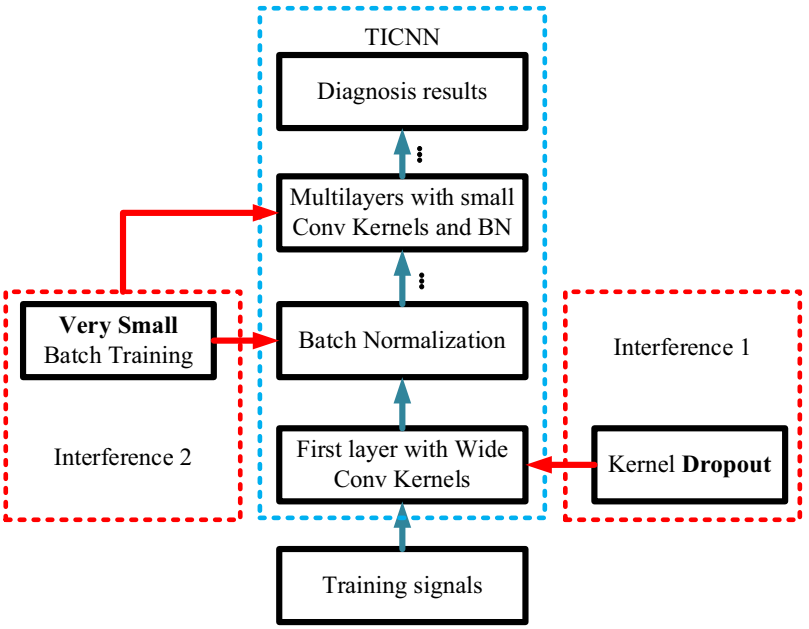


Fig. 1. The overall framework of proposed TICNN.

3. Proposed intelligent diagnosis method

The overall framework of proposed TICNN is shown in Fig. 1. We have used two tricks to serve as interference during the training process to enhance the anti-noise and domain adaptation ability of our model. First, dropout is used in the first-layer kernel. Second, very small batch training is used for the optimization algorithm. Last, ensemble learning is used to enhance the stability of our algorithm. Details of each part are illustrated in the following subsections.

As shown in Fig. 2, the input of the CNN is a segment of normalized bearing fault vibration temporal signal. The first convolutional layer extracts features from the input raw signal without any other transformation. The overall architecture of proposed TICNN model is the same as normal CNN models. It is composed of some filter stages and one classification stage. The major difference is that, in the filter stages, the first convolutional kernels are wide, and the following convolutional kernels are small (specifically, $3 * 1$). The wide kernels in the first convolutional layer can better suppress high frequency noise

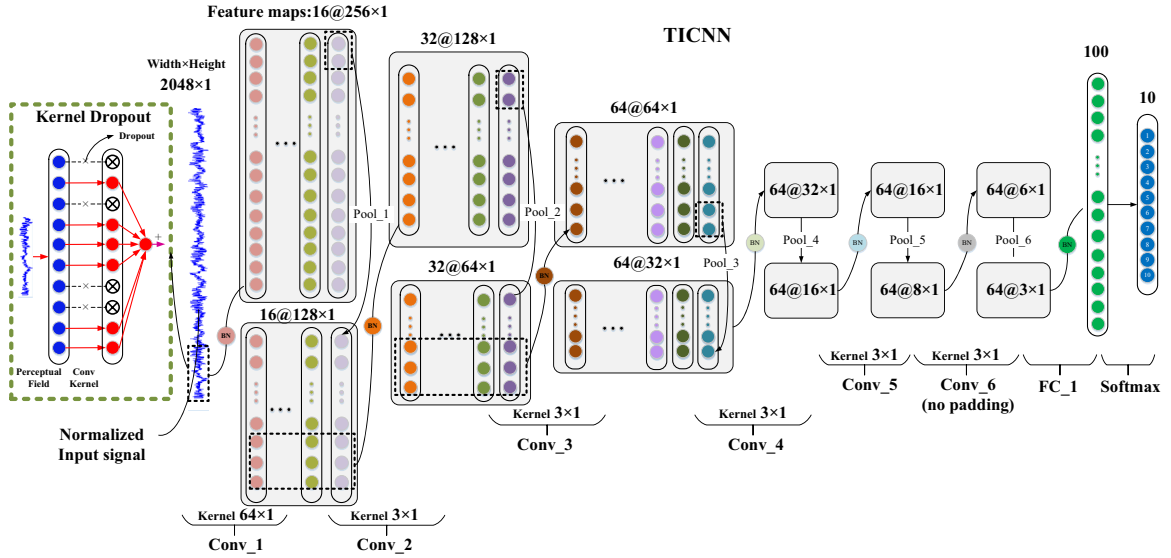


Fig. 2. Structure of the proposed TICNN model.

compared with small kernels. Multilayer small convolutional kernels make the networks deeper, which helps to acquire good representations of the input signals and improve the performance of the network. Batch normalization is implemented right after the convolutional layers and the fully-connected layer to accelerate the training process.

The classification stage is composed of two fully-connected layers for classification. In the output layer, the softmax function is used to transform the logits of the ten neurons to conform the form of probability distribution for the ten different bearing health conditions. Softmax function is described as follows:

$$q(z_j) = \text{sofmax}(z_j) = \frac{e^{z_j}}{\sum_k^{10} e^{z_k}} \tag{5}$$

where z_j denotes the logits of the j-th output neuron.

3.1. Kernel with changing dropout rate

Dropout is a trick proposed by Srivastava et al. [31] to prevent the network from overfitting. It is extremely simple but effective. The main idea is to randomly deactivate neurons along with their connections with neurons in other layers with some probability p (a hyperparameter) during training, which proves to be able to prevent units from co-adapting too much. It can be interpreted as sampling a "thinned" network within the original "full" network, and only updating the parameters of the "thinned" network during training. While during testing of the network, dropout is no longer applied, which can be viewed as an averaged prediction from exponentially many thinned models. It is found that a network trained with dropout usually leads to much better generalization ability on a wide variety of classification problems during test time, compared to training with other regularization methods.

In [31], Srivastava et al. mentioned that adding dropout in the convolutional layers has achieved additional gain in performance. This may seem unnecessary for some people because convolutional layers do not have many parameters and therefore overfitting is not really a problem. However, dropout affects all layers in the network, not just the one with dropout. Adding dropout in the lower layers can provide noisy inputs for the higher fully connected layers that has many parameters to prevent them from overfitting.

The purpose of adding dropout in first layer in the proposed model is to add noise to the raw input. Our first-layer kernel is much larger than common convolutional kernel, which makes dropout more effective, because there will be more convolutional kernels deactivated at the same time. If small kernels with the size of $1 * 3$ are used, there will only be 1 or 2 kernels deactivated at the same time under fixed dropout rate. Obviously, this is not a very good way to model the addition of noise. To further capture the uncertainty feature of noise, we use changing dropout rate. To be more specific, during every batch training, dropout rate is a random value between 0.1 and 0.9. and in every five times, dropout rate is set to 1 to make the model learn about the raw input without and noise added, which is to ensure high accuracy of the proposed model under noise free environment. The details are shown in Fig. 3

$$\begin{aligned} & p \sim \text{Uniform}(0.1, 0.9) \\ & \mathbf{r}_i^1(k) \sim \text{Bernoulli(p)} \\ & \tilde{\mathbf{K}}_i^1 = \frac{\mathbf{r}_i^1 * \mathbf{K}_i^1}{p} \\ & \mathbf{z}_i^1(j) = \tilde{\mathbf{K}}_i^1 * \mathbf{x}_j^1 \end{aligned} \tag{6}$$

where $*$ denotes the element-wise product, the value of dropout rate follows uniform distribution U(0.1,0.9), and $\mathbf{r}_i^1(k)$ follows Bernoulli distribution, which is used to decide whether the k-th element in the i-th frame of the first-layer convolutional kernel $\mathbf{K}_i^1$ is dropped or not. $\mathbf{z}_i^1(j)$ is the thinned output of input $\mathbf{x}_j^1$ after convolution operation. Different from the method given by [31], the weights are scaled by $1/p$ at training time instead of timing p when testing. This operation is necessary when the dropout date changes in every batch training.

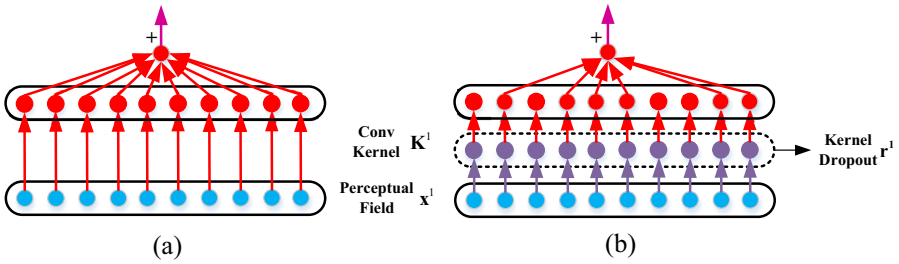


Fig. 3. Kernel dropout: (a) without dropout, (b) with dropout.

3.2. Small mini-batch training

In [32], it is found that in practice the generalization ability of the model is better when using smaller batch to compute an approximation of the gradient and then update parameters. In this paper, the selected batch size is even smaller than that is used in [32]. We set the batch size to be of the same value as the number of classification types, i.e. 10. As shown in Eq. (4), in BN layer, during every forward propagation process, input to a BN layer is subtracted by the average value of each batch, and then divided by the variance of each batch, but when testing, the statistics of the population, instead of the mini-batch are used. In [33], the average and variance of the whole training samples are replaced by the average and variance of testing samples, which improved the domain adaptation ability of the model. However, it is pretty hard to acquire the statistics of the all test samples in real world application. Our idea is that, compared with spending efforts on acquiring the statistics of test samples, it's better to train a model that can achieve relatively high accuracy even when the average and variance value used in BN is changed. This idea is based on improving the model's own generalization ability. According to this idea, we should make the statistics of batch and population be shifted as much as possible during model training, and small batch training can provide this condition. Given the q-dimension input is randomly selected to a BN layer $\mathbf{y}^l = (\mathbf{y}^{l(1)}, \ldots, \mathbf{y}^{l(q)})$, and the expected value of $y^{l(i,j)}$ is: $\mathrm{E}[y^{l(i,j)}] = \frac{1}{p|b|}\sum_{k=1}^{|b|}\sum_{j=1}^{p} y_k^{l(i,j)}$. For two batches b_1, b_2, if $|b_1| \gg |b_2|$, then according to central limit theorem, $\mathrm{Var}(\mathrm{E}[y^{l(i,j)}])_{b_1} < \mathrm{Var}(\mathrm{E}[y^{l(i,j)}])_{b_2}$. This also shows that during training, the average value of small batch has bigger fluctuations. Therefore, if the mode can achieve pretty low loss on validation, the generalization ability of the model trained with small batch will be higher.

Besides, when BN is implemented, the decrease of batch size almost has no effect on optimization algorithm. Let $H_i(x)$ denote the loss function of the input x_i of the k-th training mini batch B_k. To minimize the loss function $H_i(x)$, stochastic Gradient descent and its variants are often used for training deep models. The iterative method is shown in Formula (7). The decrease of B_k can be seen as the increase of learning rate α_k [32] has already showed that, with Batch Normalization, bigger learning rate is allowed. We will verify this in Section 4.4.

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} H(x,\theta) := \frac{1}{|B_k|}\sum_{i=1}^{|B_k|} H(x_i,\theta) \\ &\theta_{k+1} = \theta_k - \alpha_k\left(\frac{1}{|B_k|}\sum_{i=1}^{|B_k|}\nabla H(x_i,\theta)\right) \end{aligned} \tag{7}$$

3.3. Ensemble learning based on voting

Majority voting [34] is used to improve the stability and precision of our model, and reduce the influence of random initial values. This ensemble learning method operates on labels only. Assume that $d_{i,j}$ is a poll matrix of a classifier i voting on a test sample j. Then in a n-class classifying problem, $d_{i,j}$ is a matrix that only one element is 1 and others are 0, denoting which class the classifier i votes for on the test sample j. Assume there are T classifiers, then the class that receives more than half of the total polls is chosen; and if none of the class labels receives more than half of the polls, a rejection option will be given and the combined classifier makes no prediction:

$$H(x) = \begin{cases} c_j & \text{if} \sum_{i=1}^{T} d_i^j(x) > \frac{1}{2}\sum_{k=1}^{l}\sum_{i=1}^{T} d_i^k(x) \\ \text{rejection} & \text{otherwise} \end{cases} \tag{8}$$

Under the condition that the classifier outputs are independent, it can be shown that the majority voting combination will always lead to a performance improvement. If there are a total of T classifiers for an n-classes problem, the ensemble decision will be correct if the number of classifiers choosing the correct class is larger than the numbers of classifiers choosing any other classes. Now, assume that each classifier has a probability p of making a correct decision, then the probability of obtaining at least $[T/2+1]$correct classifiers out of T is:

$$P_{ens} = \sum_{k=\left(\frac{T}{2}\right)+1}^{T}\binom{T}{k}p^k(1-p)^{T-k} \tag{9}$$

Then, if $p > 0.5$, $p_{ens} \to 1$ as $T \to \infty$.

Note that the requirement $p > 0.5$ is necessary and sufficient in the majority voting. And this result is based on the assumption that the classifiers are independent, yet in practice the classifiers trained with the same data are highly correlated. Therefore, it is obvious that ensemble learning will increase the accuracy and stability of the classification to some extent.

3.4. Training of TICNN

The architecture of TICNN is designed to take advantage of the 1D structure of the input signal. Details about the architecture of TICNN can be found in Section 4.2. The training processing of 1-D CNN is detailed in our former paper, in this section, we will elaborate the training process of TICNN on kernel dropout and batch normalization.

The loss function of our CNN model is the cross-entropy between the estimated softmax output probability distribution and the target class probability distribution. No regularization term is added to the loss function, considering batch normalization, data augment, and kernel dropout have been used as regularization in this paper. Let $p(\mathbf{x})$ denote the target distribution and $q(\mathbf{x})$ denote estimated distribution, so the cross-entropy between $p(\mathbf{x})$ and $q(\mathbf{x})$ is:

$$Loss = H(p(x), q(x)) = -\sum_{x} p(x) \log q(x) \tag{10}$$

During training we need to back propagate the gradient of loss H through BN layers. When BN is added right after convolutional layers, the chain rule for BN layers is described as follows:

$$\begin{aligned}
&\frac{\partial L}{\partial \hat{y}_k^{l(i,j)}} = \frac{\partial L}{\partial z_k^{l(i,j)}} \cdot \gamma^{l(i)} \\
&\frac{\partial L}{\partial \sigma_{\mathcal{B}}^2} = \sum_{k=1}^{m}\sum_{j=1}^{p} \frac{\partial L}{\partial \hat{y}_k^{l(i,j)}} \cdot (y_k^{l(i,j)} - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2} \\
&\frac{\partial L}{\partial \mu_{\mathcal{B}}} = \left(\sum_{k=1}^{m}\sum_{j=1}^{p} \frac{\partial L}{\partial \hat{y}_k^{l(i,j)}} \cdot \frac{-1}{\sqrt{(\sigma_{\mathcal{B}}^2 + \epsilon)}} \right) + \frac{\partial L}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{k=1}^{m}\sum_{j=1}^{p} -2(y_k^{l(i,j)} - \mu_{\mathcal{B}})}{mp} \\
&\frac{\partial L}{\partial y_k^{l(i,j)}} = \frac{\partial L}{\partial \hat{y}_k^{l(i,j)}} \cdot \frac{1}{\sqrt{(\sigma_{\mathcal{B}}^2 + \epsilon)}} + \frac{\partial L}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(y_k^{l(i,j)} - \mu_{\mathcal{B}})}{mp} + \frac{\partial L}{mp \partial \mu_{\mathcal{B}}} \\
&\frac{\partial H}{\partial \gamma^{l(i)}} = \sum_{k=1}^{m}\sum_{j=1}^{p} \frac{\partial H}{\partial z_k^{l(i,j)}} \cdot \hat{y}_k^{l(i,j)} \\
&\frac{\partial H}{\partial \beta^{l(i)}} = \sum_{k=1}^{m}\sum_{j=1}^{p} \frac{\partial L}{\partial z_k^{l(i,j)}}
\end{aligned} \tag{11}$$

where the subscript k denotes the k-th sample in the training mini-batch, and m is size of mini-batch.

According to the definition in Section 2.2, when BN is added right after fully-connected layers, the value of p in Eq. (11) should be set to 1.

When $\frac{\partial H}{\partial y_k^{l(i,j)}}$ is computed, we can use chain rule to calculate the gradient of convolutional kernels with (first convolutional layer) or without (the rest convolutional layers) dropout, which is described as follows:

$$\begin{cases}
\frac{\partial H}{\partial \mathbf{K}_i^{l}(j')} = \sum_{k=1}^{m} \sum_{j} \left(\frac{\mathbf{x}^{l(j)}}{p} \right) * \left(\frac{\partial H}{\partial y_k^{l(i,j)}} \right), l = 1 \\
\frac{\partial H}{\partial \mathbf{K}_i^{l}(j')} = \sum_{k=1}^{m} \sum_{j} (\mathbf{x}^{l(j)}) * \left(\frac{\partial H}{\partial y_k^{l(i,j)}} \right), l > 1
\end{cases} \tag{12}$$

where p denotes the kernel dropout rate.

In order to minimize the loss function, the Adam Stochastic optimization algorithm is applied to train our TICNN model. Details of this optimization algorithm can be found in [35].

4. Validation of proposed TICNN model

In real world applications, the working environment of mechanical system varies a lot. There are two main variations. First, the working load may change from time to time according to the requisite of the production, so it is unrealistic to collect and label enough training samples to make the classifier robust to all the working loads. So it is significant for feature extractors and classifiers trained with samples collected in one working load to be able to classify samples from another working load. Second, the noise is unavoidable in industrial production, as the vibration signals are easily contaminated by noise. The ability to diagnose fault types under the noisy environment is crucial and challenging. In the reminder of this section, we will investigate how well the TICNN method performs under these two scenarios.

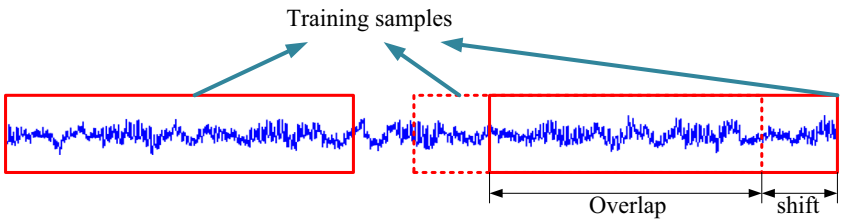


Fig. 4. Data augment with overlap.

Table 1
Description of rolling element bearing datasets.

Fault location		None	Ball			Inner race			Outer race			Load
Category labels		1	2	3	4	5	6	7	8	9	10	
Fault diameter (in.)		0	0.007	0.014	0.021	0.007	0.014	0.021	0.007	0.014	0.021	
Dataset A no.	Train	660	660	660	660	660	660	660	660	660	660	1
	Test	25	25	25	25	25	25	25	25	25	25	
Dataset B no.	Train	660	660	660	660	660	660	660	660	660	660	2
	Test	25	25	25	25	25	25	25	25	25	25	
Dataset C no.	Train	660	660	660	660	660	660	660	660	660	660	3
	Test	25	25	25	25	25	25	25	25	25	25	
Dataset D no.	Train	1980	1980	1980	1980	1980	1980	1980	1980	1980	1980	1,2,3
	Test	75	75	75	75	75	75	75	75	75	75	

4.1. Data description

As mentioned above, the deep learning algorithm in [11,22,23] achieved higher accuracy than traditional algorithm when trained with a pretty small training set, e.g. a few hundred training samples. However, this scenario is not consistent with real world fault diagnosis where thousands of training samples can be easily acquired when using overlapped samples. With the increase of training data, deep learning algorithm no longer has advantage over traditional algorithm. Besides, when trained with large dataset, the denoising ability of DNN is also not as good as that of SVM.

In this paper, a simple data augmentation technique is used to increase the number of training samples, that is, slicing the training samples with overlap. The process is shown in Fig. 4. The training samples is prepared with overlap. For example, a vibration signal with 60,000 points can provide the TICNN at most 57,953 training samples with length 2048 when the shift size is 1.

The original experiments data was obtained from the accelerometers of the motor driving mechanical system at a sampling frequency of 12 kHz from the Case Western Reserve University (CWRU) Bearing Data center [36]. There are four fault types of the bearing: normal, ball fault, inner race fault and out race fault. Each fault type contains fault diameters of 0.007 in., 0.014 in. and 0.021 in. respectively, so we have ten fault conditions in total. In this experiment, each sample contains 2048 data points, which is easy to implement FFT for the baseline algorithm. Dataset A, B and C each contains 6600 training samples and 250 testing samples of ten different fault conditions under loads of 1, 2 and 3 hp. Dataset D contains 19,800 training data and 750 testing data of all three loads. In addition, the training samples are overlapped to augment data and there is no overlap among the test samples. The details of all the datasets are described in Table 1.

4.2. Experimental setup

4.2.1. Baseline system

We compare our methods with the deep neural network (DNN) system [11] with frequency features proposed by Lei et al. in 2016. The DNN system has two steps, namely, pre-training with unsupervised stacked auto-encoder and supervised fine tuning. This neural network consists of three hidden layers. The number of neurons in each layer is 1025, 500, 200, 100 and 10. The input of the network is the normalized 1025 Fourier coefficients transformed from the raw temporal signals using Fast Fourier transformation (FFT). Softmax is used as the classifier for supervised learning.

4.2.2. Parameters of the proposed TICNN

The architecture of the proposed TICNN used in experiments is composed of 6 convolutional and pooling layers followed by a fully-connected hidden layers and a softmax layer. The size of first convolutional kernel is $64 * 1$, and the rest kernel size is $3 * 1$. The pooling type is max pooling and the activation function is ReLU. After each convolutional layer and fully-connected layer, batch normalization is used to improve the performance of TICNN. The parameters of the convolutional and pooling layers are detailed in Table 2. The experiments were implemented using Tensorflow toolbox of Google.

Table 2
Details of proposed TICNN model used in experiments.

No.	Layer type	Kernel size/stride	Kernel channel size	Output size (width * depth)	Padding
1	Convolution1	64 * 1/8 * 1	16	256 * 16	Yes
2	Pooling1	2 * 1/2 * 1	16	128 * 16	No
3	Convolution2	3 * 1/1 * 1	32	128 * 32	Yes
4	Pooling2	2 * 1/2 * 1	32	64 * 32	No
5	Convolution3	3 * 1/1 * 1	64	64 * 64	Yes
6	Pooling3	2 * 1/2 * 1	64	32 * 64	No
7	Convolution4	3 * 1/1 * 1	64	32 * 64	Yes
8	Pooling4	2 * 1/2 * 1	64	16 * 64	No
9	Convolution5	3 * 1/1 * 1	64	16 * 64	Yes
10	Pooling5	2 * 1/2 * 1	64	8 * 64	No
11	Convolution6	3 * 1/1 * 1	64	6 * 64	No
12	Pooling6	2 * 1/2 * 1	64	3 * 64	No
13	Fully-connected	100	1	100 * 1	
14	Softmax	10	1	10	

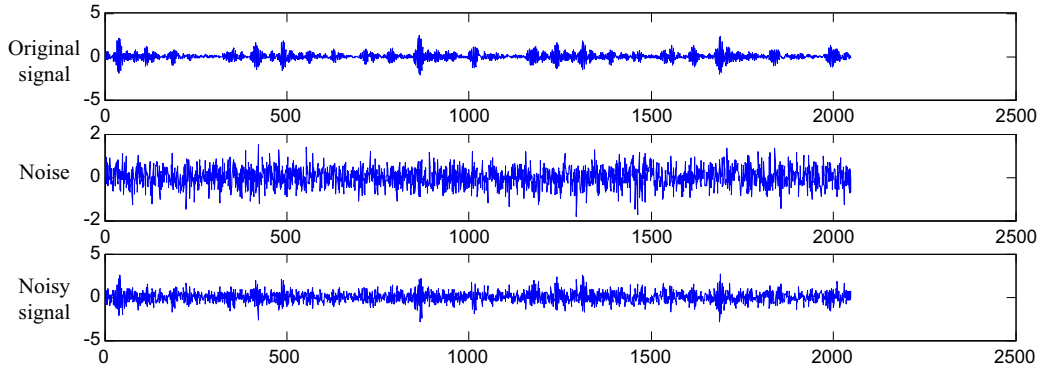


Fig. 5. Figures for original signal of inner race fault, the additive white Gaussian noise, and the composite noisy signal with SNR = 0 dB respectively.

4.3. Case study I: performance under noise environment

In this section, we will discuss the diagnosis accuracy of the proposed TICNN method under noisy environment. In our experiments, the model is trained by the original data provided by CWRU, then it is tested on the noisy data. This scenario is closer to the real world condition in industrial production, because the noise varies a lot, and we can't get all the labeled training samples under different noisy environment. We add additive white Gaussian noise to the original signals to composite signals with different SNR, and the definition of SNR is shown as follows:

$$\mathrm{SNR}_{\mathrm{dB}} = 10\log_{10}\left(\frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}}\right) \tag{13}$$

where P_{signal} and P_{noise} are the power of signal and the noise respectively.

In Fig. 5, the original signal of inner race fault is added by the additive white Gaussian noise. The SNR for the composite noisy signal is 0 dB, which means the power of noise is equal to that of the original signal. We test the proposed TICNN model with noisy signals ranging from −4 dB to 10 dB. In order to testify the necessity of the small batch training, batch size varies from 10 to 100, and the remaining structure of the network is unchanged.

The results of the proposed TICNN model diagnosing with noisy signal are shown in Table 3 and Fig. 6. Table 3 compares the accuracy of ensemble TICNN trained with different size of mini batch on test samples with different SNR values. It can be seen that, regardless of the size of mini batch used during training, all the models achieve pretty high accuracy when SNR value is high. For instance, when SRN >= 4 dB, the accuracy of all models are above 99.4%. However, as SNR value decreases, so does the accuracy of models trained with big mini batch size. When SNR value equals −4 dB, the accuracy of the model that mini batch size is 100 during training, decreases to 68%. In contrast, models that are trained with small batch size could still achieve relatively high accuracy when SNR value is low. For example, when SNR equals −4 dB, the model whose batch size is 10 during training has 82% accuracy, and its standard variation in 20 trials is only 0.66%, which is much smaller than 2.79%, the standard variation of the model whose batch size is 100. Fig. 6 compares the fault diagnosis accuracy of TICNN, SVM, MLP, DNN and WDCNN with different SNR of the input signals. All three reference algorithms are based on frequency features extracted by FFT. The kernel function of SVM is radial basis function. The number of neurons in each layer of MLP is

Table 3
The accuracy of ensemble TICNN trained with different batch size and SNR value.

Batch size	SNR(dB)							
	−4	−2	0	2	4	6	8	10
100	68.29 ± 2.79	88.39 ± 1.21	96.04 ± 0.51	98.87 ± 0.33	99.48 ± 0.21	99.64 ± 0.17	99.71 ± 0.17	99.71 ± 0.18
50	74.10 ± 1.38	91.25 ± 1.16	96.96 ± 0.50	98.99 ± 0.20	99.52 ± 0.13	99.64 ± 0.14	99.69 ± 0.11	99.75 ± 0.06
40	76.69 ± 2.24	92.71 ± 1.20	97.62 ± 0.47	99.17 ± 0.23	99.51 ± 0.14	99.53 ± 0.09	99.59 ± 0.10	**99.77 ± 0.07**
30	80.35 ± 2.97	93.85 ± 1.02	97.91 ± 0.40	**99.31 ± 0.12**	**99.61 ± 0.09**	**99.66 ± 0.11**	99.70 ± 0.10	99.69 ± 0.08
20	81.68 ± 1.63	95.55 ± 1.08	98.11 ± 0.33	99.11 ± 0.12	99.57 ± 0.09	99.61 ± 0.14	99.71 ± 0.13	99.69 ± 0.12
10	**82.05 ± 0.66**	**96.47 ± 0.34**	**98.22 ± 0.27**	99.27 ± 0.16	**99.61 ± 0.18**	99.59 ± 0.08	**99.75 ± 0.08**	99.63 ± 0.06

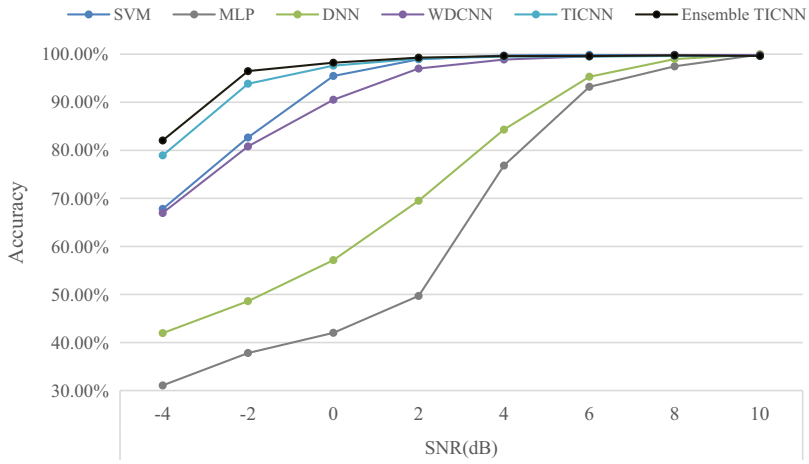


Fig. 6. Comparison of accuracy testing on signals with different SNR values.

1025, 500, 200, and 10, and its activation function is sigmoid. Fig. 6 illustrates that when trained with 19,800 training samples with no noise added, all five algorithms achieve nearly 100% accuracy on test samples with high SNR values. However, as the SNR value decreases, the accuracy of DNN and MLP also suffer from remarkable decrease. When SNR of test samples equals 0 dB, the accuracy of DNN and MLP are both below 60%, while the accuracy of the proposed TICNN still remains higher than 90% even when testing on samples with SNR value of −2 dB. This suggests that compared with the traditional intelligent fault diagnosis method and the state of the art DNN method, TICNN has much better robustness against noise. In addition, the processing time for TICNN to diagnose a signal is about 1.02 ms, and the program is performed on an i7 6700 processor at 3.4 GHz with 16 GB memory

4.4. The necessity of BN, kernel dropout and ensemble learning

In order to test the effect of the tricks used in our network, experiments are conducted to compare the performances of networks trained with BN, without BN, and network that is trained with BN and has dropout in first convolutional kernels. The mini batch size is set to 10 in this experiment. Fig. 7(a) shows the result of single network, while Fig. 7(b) shows the result of 5 ensemble networks. As can be seen from Fig. 7(a), without BN, it's hard to train the network to achieve high accuracy when the mini batch size is 10. When SNR value equals 10 dB, the diagnosis accuracy is below 95%. When SNR value equals 0 dB, the accuracy even drops below 50%, and its standard deviation in 20 trials reaches about 10%. However, after applying BN, the accuracy of the model is greatly improved. By testing on samples with high SNR value, the accuracy can be close to 100%. And the accuracy is increased to roughly about 60% with SNR being −4 dB, but its standard deviation is still 10%. For further improvement, in addition to BN, dropout is applied on first layer convolutional kernels. This time, the accuracy reaches 90% when SNR value is −2 dB, and the stability of the algorithm has been improved, with standard deviation being around 2% in 20 trials.

Fig. 7(b) shows that after using ensemble learning, the stability of all three algorithms has been improved. The ensemble version of model with BN and dropout in convolutional kernel achieves over 80% accuracy when SNR equals −4 dB, and its standard deviation is below 1% in 20 trials. When SNR is −2 dB, the accuracy is even well above 95%.

4.5. Case study II: performance across different load domain

In this set of experiments, the adaptation performance across different load domain of TICNN is tested and the ensemble learning is used to improve the accuracy of the proposed TICNN model. The results of our methods are compared with

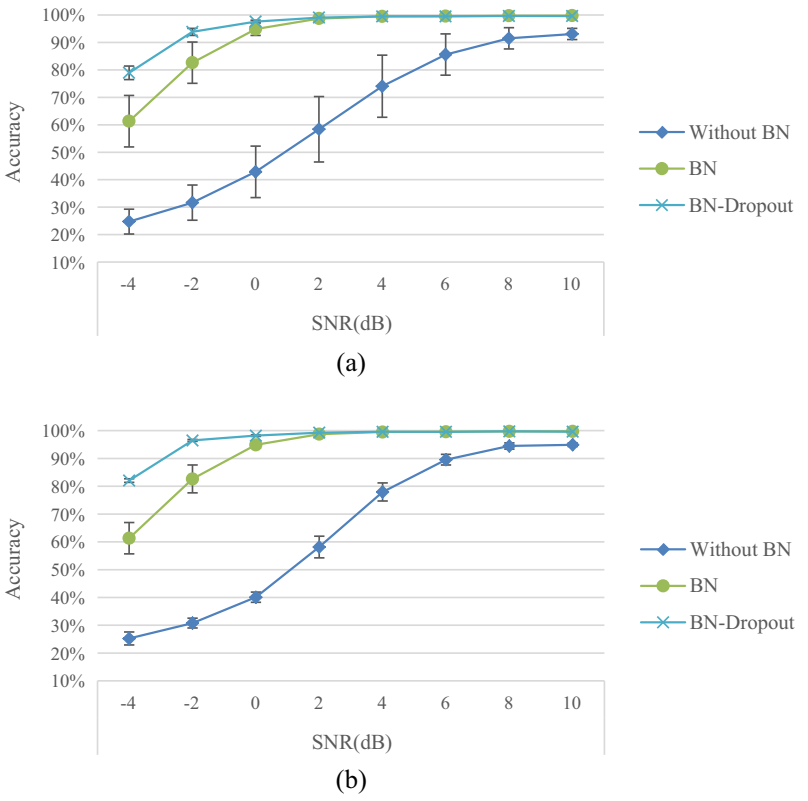


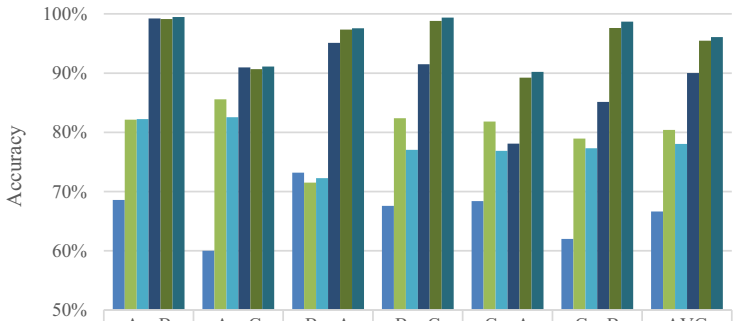
Fig. 7. Performance of models (a) using different tricks: without BN, BN, and BN-Dropout, (b) improved by majority voting.

Table 4
Description of scenario settings for load domain adaptation.

Scenario settings for load domain adaptation			
Domain types	Source domain	Target domain	
Description	Labeled signals under one single load	Unlabeled signals under another load	
Domain details	Training set A	Test set B	Test set C
	Training set B	Test set C	Test set A
	Training set C	Test set A	Test set B
Target	Diagnose unlabeled vibration signals in target domain		

traditional SVM, MLP, DNN algorithm which work in frequency domain (the data is transformed by FFT). The description of scenario settings for domain adaptation is illustrated in Table 4, and the results of the experiments are shown in Fig. 8.

As shown in Fig. 8, SVM, MLP and DNN perform poorly in domain adaptation, with average accuracy in the 6 scenarios being around 65%, 80% and 80%, respectively. In contrast, the proposed TICNN method is much more precise than the other three algorithms, achieving 95.5% accuracy in average, which proves that the features learned by TICNN from raw signals are more domain invariant than the traditional frequency features. Besides, after applying ensemble learning on TICNN, the result is improved to 96.1%, and in each scenario the accuracy is above 90%. As can be seen from Fig. 8, when adapting from Domain A to B, from B to A, from B to C, and from C to B, the classification accuracy of the proposed TICNN is significantly higher than that from A to C and C to A. This result conforms to the intuition that when the target domain is becoming more different from the source domain, the result is supposed to be lower, because the target domain becomes harder to adapt to. It is worth noticing that, MLP and DNN achieved high diagnosis accuracy from Domain A to B with about 82%, while the accuracy drops by 10% when adapting from Domain B to A. In contrast, the accuracy of proposed model only decreases by 2%, which shows a better stability than other reference algorithms. As can be seen from Figs. 6 and 8, it's interesting that SVM is robust to noise but has bad performance on domain adaptation, while MLP and DNN has good domain adaptation ability but lack robustness again noise. It seems robustness against noise and domain adaptation ability are not very easy to be compatible with each other. Compared to SVM and MLP, the proposed algorithm has achieved pretty high robustness to noise and good domain adaptation ability at the same time.



	A→B	A→C	B→A	B→C	C→A	C→B	AVG
SVM	68.6%	60.0%	73.2%	67.6%	68.4%	62.0%	66.6%
MLP	82.1%	85.6%	71.5%	82.4%	81.8%	79.0%	80.4%
DNN	82.2%	82.6%	72.3%	77.0%	76.9%	77.3%	78.1%
WDCNN	99.2%	91.0%	95.1%	91.5%	78.1%	85.1%	90.0%
TICNN	99.1%	90.7%	97.4%	98.8%	89.2%	97.6%	95.5%
Ensemble TICNN	99.5%	91.1%	97.6%	99.4%	90.2%	98.7%	96.1%

Fig. 8. Results of the proposed TICNN and ensemble TICNN of 6 domain shifts on the Dataset A, B and C, compared with SVM, MLP, DNN and WDCNN.

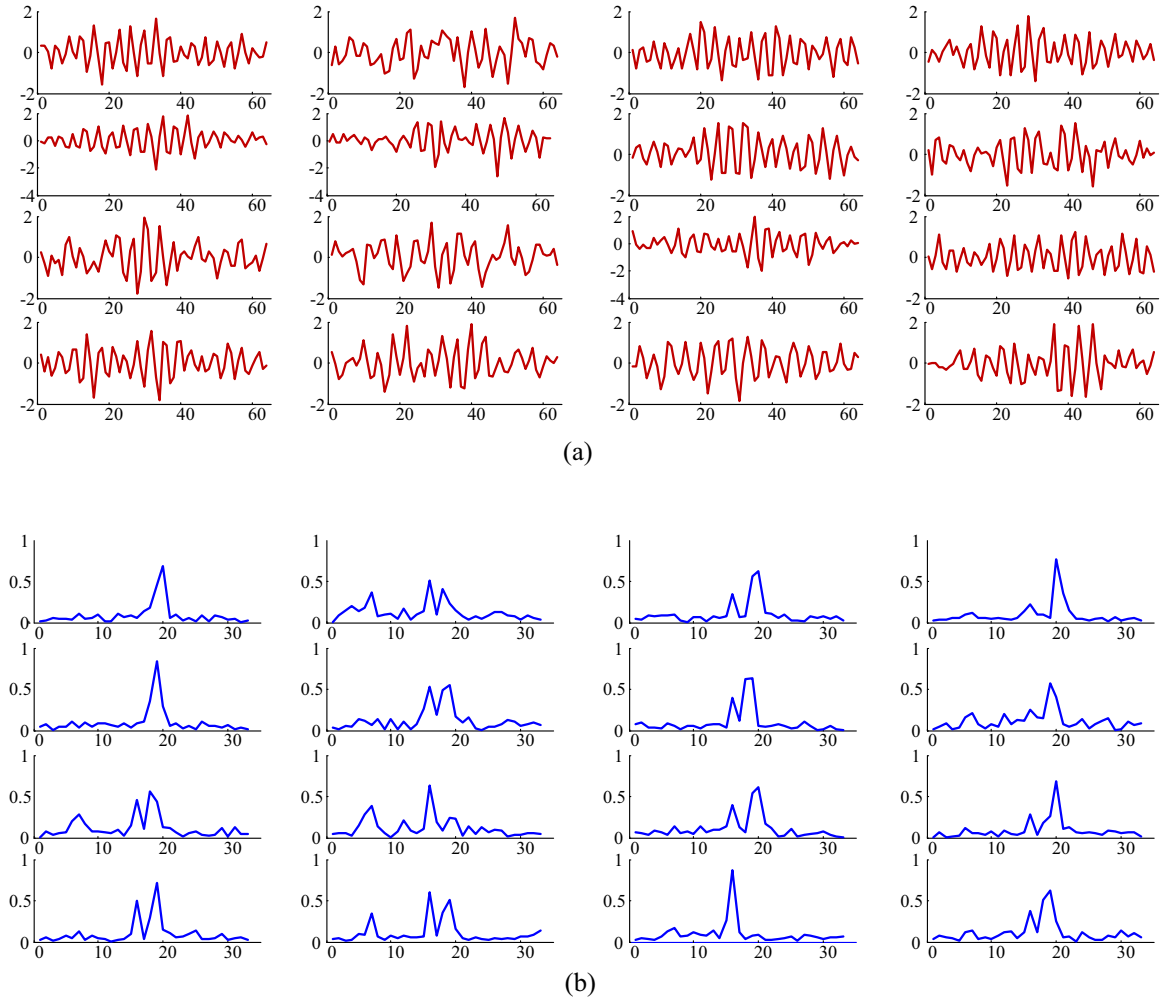


Fig. 9. Visualization of (a) first layer convolutional kernels learned by TICNN and (b) corresponding frequency-domain representations.

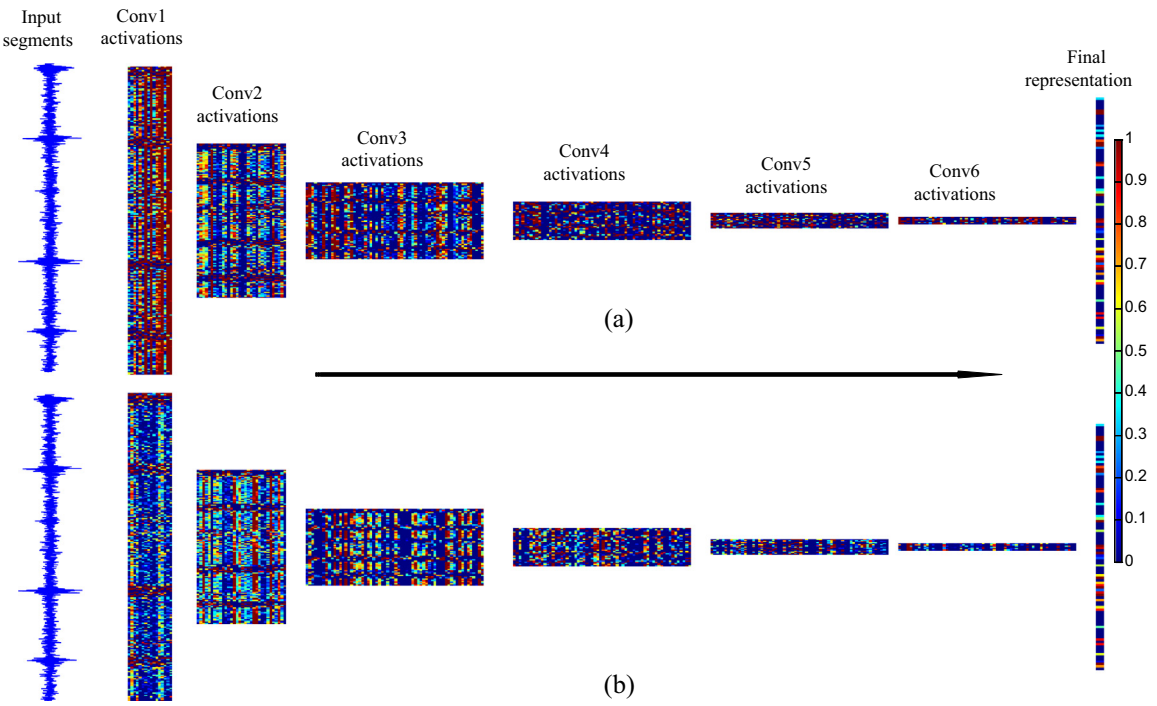


Fig. 10. Visualization of all convolutional neuron activations in TICNN (a) with training interference and (b) without training interference. The input is a C10 signal with SNR equal to 0 dB. Red represents an activation of maximum, while blue means the neuron is not activated. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.6. Networks visualizations

Generally, CNN is regard as a black box, it is hard to understand the inner operating mechanism of CNN, and none the existing fault diagnosis papers using CNN try to explore this. In this paper, we try to explore the inner operating process of the proposed TICNN model by visualize the activations in this neural network.

First, to better understand what kinds of features are extracted by the fist-layer convolutional kernels, we plot the filter kernels learned by TICNN and their frequency-domain representation in Fig 9. From the frequency representations we can see that, most of the features extracted by convolutional kernels are medium frequency features, and some may extract medium and low frequency features at the same time. Therefore, these convolutional kernels in our network can be taken as high-cut filters. This makes the network more robust when diagnosing under noisy environment. In addition, these kernels only extract features in certain frequency band. Compared to FFT which extracts features of all frequency bands, extracting features by convolutional kernels is more direct and efficient.

Second, in Fig. 10, we visualize the reactions of neurons in all convolutional layers to gain some insights of what each layer sees in TICNN. As a comparison, the visualization of a well-trained CNN with almost the same structure as TICNN except that it has no dropout in first-layer kernels and the mini-batch size is 100. The input is a C10 signal with SNR equal to 0 dB. As can be seen from the visualization, most neurons in the first convolutional layer of TICNN are active, while most neurons in the first convolutional layer of the reference CNN are not. However, the representations of the fully connected layer in both networks are the same. With the increase of depth in network, the number of active neurons in TICNN decreases, which suggests that the use of dropout and small batch training mainly influences first convolutional layer by using more neurons to describe the input samples. The advantage of this is that, even when the signal is interfered by noise or change of working load, there will still be enough active neurons to describe the input signal, thus improving the denoising and domain adaptation ability of the networks.

Last, Visualization of the feature distribution of all the test samples with SNR = 0 dB extracted from each convolutional layers and the last fully-connected layer via t-SNE method is given in Fig. 11. There are some interesting phenomena worth noticing. First, the features become more and more divisible as the layer goes deeper. It is not divisible in early layers, while in the fully-connected layer, the features are very easy to divide as we can see from Fig. 11. Second, as we can see from the visualization of FC layer 1, C2 and C4 have some overlapped region, which suggests that the model may not perform very well in discriminating between ball faults 0.007 in. and 0.021 in., but it has no problem diagnosing the position of failure. Third, as we can see from the visualization of Conv Layer 1, the feature representations of first convolutional layer are evenly distributed. It looks like the convolutional operation implemented by dropout kernel and the following batch normalization have a similar effect as a decorrelation stage. Besides, as shown in Conv Layer2, the feature points of each category starts to

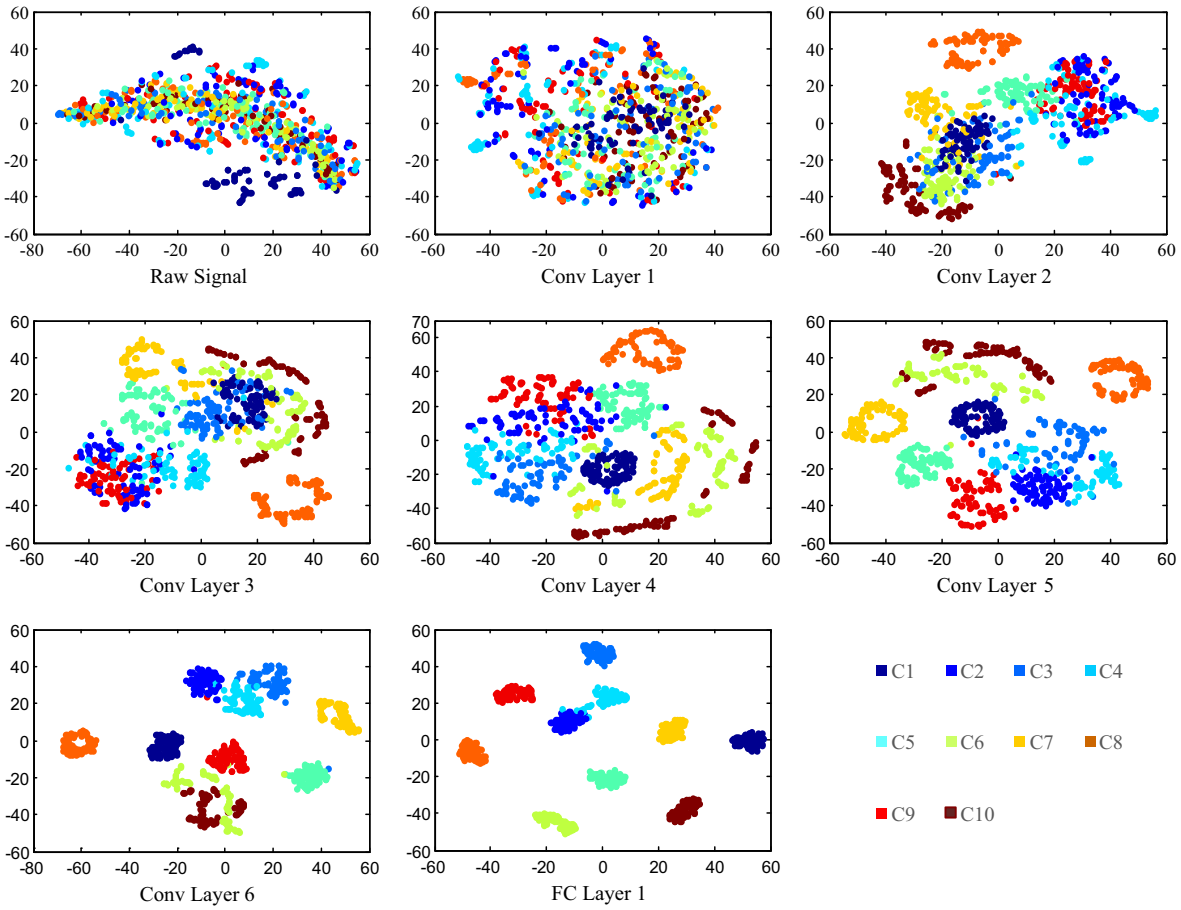


Fig. 11. Feature visualization via t-SNE: feature representations for all test signals extracted from raw signal, six convolutional layers and the last fully-connected layer respectively.

cluster together, which suggests that the following operations work as a correlation stage. This result matches with the idea of the state-of-art domain adaptation method named CORAL, whose main idea is to decorrelate first and then re-correlate the responses in networks to acquire better domain invariant features. However, our method doesn't need to add any regulation terms to achieve this and doesn't need any statistics information of target domain.

5. Conclusion

This paper proposes a new model named TICNN, to address the fault diagnosis problem. TICNN works directly on raw vibration signals without any time-consuming hand-crafted feature extraction process. TICNN has two main interferences, first-layer kernel dropout with constantly changing rate and very small batch training. With the help of data augmentation, the proposed TICNN model works well under noisy environment and performs well when the working load changes.

Results in Section 4 shows that, although state of the art DNN model could achieve pretty high accuracy on normal dataset, its performance suffer from rapid degradation under noisy environment or when working load changes. However, TICNN, with high classification accuracy, is also very robust to change of working load and noise.

Ensemble Learning has improved the stability and accuracy of the algorithm. In addition, Networks Visualizations are used to investigate the inner mechanism of the proposed TICNN model.

Compared with common denoising preprocessing algorithm and domain adaptation algorithm that requires additional information, our algorithm does not need any subsidiary algorithm. Therefore, in future work, we can try combining some denoising preprocessing and using information about the target domain to improve the performance of the proposed model.

Acknowledgements

This work was supported by National High-tech R&D Program of China (863 Program, No. 2015AA042201), National Natural Science Foundation of China (No. 51275119), and Self-Planned Task (No. SKLRS201708A) of State Key Laboratory of Robotics and System (HIT)".

References

[1] Z. Gao, C. Cecati, S.X. Ding, A survey of fault diagnosis and fault-tolerant techniques—Part I: fault diagnosis with model-based and signal-based approaches, IEEE Trans. Ind. Electron. 62 (6) (2015) 3757–3767.
[2] Y. Lei, J. Lin, M.J. Zuo, Z. He, Condition monitoring and fault diagnosis of planetary gearboxes: a review, Measurement 48 (2014) 292–305.
[3] A.C.M. Fong, S.C. Hui, An intelligent online machine fault diagnosis system, Comput. Control Eng. J. 12 (5) (2001) 217–223.
[4] B. Li, M.Y. Chow, Y. Tipsuwan, J.C. Hung, Neural-network-based motor rolling bearing fault diagnosis, IEEE Trans. Ind. Electron. 47 (5) (2000) 1060–1069.
[5] B. Muruganatham, M.A. Sanjith, B. Krishnakumar, S.S. Murty, Roller element bearing fault diagnosis using singular spectrum analysis, Mech. Syst. Signal Process. 35 (1) (2013) 150–166.
[6] R. Socher, B. Huval, B.P. Bath, C.D. Manning, A.Y. Ng, Convolutional-Recursive Deep Learning for 3D Object Classification, In NIPS, vol. 3, No. 7, 2012, December, p. 8.
[7] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[8] L. Deng, J. Li, J.T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, Recent advances in deep learning for speech research at Microsoft, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013 May, pp. 8604-8608, IEEE.
[9] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process. Magaz. 29 (6) (2012) 82–97.
[10] C. Lu, Z.Y. Wang, W.L. Qin, J. Ma, Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification, Signal Process. 130 (2017) 377–388.
[11] F. Jia, Y. Lei, J. Lin, X. Zhou, N. Lu, Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data, Mech. Syst. Signal Process. 72 (2016) 303–315.
[12] Z. Huijie, R. Ting, W. Xinqing, Z. You, F. Husheng, Fault diagnosis of hydraulic pump based on stacked autoencoders, in: 2015 12th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), vol. 1, 2015, July, pp. 58–62, IEEE.
[13] L. Guo, H. Gao, H. Huang, X. He, S. Li, Multifeatures fusion and nonlinear dimension reduction for intelligent bearing condition monitoring, Shock Vib. (2016).
[14] N.K. Verma, V.K. Gupta, M. Sharma, R.K. Sevakula, Intelligent condition based monitoring of rotating machines using sparse auto-encoders, in: 2013 IEEE Conference on Prognostics and Health Management (PHM), 2013, June, pp. 1–7, IEEE.
[15] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[16] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014 Also available at: arXiv preprint arXiv:1409.1556.
[17] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[18] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, Inception-Resnet And The Impact of Residual Connections On Learning, 2016, Also available at: arXiv preprint arXiv:1602.07261.
[19] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufier, S. Verstockt, R. Van de Walle, S. Van Hoecke, Convolutional neural network based fault detection for rotating machinery, J. Sound Vib. 377 (2016) 331–345.
[20] O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, D.J. Inman, Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks, J. Sound Vib. 388 (2017) 154–170.
[21] W. Zhang, G. Peng, C. Li, Y. Chen, Z. Zhang, A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals, Sensors 17 (2) (2017) 425.
[22] M. Gan, C. Wang, Construction of hierarchical diagnosis network based on deep learning and its application in the fault pattern recognition of rolling element bearings, Mech. Syst. Signal Process. 72 (2016) 92–104.
[23] C. Li, R.V. Sánchez, G. Zurita, M. Cerrada, D. Cabrera, Fault diagnosis for rotating machinery using vibration measurement deep statistical feature learning, Sensors 16 (6) (2016) 895.
[24] Y. Tang, Deep Learning Using Linear Support Vector Machines, 2013, Also Available at: arXiv preprint arXiv:1306.0239.
[25] F. Shen, C. Chen, R. Yan, R.X. Gao, Bearing fault diagnosis based on svd feature extraction and transfer learning classification, in: Prognostics and System Health Management Conference (PHM), 2015, 2015, October, pp. 1–6, IEEE.
[26] W. Lu, B. Liang, Y. Cheng, D. Meng, J. Yang, T. Zhang, Deep model based domain adaptation for fault diagnosis, IEEE Trans. Ind. Electron. (2016).
[27] M. Amar, I. Gondal, C. Wilson, Vibration spectrum imaging: A novel bearing fault classification approach, IEEE Trans. Ind. Electron. 62 (1) (2015) 494–502.
[28] C. Combastel, Merging kalman filtering and zonotopic state bounding for robust fault detection under noisy environment, IFAC-PapersOnLine 48 (21) (2015) 289–295.
[30] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015, Also Available at: arXiv preprint arXiv:1502.03167.
[31] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[32] N.S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, P.T.P. Tang, On Large-Batch Training For Deep Learning: Generalization Gap and Sharp Minima, 2016, Also Available at: arXiv preprint arXiv:1609.04836.
[33] Y. Li, N. Wang, J. Shi, J. Liu, X. Hou, Revisiting Batch Normalization For Practical Domain Adaptation, 2016, Also Available at: arXiv preprint arXiv:1603.04779.
[34] Z.H. Zhou, Ensemble learning, Encyclop. Biomet. (2015) 411–416.
[35] D. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, 2014, Also Available at: arXiv preprint arXiv:1412.6980.
[36] X. Lou, K.A. Loparo, Bearing fault diagnosis based on wavelet transform and fuzzy inference, Mech. Syst. Signal Process. 18 (5) (2004) 1077–1095.

Further reading

[29] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.