Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Intelligent fault diagnosis under varying working conditions based on domain adaptive convolutional neural networks

BO ZHANG[1], WEI LI[2], XIAO-LI LI[3], (Senior Member, IEEE), and SEE-KIONG NG[4]
[1]School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China (e-mail: zbcumt@163.com)
[2]School of Mechanical and Electrical Engineering, China University of Mining and Technology, Xuzhou 221116, China (e-mail: liwei_cmee@163.com)
[3]Institute for Infocomm Research (I^{2}R), A*STAR, 138632, Singapore (e-mail: xlli@i2r.a-star.edu.sg)
[4]Institute of Data Science, National University of Singapore, 117602, Singapore (e-mail: seekiong@nus.edu.sg)

Corresponding author: Wei Li (e-mail: liwei_cmee@163.com).

The work was supported by the National Key R&D Program of China (No. 2016YFC0802900), the Natural Science Foundation of Jiangsu Province (No. BK20160276), the China Postdoctoral Science Foundation (No. 2017M621862) and Jiangsu Planned Projects for Postdoctoral Research Funds (No.1701193B).

ABSTRACT Traditional intelligent fault diagnosis typically works well when the labeled training data (source domain) and unlabeled testing data (target domain) are drawn from the same distribution. However, in many real-world applications, this does not hold as the working conditions can vary between training and testing time. In this paper, a Domain Adaptive Convolutional Neural Networks called DACNN is proposed to address the issues of intelligent fault diagnosis when the data at training and test time do not come from the same distribution as a domain adaptation problem. DACNN consists of three parts: a source feature extractor, a target feature extractor, and a label classifier. In order to obtain strong fault-discriminative and domain-invariant capacity, we adopt a two-stage training process. First, to get the fault-discriminative features, the source feature extractor is pre-trained with labeled source training examples to minimize the label classifier error. Then, during the domain adaptive fine-tuning stage, the target feature extractor is trained to minimize the squared Maximum Mean Discrepancy (MMD) between the output of the source and target feature extractor, such that the instances sampled from the source and target domains have similar distributions after the mapping. Moreover, the layers between the source and target feature extractors in our DACNN are partially untied during the training stage in order to take both training efficiency and domain adaptation into consideration.
Experiments on the bearing and gearbox fault data showed that DACNN can achieve high fault diagnosis precision and recall under different working conditions, outperforming other intelligent fault diagnosis methods. We also demonstrate the ability to visualize the learned features and the networks to better understand the reasons behind the remarkable performance of our proposed model.

INDEX TERMS convolutional neural networks, domain adaptation, deep learning, intelligent fault diagnosis, transfer learning

I. INTRODUCTION

MACHINE health monitoring is of great importance in modern industry. Machine failures could cause great economic loss, and sometimes even pose threats to the people who work with the machines. There is, therefore, an unceasing need to keep the industrial machines working properly and reliably through better and more intelligent machine health monitoring technique [1], [2].

In recent years, deep learning techniques have achieved huge successes in computer vision [3], [4] and speech recognition [5], [6]. Some deep learning techniques have recently found their way into machine health monitoring systems. For example, Jia et al. took the frequency spectra generated by fast Fourier transform (FFT) as the input of a stacked autoencoder (SAE) with three hidden layers for fault diagnosis of rotary machinery components [7]. Zhu et al. proposed a SAE model for hydraulic pump fault diagnosis that used frequency features generated by Fourier transform [8]. Liu et al. used the normalized spectrum generated by the Short-time Fourier transform (STFT) of sound signals as the input of a SAE model consisting of two layers. Multi-domain statistical features including time domain features, frequency domain features, and time-frequency domain features have also been fed into the SAE model as a way of feature fusion [9], [10]. While there were some researchers focusing on deep belief networks (DBN) [11]–[13], the convolutional neural networks (CNN) [14], [15], which is popular for deep learning for image recognition, is also becoming popular for intelligent fault diagnosis of mechanical parts. For example, 1-D raw time vibration signals were used as the inputs of the CNN model for motor fault detection in [16], which successfully avoided the time-consuming feature extraction process. Guo et al. [17] proposed a hierarchical CNN consisting of two functional layers, where the first part is responsible for fault-type recognition and the other part is responsible for fault-size evaluation.

Most of the above proposed methods work well in the situation when the data used to train classifier and the data for testing are under the same working condition, in other words, under the common assumption that the labeled training data (source domain) and unlabeled testing data (target domain) are drawn from the same distribution. However, this assumption does not hold in practice. As the working condition varies in real-world applications, the labeled data obtained in one working condition may not follow the same distribution in another different working condition. When the distribution changes, most fault diagnosis models need to be rebuilt from scratch using newly recollected labeled training data. However, it is very expensive, if not impossible, to annotate huge amounts of training data in the target domain to rebuild a new model.

As one of the important research directions of transfer learning, domain adaptation (DA) typically aims at minimizing the differences between distributions of different domains in order to minimize the cross-domain prediction error by taking full advantage of information coming from both source and target domains. DA has recently been introduced into the field of fault diagnosis, such as [18]–[22]. For instance, Zhang et al. [20] took 1-D raw time vibration signal as the input of the CNN model, which realize fault diagnosis under different working loads. The domain adaptation capacity of this model originates from the method named Adaptive Batch Normalization (AdaBN). Lu et al. [19] and Wen et al. [22] separately integrated the maximum mean discrepancy (MMD) as the regularization term into the objective function of the deep neural networks (DNN) and the three-layer sparse auto-encoder (SAE) to reduce the differences between distributions cross domains. In general, the main problem for domain adaptation is the divergence of distribution between the source domain and the target domain. We need to learn a feature representation that is both fault-discriminative and simultaneously domain-invariant. The fault-discriminative ability means that the learned feature representation should minimize the label classifier error, i.e., has a good ability to identify different faults. The domain-invariant ability means that instances sampled from the source and target domains should have similar distributions in the learned feature space.

We designed a Domain Adaptive model based on CNN named DACNN, which simultaneously satisfied the above fault-discriminative and domain-invariant requirements. The details of the model will be shown in Section IV.

The main contributions of this work are summarized as follows:

The proposed DACNN consists of three parts, namely, a source feature extractor, a target feature extractor, and a label classifier. We adopt the two-stage training process to train our DACNN to ensure its strong fault-discriminative and domain-invariant capacity. In particular, unlike other existing deep domain adaptation models, we *partially untied* the layers between the source and target feature extractors in the proposed DACNN to ensure both training efficiency and domain adaptation capability.

The rest of paper is organized as follows. Section II shows the influence of the change in working conditions on the fault diagnosis model. In Section III, some preliminary knowledge that will be used in our proposed framework is briefly reviewed. Section IV introduces the construction of our proposed DACNN. A series of experiments on the classic CWRU bearing fault data and the 2009 PHM gearbox fault data are conducted in Section V and Section VI respectively. Finally, we conclude this paper in Section VII.

II. THE INFLUENCE OF THE CHANGE IN WORKING CONDITIONS

Most of the fault diagnosis models work well only under a common assumption: The training and test data are drawn from the same distribution. However, in real-world applications of fault diagnosis, the working conditions (e.g. motor load and speed) may change from time to time according to the requirements of the production. The distributions of data collected from different working conditions are similar but nonetheless different. For example, the training samples for building the classifier may be collected from the work condition without motor load, while the resulting classifier is used to classify the defects of a bearing system under different motor load states. The target data distributions with various motor load states will be different from the source data distribution for training the fault diagnostic classifier, even though the categories of defects to be detected have remained unchanged.

In order to show the influence of the change in working conditions on the fault diagnosis model, the motor bearing signals provided by Case Western Reserve University (CWRU) [23] and the state-of-the-art two-stage intelligent fault diagnosis method proposed by Lei [24] are analyzed in this section.

The vibration signals were collected from the drive end of a motor in the test rig under four different conditions: 1) normal condition; 2) inner-race faults (IF); 3) outer-race faults (OF); and 4) ball fault (BF). For IF, OF, and BF cases, vibration signals for three different severity levels (0.007 inches, 0.014 inches, and 0.021 inches) were separately collected. The sampling frequency was 12 kHz and the signals were all collected under four working conditions with different motor load and rotating speed, i.e., Load0 = 0hp/1797rpm, Load1 = 1hp/1772rpm, Load2 = 2hp/1750rpm and Load3 = 3hp/1730rpm. Further details regarding the data description can be found in section V.

According to the description given by Lei [24], there are 100 samples for each health condition under one load, where each sample is a vibration signal of bearings containing 1200 data points. Therefore, the motor bearing dataset totally contains 4000 samples, namely D_{All}, which consists of four working conditions.

In the first learning stage, local discriminative features are extracted from raw vibration signals from D_{All} by whitening, sparse filtering and local feature averaging. PCA is implemented on the learned features, and their first three PCs are shown in Figure 1. As illustrated in Figure 1(a), without considering the working conditions of the samples, most samples of the same health condition are gathered in the corresponding cluster and most samples of the different health conditions are separated. However, in order to show the influence of the change in working conditions on the distributions of the data, we mark the samples collected from working condition Load3 and Load1 separately in Figure 1(b). From Figure 1(b), we can find that the change in working conditions obviously affected the distributions of the samples of partial health conditions, such as 0.007/IF (Inner-race Faults with severity level of 0.007 inches), 0.007/BF, 0.007/OF, 0.014/BF and 0.021/IF.

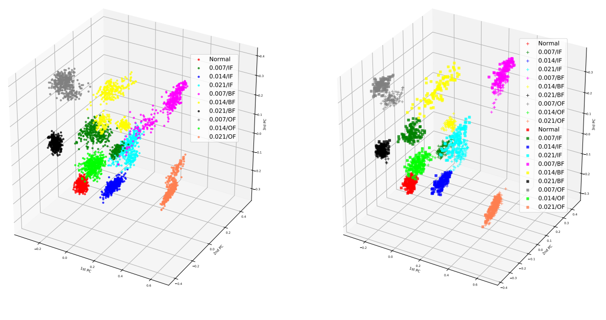


FIGURE 1: Scatter plots of PCs for the learned features. Ten different kinds of faults are denoted by ten different colors respectively. (a) the motor bearing dataset D_{All} consists of four working conditions. (b) the motor bearing dataset consists of two working conditions, where cross symbols represent the working condition Load3 and square symbols represent the working condition Load1.

In the second stage, the soft-max regression model is applied by Lei [24] to classify mechanical health conditions using the learned features. We implement the soft-max regression model using the Tensorflow toolbox of Google and keep all the parameter settings the same as Lei [24] did. Similarly, in order to demonstrate the effect of the change in working conditions on the fault diagnosis accuracy, two models are trained separately. The first soft-max regression model is trained by 10% of samples randomly selected from D_{ALL}, and the testing accuracy is 99.6% with a small standard deviation. The accuracy's and loss's trends are visualized in Figure 2. Then, 10% of samples collected from the working condition Load3 are randomly selected to train the second soft-max regression model and tested by all of the samples collected from the working condition Load1. Figure 3 shows the corresponding accuracy's and loss's trends. The validating accuracy is 99%, but the testing accuracy is merely 78.02%.

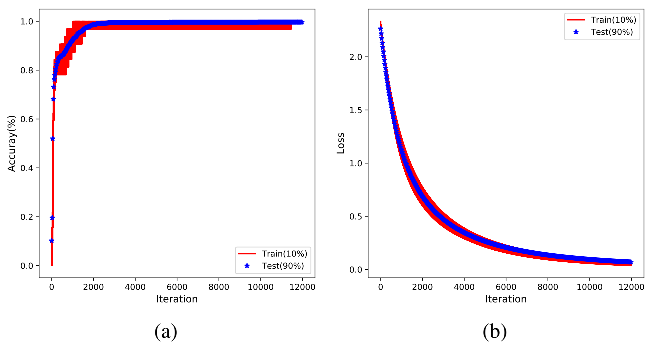


FIGURE 2: The first soft-max regression model trained by 10% of samples randomly selected from D_{ALL}. (a) The training accuracy's and test accuracy's trends. (b) The training loss's and test loss's trends.

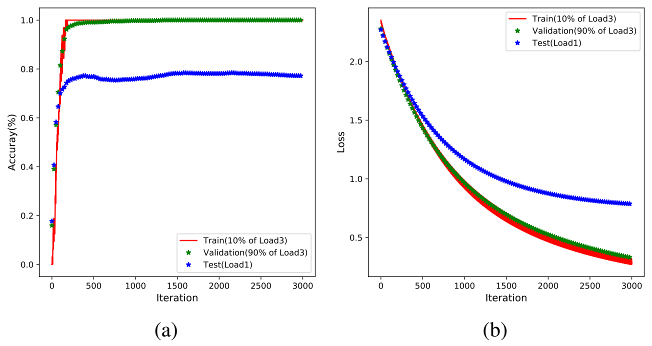


FIGURE 3: The second soft-max regression model trained by randomly selecting 10% of samples collected from the working condition Load3 and tested by all of the samples collected from the working condition Load1. (a) The training accuracy's, validating accuracy's and test accuracy's trends. (b) The training loss's, validating loss's and and test loss's trends.

From the above-mentioned experimental results, we can conclude that the change in working conditions does change the distribution of the data and affect the accuracy of the fault diagnosis model severely. Therefore, the goal of this paper is to be able to predict labels given a sample from one working condition correctly while the classifier is trained by the samples collected from another working condition.

III. PRELIMINARY KNOWLEDGE

A. DOMAIN ADAPTATION

The above practical problem of varying working conditions for fault diagnosis can be regarded as a domain adaptation (DA) problem. According to the survey on domain adaptation for classification [25], a domain $\mathcal{D}$ consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution P_X, where $X \in \mathcal{X}$. Given a specific domain, a task $\mathcal{T}$ consists of two components: a label space $\mathcal{Y}$ and a prediction function $f(X)$. Given a source domain $\mathcal{D}_S$ and a corresponding learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and a corresponding learning task $\mathcal{T}_T$, domain adaptation aims to improve the learning of the target predictive function f_T in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$, i.e., the tasks are the same but the domains are different.

The problem of fault diagnosis under varying working conditions can be framed as a domain adaptation problem by regarding samples collected from different working conditions as different domains:

- The feature spaces between domains are the same, $\mathcal{X}^S = \mathcal{X}^T$ (e.g. the fast Fourier transform (FFT) spectrum amplitudes of raw vibration temporal signals) , but the marginal probability distributions of the input data are different, $P_X^S \neq P_X^T$.
- The label spaces between domains are the same, $\mathcal{Y}^S = \mathcal{Y}^T = \{1, ..., K\}$, where K is the number of fault types.

B. DOMAIN DIVERGENCE MEASURE

The main problem in domain adaptation is the divergence of distribution between the target domain and source domain. Ben-David et al. [26], [27] defines a divergence measure $d_{H\Delta H}(P_X^S, P_X^T)$ between two domains $\mathcal{D}_S$ and $\mathcal{D}_T$, which is widely used in the theory of nonconservative domain adaptation. Using this notion, they established a probabilistic bound on the performance $\epsilon_T(h)$ of some label classifier h evaluated on the target domain $\mathcal{D}_T$ given its performance $\epsilon_S(h)$ on the source domain $\mathcal{D}_S$. Formally,

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} d_{H\Delta H}(P_X^S, P_X^T) + \lambda, \tag{1}$$

where λ is supposed to be a negligible term and dose not depend on classifier h.

Eq. 1 tells us that to adapt well, one has to learn a label classifier h which works well on source domain while reducing the $d_{H\Delta H}(P_X^S, P_X^T)$ divergence between $\mathcal{D}_S$ and $\mathcal{D}_T$.

C. MAXIMUM MEAN DISCREPANCY

Many criteria can be used to estimate the divergence of distribution between different domains, such as Kullback-Leibler (K-L) divergence and Maximum Mean Discrepancy (MMD) [28]. Different from K-L divergence, which needs an intermediate density estimation, MMD is a nonparametric estimation criterion for comparing distributions of data sets based on Reproducing Kernel Hilbert Space (RKHS).

Given observations $X_S = \{x_S^i\}|_{i=1}^{N_S}$ and $X_T = \{x_T^i\}|_{i=1}^{N_T}$, drawn independently and identically distributed (i.i.d.) from P_X^S and P_X^T respectively, the empirical estimate of the squared MMD in a reproducing kernel Hilbert space $\mathcal{H}$ with associated continuous kernel $k(\cdot,\cdot)$ can be formulated as follows:

$$\begin{aligned} MMD^2(X_S, X_T) = & \frac{1}{{N_S}^2} \sum_{i,j=1}^{N_S} k(x_i^S, x_j^S) \\ & - \frac{2}{N_S N_T} \sum_{i=1}^{N_S} \sum_{j=1}^{N_T} k(x_i^S, x_j^T) \\ & + \frac{1}{{N_T}^2} \sum_{i,j=1}^{N_T} k(x_i^T, x_j^T). \end{aligned} \tag{2}$$

$MMD(X_S, X_T)$ vanishes if and only in $P_X^S = P_X^T$, when $N_S, N_T \to \infty$.

IV. PROPOSED DOMAIN ADAPTIVE CNN

A. PROBLEM FORMALIZATION

Let the labeled source domain data be $D_S = \{(x_S^i, y_S^i)\}|_{i=1}^{N_S}$, where $x_S^i \in \mathcal{R}^{m\times 1}$ is the data instance, $y_S^i \in \{1, ..., K\}$ is its corresponding class label, and $D_T = \{(x_T^i)\}|_{i=1}^{N_T}$ is the unlabeled target domain data. Here, N_S and N_T are the numbers of instances in D_S and D_T.

The overall framework of the proposed Domain Adaptive CNN (DACNN) is shown in Figure 4. It includes a source feature extractor M_S, a target feature extractor M_T and a label classifier $\mathcal{C}$, which together form a deep feed-forward architecture that maps each input sample x_S^i (resp. x_T^i) to a K-dimensional feature vector $M_S(x_S^i)$ (resp. $M_T(x_T^i)$) (K equals to the number of class label) and predicts its class label $y \in \{1, ..., K\}$.

We use the following two-stage training procedure to enhance the domain adaptation ability of our model:

1) **Pre-training** the source feature extractor M_S to minimizes the label prediction loss $\epsilon_S(\mathcal{C})$ for the labeled source domain samples.
2) Initialize the parameters of the target feature extractor M_T with the pre-trained source feature extractor M_S, then perform **domain adaptive fine-tuning** to obtain fault-discriminative and simultaneously domain-invariant feature extractors by partially untying higher layers between M_S and M_T.

During **testing**, unlabeled target examples are mapped with the target feature extractor M_T to the latent feature space and classified by the source label classifier $\mathcal{C}$.

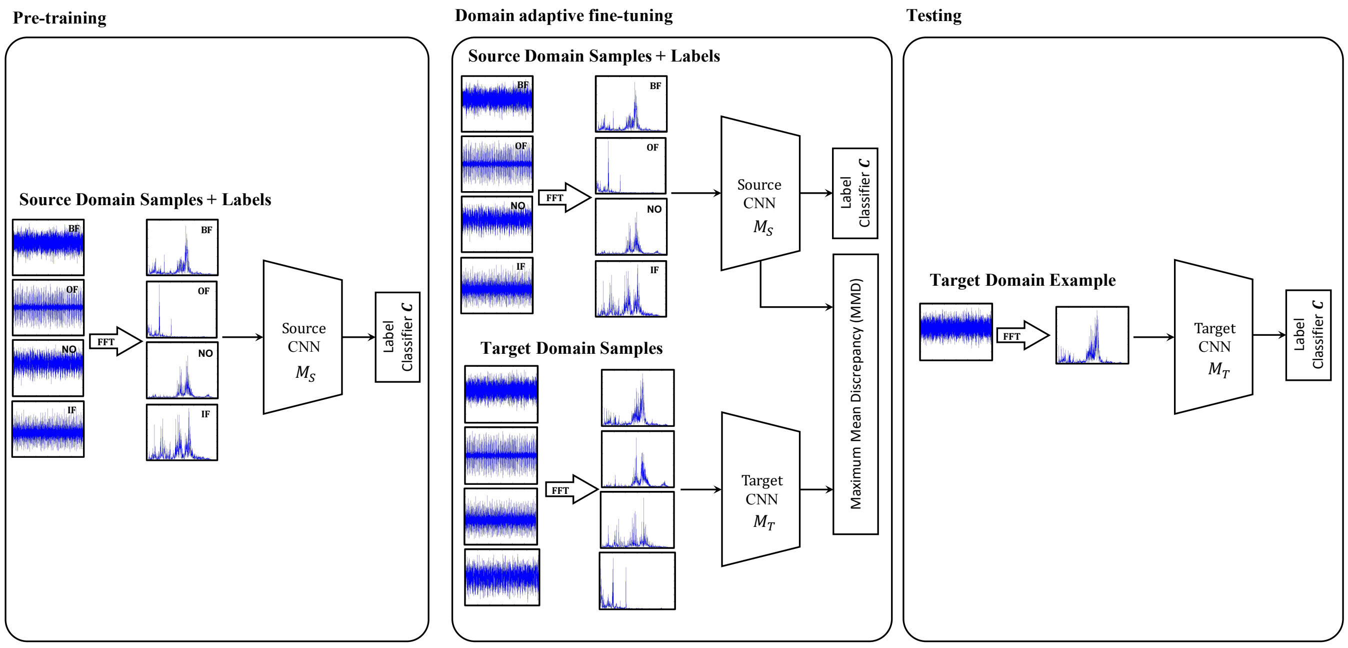


FIGURE 4: The proposed Domain Adaptive CNN (DACNN) includes a source feature extractor M_S, a target feature extractor M_T and a label classifier $\mathcal{C}$.

Our work is primarily motivated by the probabilistic bound in Eq. 1 proposed by Ben-David et al. in 2010. Based on this bound, to adapt well, one has to learn a label classifier which works well on source domain while reducing the divergence of distribution between domains. This inspired the current proposed two-stage training procedure. Firstly, the **pre-training** stage corresponds to learning a label classifier $\mathcal{C}$ which works well on the source domain using the labeled source domain samples. Then, the **domain adaptive fine-tuning** stage corresponds to making a trade-off between reducing the divergence between domains and maintaining classifier's performance on the labeled source domain data. The divergence is measured by the squared MMD between domains in our proposal.

Moreover, similar to our model, the Deep Adaptation Networks (DAN) [29] applies the multiple kernel variant of MMD (MK-MMD) for the purpose of minimizing the difference between the source and target feature distributions under the higher layers of CNN. Different from our two-stage training procedure, the DAN chooses to simultaneously fine-tune the CNN on the source labeled examples and update the higher fully connected layers for the source and target examples respectively by adding an MK-MMD-based multi-layer adaptation regularizer to the CNN risk. However, we choose to separately train the feature extractor M_S and M_T in the two-stage procedure. Because the target domain has no labeled samples, and the feature extractor M_T may quickly learn a degenerate solution if we don't initialize those higher untied layers in M_T by the parameters of the pre-trained source feature extractor M_S. Actually, the DAN also faces the problem of how to initialize the parameters of the higher fully connected layers for the unlabeled target domain. Unlike our model, they choose to start with an AlexNet model pre-trained on ImageNet 2012.

We are now ready to present each step of the proposed framework.

B. PRE-TRAINING

As shown in Figure 5, we compose the **source feature extractor** M_S from multiple convolutional layers and fully-connected layers. The input of the first convolution layer can be the fast Fourier transform (FFT) spectrum amplitudes of vibration signals, which is the most widely used approach of fault detection. The last fully-connected layer is called label layer [30] with an output of K neurons (equals to the number of the class label), which is fed to label classifier $\mathcal{C}$ to estimate the posterior probability of each class. It is common to add a pooling layer after each convolution layer in the CNN architecture separately. It functions as a down-sampling operation which results in a reduced-resolution output feature map, which is robust to small variations in the location of features in the previous layer. The most commonly used pooling layer is the max-pooling layer, which performs the local max operation over the input features. In order to capture the useful information in the intermediate and low-frequency bands, the wide kernels should be used in the first convolutional layer which can better suppress high-frequency noise [20]. The following convolutional kernels become gradually smaller which make the networks deeper to acquire good representations of the input signals and improve the performance of the networks.

For an source domain instance x_S^i, the output feature

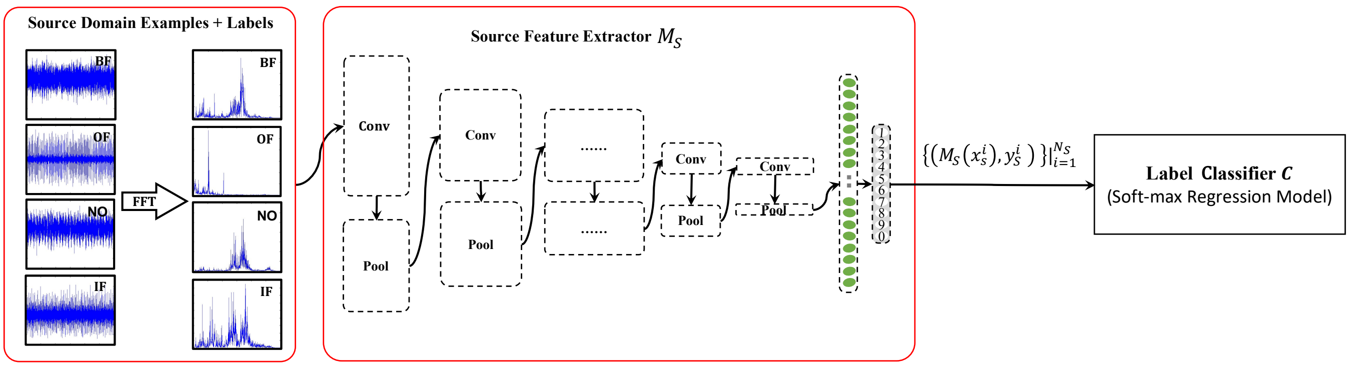

Source Domain Examples + Labels
BF
OF
NO
IF
FFT
Source Feature Extractor M_S
Conv
Pool
......
$\{(M_S(x_S^i), y_S^i)\}|_{i=1}^{N_S}$
Label Classifier C
(Soft-max Regression Model)

FIGURE 5: First training step - Pre-training.

vector $M_S(x_S^i) \in \mathcal{R}^{K\times 1}$ mapped by the source feature extractor M_S is the input of the **label classifier** $\mathcal{C}$. Here, the soft-max regression model [31] is used as the label classifier on the source domain to incorporate label information. The soft-max regression model is a generalization of the logistic regression model for multi-class classification problems. We can estimate the probabilities of each class that x_S^i belongs to as follows,

$$\mathcal{C}\left(M_S(x_S^i)\right) = \begin{bmatrix} p\left(y=1|x_S^i\right) \\ p\left(y=2|x_S^i\right) \\ \vdots \\ p\left(y=K|x_S^i\right) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} e^{u_j}} \begin{bmatrix} e^{u_1} \\ e^{u_2} \\ \vdots \\ e^{u_K} \end{bmatrix}, \tag{3}$$

where $u_j = M_S(x_S^i)_j$ is the j-th value of $M_S(x_S^i)$, $\sum_{j=1}^{K} e^{u_j}$ is a normalized term, and $p\left(y=j|x_S^i\right)$ represent the distribution of the class $j \in \{1,2,...,K\}$ given the instance x_S^i.

The details of the pre-training step are shown in Algorithm 1. Give the source domain data D_S, the parameters of the source feature extractor M_S can be derived by minimizing the following cost function,

$$L_{cls}(D_S) = \frac{1}{N_S}\sum_{i=1}^{N_S}\sum_{j=1}^{K} 1\{y_S^i = j\}\left[log\mathcal{C}\left(M_S(x_S^i)\right)_j\right], \tag{4}$$

where $1\{y_S^i = j\}$ is an indicator function, whose value is 1 if $y_S^i = j$, otherwise 0.

C. DOMAIN ADAPTIVE FINE-TUNING

Given that the target domain is unlabeled, we first initialize the parameters of the **target feature extractor** M_T with the pre-trained **source feature extractor** M_S.

As shown in Figure 6, we choose to learn the parameters of the target feature extractor M_T by *partially untying* higher layers between the source and target mappings, which is based on the following two reasons.

Firstly, the approaches of most published domain adaptation models can be summarized into two categories: symmetric transformation and asymmetric transformation. For many prior symmetric transformation methods [32], [33],

Algorithm 1: Pre-training

```
Function Pretrain():
    Data: Given one source domain
          $D_S = \{(x_S^i, y_S^i)\}|_{i=1}^{N_S}$.
    Result: The parameters in the source feature
            extractor $M_S$.
    begin
        for number of training iterations do
            Sample minibatch of $m$ instances
              $\{(x_S^1, y_S^1), ..., (x_S^m, y_S^m)\}$ from the source
              domain $D_S$;
            Update the domain discriminator $M_S$ by
              ascending its stochastic gradient;
              $\nabla_{M_S} \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{K} 1\{y_S^i =$
              $j\} \left[log\mathcal{C}\left(M_S(x_S^i)\right)_j\right]$.
        end
    end
end
```

all layers are constrained, thus enforcing exact source and target mapping consistency. Although learning a symmetric transformation can reduce the number of parameters in the model, this may make the optimization poorly conditioned, since the same networks must handle samples from two separate domains [34]. The intuitive idea behind the asymmetric transformation is to constrain a subset of the layers. Rozantsev et al. [35] showed that partially shared weights can lead to effective adaptation in both supervised and unsupervised settings.

Second, in the standard CNN, deep features must eventually transition from general to specific by the last layer of the networks, and the transferability gap grows with the domain discrepancy and becomes particularly large when transferring the higher layers.

For an instance x_S^i (resp. x_T^i), the output feature vector $M_S(x_S^i)$ (resp. $M_T(x_T^i)$) mapped by the feature extractor M_S (resp. M_T) is denoted by ξ_S^i (resp. ξ_T^i). Since there are no labeled data in the target domain, we propose to learn the divergence between domains by the squared MMD distance

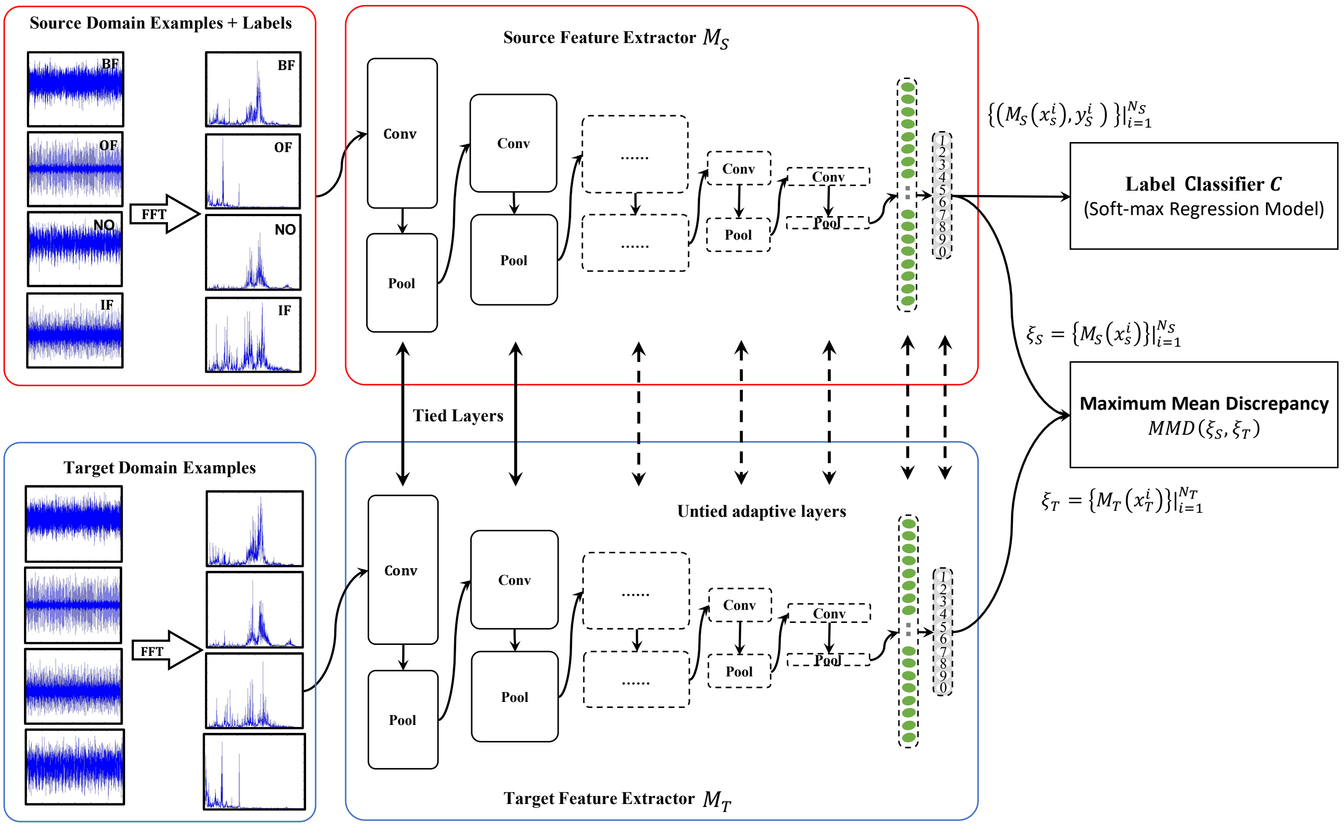

Source Domain Examples + Labels
BF
OF
NO
IF
FFT
Source Feature Extractor M_S
Conv
Pool
......
$\{(M_S(x_S^i), y_S^i)\}|_{i=1}^{N_S}$
Label Classifier C
(Soft-max Regression Model)
$\xi_S = \{M_S(x_S^i)\}|_{i=1}^{N_S}$
Maximum Mean Discrepancy
$MMD(\xi_S, \xi_T)$
$\xi_T = \{M_T(x_T^i)\}|_{i=1}^{N_T}$
Tied Layers
Untied adaptive layers
Target Domain Examples
Target Feature Extractor M_T

FIGURE 6: Second training step - Domain adaptive fine-tuning. Solid lines indicate tied layers and dashed lines indicate untying adaptive layers.

between their marginal distributions $P_{M_S(x_S)}$ and $P_{M_T(x_T)}$. So, a good trade-off between reducing the divergence between domains and maintaining classifier's performance on the labeled source domain data can be achieved by adding the following squared MMD based domain adaptation regularizer (2) to the cost function of **label classifier** C (4),

$$\min_{M_S, M_T} L_{cls} + \lambda MMD^2(\xi_S, \xi_T), \tag{5}$$

where $\lambda > 0$ is a trade-off parameter. We then assume that such M_S and M_T satisfy $P_{Y|M_S(x_S)} \approx P_{Y|M_T(x_T)}$ [36]. The details are shown in Algorithm 2.

On the other hand, the output of the last fully-connected layer could be regarded as a type of conditional distribution over the classes, and the output of other layers could be taken as the marginal distribution of features. Although the MMD matching is only done on the last layer of the networks in our current model, all the parameters of the untied layers of the target feature extractor M_T are updated during the process of back-propagation in the domain adaptive fine-tuning stage. Therefore, in the process of parameter updating, the domain discrepancy underlying both the marginal distribution and the conditional distribution could be essentially reduced.

D. TESTING

After all the parameters are learned, we can construct a classifier for the target domain by directly using the output of the last fully connected layer of the target feature extractor M_T. As shown in Figure 7, for any instance x_T^i in the target domain, the output of the target feature extractor $M_T(x_T^i)$ can compute the probability of instance x_T^i belonging to the label $j \in \{1, ..., K\}$ using Eq. 3. We choose the maximum probability using Eq. 6, and the corresponding label j as the prediction,

$$y_T^i = \max_j \frac{e^{u_j}}{\sum_{m=1}^K e^{u_m}}, \; with \; u_j = M_T(x_T^i)_j. \tag{6}$$

V. CASE STUDY 1: FAULT DIAGNOSIS OF ROLLING BEARING USING THE PROPOSED METHOD

Rolling bearings are the most commonly used components in rotating machinery, and bearing faults may result in significant breakdowns, and even casualties [37], [38]. However, learning an effective fault diagnostic model is challenging as the training vibration signals used for bearing fault diagnosis might be collected under the work condition without the motor load, while the actual application is to classify the defects from a bearing system under different motor load states. As the target data distribution changes with varying motor loads, the machine learning model must be able to use unlabeled

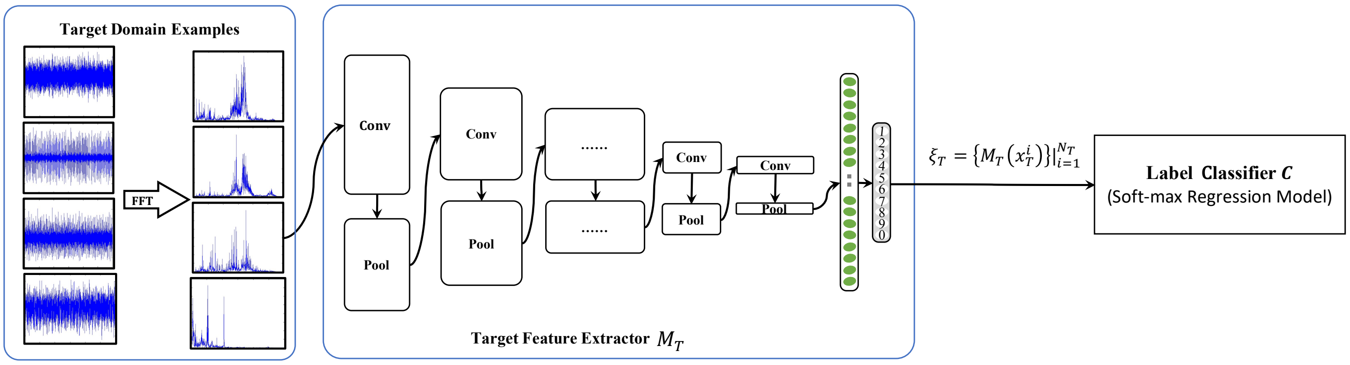


FIGURE 7: Third training step - Testing.

Algorithm 2: Domain adaptive fine-tuning

```
,
Function Finetune():
    Data:
    Given one source domain $D_S = \{(x_S^i, y_S^i)\}|_{i=1}^{N_S}$, and
    one target domain $D_T = \{(x_T^i)\}|_{i=1}^{N_T}$.
    The parameters of the pre-trained source feature
    extractor $M_S$.
    The number of adaptive layers, $l$.
    The number of trade-off parameter, $\lambda$.
    Result: The parameters in the source feature
        extractor $M_S$. The parameters in the target
        feature extractor $M_T$.
    begin
        Initialize the parameters of the target feature
          extractor $M_T$ with the pre-trained source
          feature extractor $M_S$.
        for number of training iterations do
            Sample minibatch of $m$ instances
              $\{(x_S^1, y_S^1), ..., (x_S^m, y_S^m)\}$ from the source
              domain $D_S$.
            Sample minibatch of $m$ instances
              $\{x_T^1, ..., x_T^m\}$ from the target domain $D_T$.
            Update the final $l$ adaptive layers of the
              source feature extractor $M_S$ by ascending
              its stochastic gradient:
              $\nabla_{M_S}\left(L_{cls} + \lambda MMD^2(\xi_S, \xi_T)\right)$.
            Update the final $l$ adaptive layers of the
              target feature extractor $M_T$ by ascending
              its stochastic gradient:
              $\nabla_{M_T}\lambda MMD^2(\xi_S, \xi_T)$.
        end
    end
end
```

data under any load condition to rebuild the classifier trained with samples collected in one load condition. In this section, we demonstrate the effectiveness of the proposed DACNN method for fault detection under this scenario on the bearing fault dataset provided by Case Western Reserve University (CWRU) Bearing Data Center.

A. DATASETS AND PREPROCESSING

The basic layout of the test rig is shown in Figure 8. It consists of a 2 hp motor (left), a torque transducer/encoder (center), a dynamometer (right), and control electronics (not shown). The test bearings support the motor shaft. Further details regarding the test rig can be found at the CWRU Bearing Data Center website [23].

Drive end bearing faulty data are adopted in this study. Subjected to electro-sparking, inner-race faults (IF), outer-race faults (OF) and ball fault (BF) with different sizes (0.007 inches, 0.014 inches, and 0.021 inches) are introduced into the drive-end bearing of the motor. Outer-race faults are stationary faults, therefore placement of the fault relative to the load zone of the bearing has a direct impact on the vibration response of the motor/bearing system. In order to quantify this effect, the outer-race faults themselves are grouped into three categories according to the fault position relative to the load zone: 'centered' (fault in the 6 o'clock position), 'orthogonal'(3 o'clock) and 'opposite' (12 o'clock). The vibration signals were sampled by the accelerometers attached to the rack with magnetic bases under the sampling frequency of 12 kHz and were post-processed in a Matlab environment. The experimental scheme simulates three working conditions with different motor load and rotating speed, i.e., Load1 = 1hp/1772rpm, Load2 = 2hp/1750rpm and Load3 = 3hp/1730rpm. The vibration signals of normal bearings (NO) under each working condition are also gathered. All data files released by the CWRU Bearing Data Center are in Matlab (*.mat) format. Each data file corresponds to one kind of fault data under one working condition and is defined by a unique file ID. For example, the file '106.mat' contains the data of the fault type '0.007/IF' collected from Load1 (i.e., Domain A). The details of the chosen data and its corresponding file ID are described in Table 1.

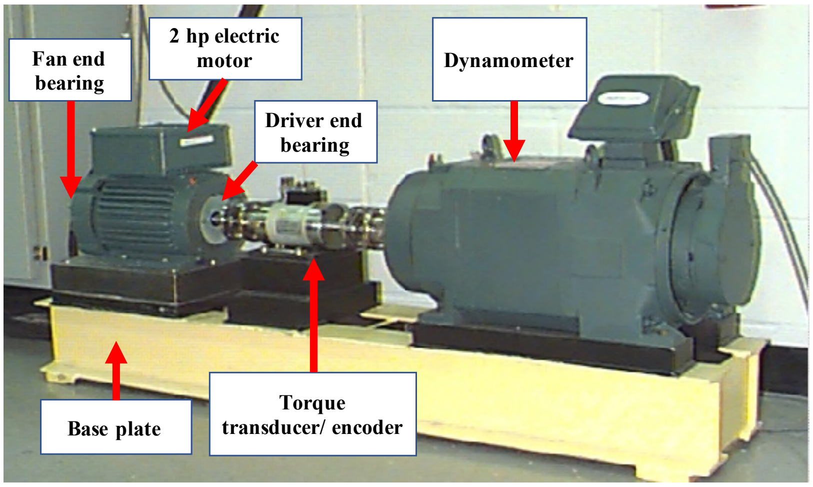


FIGURE 8: CWRU bearing test rig [23]

TABLE 1: Details of chosen data and its corresponding file ID

Category labels	1	2	3	4	5	6	7	8	9	10
Fault location	None		IF			BF			OF (Centered @6:00)	
Fault diameter (in.)		0.007	0.014	0.021	0.007	0.014	0.021	0.007	0.014	0.021
Domain						File ID				
A (Load1)	98	106	170	210	119	186	223	131	198	235
B (Load2)	99	107	171	211	120	187	224	132	199	236
C (Load3)	100	108	172	212	121	188	225	133	200	237

In this paper, a vibration signal with length 4096 is randomly selected from the raw vibration signal. Then, the fast Fourier transform (FFT) is implemented on each signal and the 4096 Fourier coefficients are generated. Since the coefficients are symmetric, the first 2048 coefficients are used in each sample. The samples collected from the above three different conditions form three domains, namely A, B, and C, respectively. There are ten classes under each working condition, including nine kinds of faults and a normal state, and each class consists of 800 samples. Therefore, each domain contains 8000 samples of ten classes collected from the corresponding working condition.

To construct domain adaptation problems, we randomly choose two from the three domains, where one is considered as the source domain and the other is considered as the target domain. Therefore, we construct six (P_3^2) domain adaptation problems. Take the domain adaptation problem $A \rightarrow B$ as an example. The examples of domain A are used as the source domain data D_S, and the examples of domain B are used as the target domain data D_T.

B. EXPERIMENTAL SETUP

1) Baseline Methods

We compare our methods with the following state-of-the-art fault diagnosis methods:

1) The deep neural networks (DNN) system with frequency features [7] proposed by Lei et al. in 2016. This neural networks consists of three hidden layers. The number of neurons in each layer is 1025, 500, 200, 100 and 10. The input of the networks is the normalized 1025 Fourier coefficients transformed from the raw temporal signals using FFT.
2) The two-stage intelligent fault diagnosis method proposed by Lei et al. in 2016. [24]. In the first learning stage, local discriminative features are extracted from raw vibration signals by whitening, sparse filtering and local feature averaging. In the second stage, the softmax regression model is applied to classify mechanical health conditions using the learned features.
3) The Deep Convolution Neural Networks with Wide first-layer kernels (WDCNN) system [20] proposed by Zhang et al. in 2017. The WDCNN system works directly on raw temporal signals. It contains five convolutional layers and batch normalization layers. The domain adaptation capacity of this model originates in the domain adaptation method named Adaptive Batch Normalization (AdaBN).

2) Parameters of the proposed DACNN

The feature extractors M_S and M_T used in our experiments are composed of five convolutional layers and pooling layers followed by two fully-connected hidden layers. The pooling type is max pooling and the activation function is ReLU.

The principles of selecting the hyperparameters of the feature extractors M_S and M_T are introduced in Section IV-B, i.e., the wide kernels should be used in the first convolutional layer which can better suppress high-frequency noise and the following convolutional kernels become gradually smaller which make the networks deeper to acquire good representations of the input signals and improve the performance of the networks. Based on these principles and extensive parameter tuning, we determine the parameters of the convolutional and pooling layers detailed in Table 2. In order to minimize the loss function, the Adam Stochastic optimization algorithm is applied to train our CNN model. The final l ($l \in [1, 7]$) layers of the target feature extractor M_T is untied and used as adaptive layers.

The experiments are implemented using the Tensorflow toolbox of Google and a sum of m multiple Gaussian kernels $\{k_i(x, x') = exp(-\frac{\|x-x'\|^2}{2\sigma_i^2})\}$ is applied as the kernel function $k(x, x')$ to train the parameters of the target feature extractor M_T, i.e., $k(x, x') = \sum_{i=1}^{m} k_i(x, x')$ [1]. Exponentially growing sequence of the parameter σ_i is used for each Gaussian kernel k_i (for example, $\sigma_i = 10^{-3}, 10^{-2}, 10^{-1}, 10, 10^1, 10^2, 10^3$).

During the pre-training stage, we randomly select 75% of the source labeled instances to train the M_S and remaining is for validation. Then, during the domain adaptive fine-tuning stage, we use $\mathcal{A}$-distance as a measure of domain discrepancy which is suggested by Ben-David et al. [27], and automatically select the parameter of l and λ by assessing the $\mathcal{A}$-distance. It involves the following steps:

1) Pseudo-labeling the output feature vectors ξ_S and ξ_T with 0 and 1.
2) Randomly sampling two sets of instances from $(\xi_s^i, 0)|_{i=1}^{N_S}$ and $(\xi_T^i, 1)|_{i=1}^{N_T}$ as the training and testing set.
3) Learning a two-sample classifier (SVM in our case) as domain classifier to distinguish the input instances between the source and target domains on the training set and verifying its performance on the testing set.
4) Estimating the $\mathcal{A}$-distance as $d_A = 2(1 - 2\epsilon)$, where ϵ is the test error.

It's obvious that if two domains perfectly overlap with each other, $\epsilon \approx 0.5$, and $d_A \approx 0$. On the contrary, if two domains are completely distinct from each other, $\epsilon \approx 0$, and $d_A \approx 2$. Therefore, $d_A \in [0, 2]$. The lower the value is the smaller two domains divergence.

To compare the effectiveness of domain adaptation, for an instance from the target domain x_T^i, we use the corresponding output of the pre-trained source feature extractor $M_S(x_T^i)$ to compute the probability of the instance x_T^i belonging to a label $j \in \{1, ..., K\}$ using Eq. 3, which is denoted as DACNN$_S$.

C. ACCURACY ACROSS DIFFERENT DOMAINS

As Table 3 shows, DNN performed poorly in domain adaptation, with average accuracies in the six scenarios being around 78.05%, which prove that samples under different working conditions draw from the different distributions and existing models trained under one working condition is not suitable for fault classification under another working load condition.

[1]https://github.com/tensorflow/models/blob/master/research/domain_adaptation/domain_separation/utils.py

Compared with the two-stage fault diagnosis method using sparse filter and WDCNN with AdaBN with average accuracy being 96.37% and 95.95% respectively, DACNN achieves the best accuracy in the average of 99.60%. This result proves that the features learned by DACNN are more domain invariant than the features learned by the other methods.

In addition, by comparing DACNN with DACNN$_S$, we can find that in every scenario, the performance of DACNN is superior to DACNN$_S$. This means that the domain adaptation fine-tuning stage can significantly improve the fault diagnosis under varying working conditions

It is also interesting that when adapting from Domain A to B, from B to A, from B to C, and from C to B, the fault diagnosis accuracy of the proposed DACNN is a bit better than WDCNN (AdaBN). However, when adapting from domain A to C and C to A, the proposed DACNN is significantly better than the other methods.

D. SENSITIVITY ANALYSIS OF FAULTS

Accuracy has been widely used as the metric to evaluate the fault diagnosis model. However, fault diagnosis is by definition an imbalanced classification problem where the positive class (machine faults) is greatly outnumbered by the negative class. The accuracy metric is therefore not an appropriate measure for assessing model performance - a classifier with a focus on merely getting all the negative instances correct will have a high accuracy by definition, but it will not be useful for identifying the few positive instances (i.e. machine faults) when it really matters. We need a metric that assesses the model's ability to find all the relevant cases in the dataset so that a good model does not miss the relevant cases. Also, in the case of fault diagnosis, the cost of false positives (e.g. halting the production line for unnecessary maintenance) can be quite high. As such, we need another useful metric that tells us out of those predicted positive, precisely how many of them are actually positive. As such, in this work we propose to employ two additional evaluation indicators, i.e. *precision* and *recall*, which have been widely used in other fields such as pattern recognition, information retrieval, and binary classification, to assess the two aspects of the model performance respectively.

In the fault diagnosis context, the *precision* and *recall* for a fault type f can be calculated as below,

$$precision(f) = \frac{TP}{TP + FP}, recall(f) = \frac{TP}{TP + FN}, \quad (7)$$

where *true positives* (TP) means the number of faults correctly identified as f, *false positives* (FP) means the number of faults incorrectly labeled as f and *false negatives* (FN) means the number of faults f incorrectly labeled as not belonging to f.

A *precision* score of 1.0 for a fault type f means that every sample labeled as belonging to class f does indeed belong to class f (i.e. there is no false alarm), but it can't tell us about

TABLE 2: Details of the feature extractor M_S and M_T used in experiments.

No.	Layer type	Layer name	Kernel	stride	Channel	Output	Padding
1	Convolution	Conv1	32×1	2×1	8	1009×8	Yes
	&MaxPool		2×1	2×1	8	504×8	No
2	Convolution	Conv2	16×1	2×1	16	245×16	Yes
	&MaxPool		2×1	2×1	16	122×16	No
3	Convolution	Conv3	8×1	2×1	32	58×32	Yes
	&MaxPool		2×1	2×1	32	29×32	No
4	Convolution	Conv4	8×1	2×1	32	11×32	Yes
	&MaxPool		2×1	2×1	32	5×32	No
5	Convolution	Conv5	3×1	2×1	64	2×64	Yes
	&MaxPool		2×1	2×1	64	1×64	No
6	Fully-connected	FC1	500		1	500	
7	Fully-connected	FC2	10		1	10	

TABLE 3: Accuracy (%) on six domain adaptation problems.

	A→B	A→C	B→A	B→C	C→A	C→B	AVG
DNN [7]	82.20%	82.60%	72.30%	77.00%	76.90%	77.30%	78.05%
Two-stage using Sparse Filter [24]	99.70%	89.62%	99.80%	99.89%	78.02%	96.37%	96.37%
WDCNN(AdaBN) [20]	99.40%	93.40%	97.50%	97.20%	88.30%	99.90%	95.95%
DACNN$_S$	99.86%	98.40%	97.89%	89.46%	89.65%	99.14%	95.73%
DACNN	**100.00%**	**99.69%**	**100.00%**	**99.90%**	**97.98%**	**100.00%**	**99.60%**

the number of samples from class f that were not labeled correctly (i.e. how many failures are missing ?).

Whereas a *recall* of 1.0 means that every item from a fault type f was labeled as belonging to class f (i.e. there is no missing alarm), but says nothing about how many other items were incorrectly also labeled as belonging to class f (i.e. how many false alarms are there ?).

The *precision* and *recall* of every class processed by DACNN and DACNN$_S$ are detailed in Table 4 and Table 5.

In Table 4, for the 3rd kinds of fault (i.e. IF with fault size being 0.014 in.), DACNN$_S$ has low *precision* when adapting from domain B to C and from C to A, which are 49.63% and 57.55% respectively. This means that about half of that kind of fault alarms are unreliable.

Meanwhile, in Table 5, for the 2nd kinds of fault (i.e. IF with fault size being 0.007 in.), DACNN$_S$ has very low *recall* when adapting from domain B to C and from C to A, which are 1.13% and 26.75% respectively. This means that about a large number of that kind of failures are not detected.

In general, the *precision* and *recall* of DACNN are higher than that of DACNN$_S$, which implies that DACNN has fewer false alarms (i.e. high *precision* score) and missed alarms (i.e. high *recall* score). Smith et al have pointed out that the ball fault cases are the most difficult to diagnose [39]. This is consistent with our experimental results. We can find that DACNN can make almost all class classified into right class, except BF with fault size being 0.014 in and BF with fault size being 0.021 in. This result shows that after the domain adaptive fine-tuning stage, the classification performance on every class achieves remarkable improvement.

E. PARAMETER SENSITIVITY

In this section, we investigate the influence of the parameter l, which represents the number of untied layers in the target feature extractor M_T during the domain adaptive fine-tuning stage.

Given that the target feature extractor M_T contains five convolutional layers and pooling layers and two fully-connected hidden layers, l is selected from $\{1, ..., 7\}$ in our experiment. We use DACNN$_l$ to denote the DACNN model with the parameter l. For example, DACNN$_1$ indicates that only the last fully-connected hidden layer is untied (i.e. FC2 in Figure 4), and DACNN$_7$ means all the seven layers in M_T are untied (i.e. from 'Conv1' to 'FC2' in Figure 4). Figure 9 reports the results.

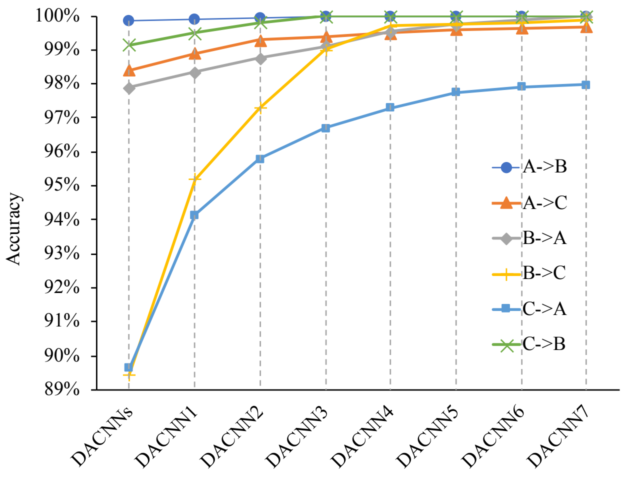


FIGURE 9: The Parameter influence of the number of untied layers l on DACNN.

According to figure, when adapting from domain B to C, untying the last fully-connected hidden layer (i.e., DACNN$_1$) amounts to only have the conditional distributions over the classes that are similar across domains, which is not enough for the domain adaptation problem of large distribution discrepancy. We have to untie more layers so that we can reduce both the marginal distribution and the conditional distribution simultaneously during the process of parameter updating to

TABLE 4: *precision* of the proposed DACNN$_S$ and DACNN on six domain adaptation problems.

Fault location	None	IF			BF			OF (Centered @6:00)		
Fault diameter (in.)		0.007	0.014	0.021	0.007	0.014	0.021	0.007	0.014	0.021
Category labels	1	2	3	4	5	6	7	8	9	10
precision of DACNN$_S$										
A→B	100%	100%	100%	100%	100%	100%	**98.64%**	100%	100%	100%
A→C	**92.27%**	100%	100%	100%	**99.75%**	**92.82%**	**99.46%**	100%	100%	100%
B→A	100%	100%	100%	100%	**87.43%**	**93.68%**	100%	100%	100%	100%
B→C	**96.74%**	100%	**49.63%**	100%	100%	**99.49%**	100%	100%	100%	100%
C→A	100%	100%	**57.55%**	100%	**83.33%**	95.40%	100%	**95.12%**	100%	100%
C→B	100%	100%	**93.13%**	**99.88%**	**98.89%**	100%	100%	100%	100%	100%
precision of DACNN										
A→B	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
A→C	96.97%	100%	100%	100%	100%	100%	100%	100%	100%	100%
B→A	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
B→C	100%	100%	99.01%	100%	100%	100%	100%	100%	100%	100%
C→A	100%	100%	100%	100%	90.91%	**90.81%**	100%	100%	100%	100%
C→B	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

TABLE 5: *recall* of the proposed DACNN$_S$ and DACNN on six domain adaptation problems.

Fault location	None	IF			BF			OF (Centered @6:00)		
Fault diameter (in.)		0.007	0.014	0.021	0.007	0.014	0.021	0.007	0.014	0.021
Category labels	1	2	3	4	5	6	7	8	9	10
recall of DACNN$_S$										
A→B	100%	100%	100%	100%	**98.63%**	100%	100%	100%	100%	100%
A→C	100%	100%	**99.75%**	100%	**99.50%**	**92.13%**	**92.88%**	100%	**99.50%**	100%
B→A	100%	100%	100%	100%	100%	100%	**78.88%**	100%	100%	100%
B→C	100%	**1.13%**	100%	100%	100%	**96.63%**	**99.50%**	100%	**97.38%**	100%
C→A	100%	**26.75%**	100%	100%	100%	**96.00%**	**73.75%**	100%	100%	100%
C→B	100%	**92.63%**	100%	100%	100%	**99.88%**	**98.88%**	100%	100%	100%
recall of DACNN										
A→B	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
A→C	100%	100%	100%	100%	100%	96.88%	100%	100%	100%	100%
B→A	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
B→C	100%	100%	100%	100%	100%	100%	100%	100%	99.00%	100%
C→A	100%	100%	100%	100%	100%	100%	79.88%	100%	100%	100%
C→B	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

reduce the distribution discrepancy across domains. We can generally observe that the more untied layers involved in the domain adaptive fine-tuning stage, the higher the accuracy of recognition.

However, the sensitivity of different adaptive problems to parameters l is different. When adapting from domain A to B, the enhancement of recognition accuracy is limited. We can use DACNN$_S$ directly to achieve the accuracy of 99.86%, which is only a little worse than DACNN$_7$. For the domain adaptation from A to C, and from C to B, we only need to untie the last three fully-connected hidden layers (i.e. DACNN$_3$) to achieve the same highest accuracy as DACNN$_7$.

Moreover, the distribution differences between domains are not symmetrical. By contrast, we have to respectively untie the last six layers (i.e. DACNN$_6$) for the domain adaptation from B to A and from C to A and untie the last five layers (i.e. DACNN$_5$) for the domain adaptation from B to C to achieve the best accuracy as DACNN$_7$.

F. NETWORKS VISUALIZATIONS

Deep learning is often viewed as an empirical success rather than a mathematical solution to the learning problem. In order to understand better why the proposed DACNN model can achieve the remarkable performance in bearing fault diagnosis under varying working conditions, the features extracted by the M_S and M_T are visualized in this subsection.

t-Distributed Stochastic Neighbor Embedding (*t-SNE*) is a technique for dimensionality reduction that is widely used for the visualization of the deep neural networks. The goal of *t-SNE* is to take a set of points in a high-dimensional space and find a faithful representation of those points in a lower-dimensional space, typically the 2D plane. In this paper, *t-SNE* is used to visualize the features extracted by DACNN. For more details about *t-SNE*, we refer to Ref. [40].

Take the domain adaptation task $B \rightarrow C$ as an example, *t-SNE* is used to visualize the high-dimensional features extracted by the source feature extractor M_S and the target feature extractor M_T. The result is shown in Figure 10. In all subgraphs of Figure 10, features of the source sample $\{x_S^i\}|_{i=1}^{N_S}$ are extracted by M_S, i.e., $M_S(x_S^i)|_{i=1}^{N_S}$, which is represented by square symbols. For the target sample, features extracted by M_S (i.e., $M_S(x_T^i)|_{i=1}^{N_T}$) are shown in (a) and features extracted by the fine-tuning M_T after 1000 and 2000 iterations are shown in (b) and (c). For convenience, these features are denoted by $M_S(x_S)$, $M_S(x_T)$ and

$M_T^{it}(x_T)$, where the number of iterations it is selected from $\{1000, 2000\}$.

There are some interesting observations as follows.

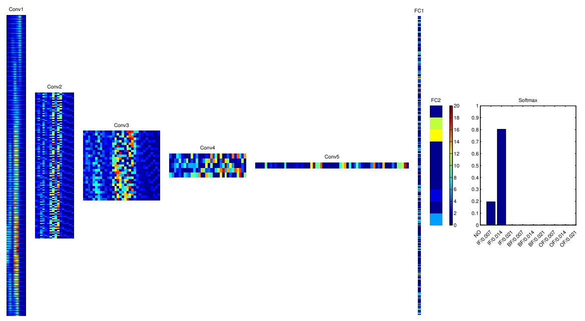
1) $M_S(x_S)$, $M_S(x_T)$ and $M_T^{it}(x_T)$ are classifiable when they are diagnosed separately. This illustrates that the CNN used for M_S and M_T has a very strong ability to distinguish various rolling bearing faults and explains the reason why DACNN$_S$ can even achieve such a good classification accuracy.
2) In Figure10(a), the distribution of fault '0.007/IF' is completely different between domains. This explains why DACNN$_S$ has the very low recall of 1.13% when adapting from domain B to C.
3) During the domain adaptive fine-tuning stage, with the increasing of iterations, the distributions of features between $M_S(x_S)$ and $M_T^{it}(x_T)$ gradually become consistent. When the features $M_T^{1000}(x_T)$ and $M_T^{2000}(x_T)$ are applied to fault detection, the accuracies are 99.75% and 99.90% respectively.

Finally, we visualize all nodes in the entire DACNN model, including M_S, M_T and the soft-max outputs of the label predictor $\mathcal{C}$. We randomly select a sample of the fault type '0.007/IF' from domain C, denoted by $x_C^{0.007/IF}$, as the input of the DACNN model trained for adapting from domain B to domain C. The visualized results are shown in Figure 11(a) and Figure 11(b).

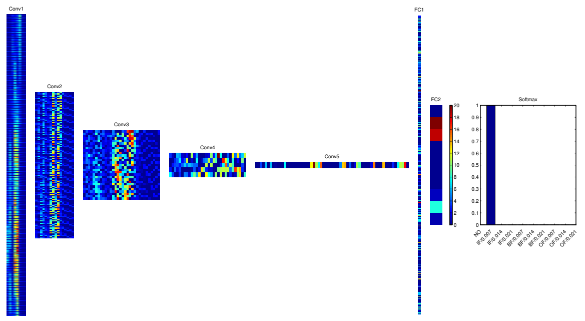
From these visual results, we can find out that the output of the first three convolutional layers (i.e. 'Conv1', 'Conv2' and 'Conv3') are very similar. Starting from the fourth layer convolution layer (i.e. 'Conv4'), the extracted features of $M_S(x_C^{0.007/IF})$ and $M_T(x_C^{0.007/IF})$ gradually change to some extent. This observation is consistent with the result in the section V-E. That is, for the domain adaptation from B to C, we only have to untie last four layers starting from 'Conv4' to 'FC2' as the features extracted from the first three convolutional layers are almost the same.

The last fully-connected layer (i.e. 'FC2') has K neurons (equals the number of the class label). The output of 'FC2' is fed to label classifier $\mathcal{C}$ to estimate the posterior probability of each class using soft-max regression model.

According to the results of 'FC2' and 'Softmax' in Figure 11(a), $x_C^{0.007/IF}$ is misdiagnosed as the fault type of '0.014/IF', based on the extracted features of $M_S(x_C^{0.007/IF})$. As a contrast, in Figure 11(b), $x_C^{0.007/IF}$ is correctly identified as the fault type of '0.007/IF', based on the extracted features of $M_T(x_C^{0.007/IF})$.



(a) $M_S(x_C^{0.007/IF})$ and the corresponding soft-max result.



(b) $M_T(x_C^{0.007/IF})$ and the corresponding soft-max result.

FIGURE 11: Visualization of all nodes in M_S, M_T and the soft-max results of the label predictor $\mathcal{C}$. Domain B is the source domain and domain C is the target domain.

VI. CASE STUDY 2: FAULT DIAGNOSIS OF GEARBOX USING THE PROPOSED METHOD

In this section, the 2009 PHM data challenge of gearboxes [41] is used to evaluate the effectiveness of the proposed method.

A. DATASETS AND PREPROCESSING

The 2009 PHM gearbox fault data are representative of generic industrial gearbox data, which contains 3 shafts, 4 gears, and 6 bearings. Two geometries are used, one using a spur gears, the other using helical gears. Data were sampled synchronously from accelerometers mounted on both the input and output shaft retaining plates. Another attached tachometer generates 10 pulses per revolution providing very accurate zero crossing information. The schematic of the gearbox used to collect the data is shown in Figure 12.

The experimental dataset is comprised of six different health conditions. The detailed description of the health conditions is shown in Table 6. For each health condition, signals were collected at 30, 35, 40, 45 and 50 Hz shaft speed under high and low load, with a sampling frequency of 66.67 kHz and acquisition time of 4 s. In this section, only the input channel of the vibration signals of the helical gearbox under low load is used to test the performance of the proposed method.

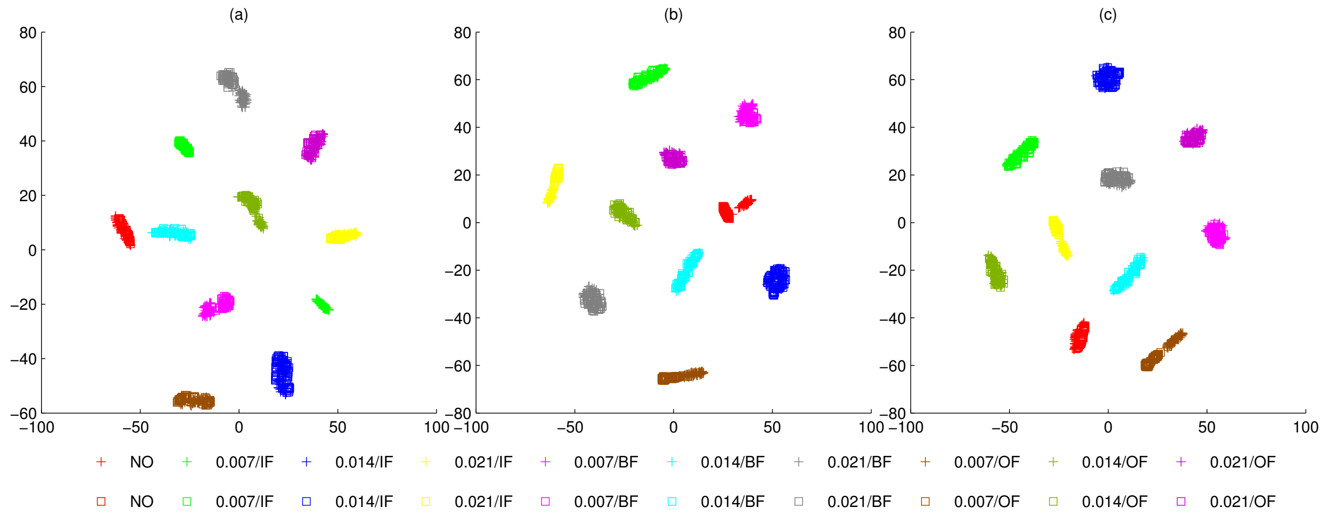
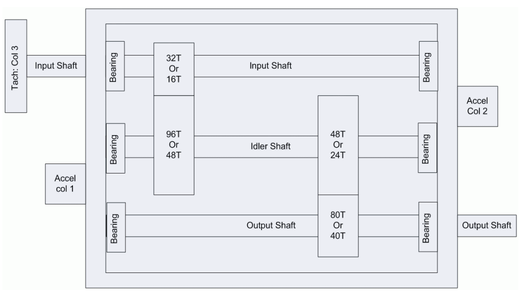


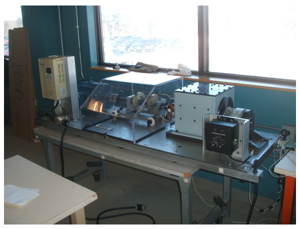
FIGURE 10: Visualization of the extracted features of samples collected from the source domain B and target domain C via *t-SNE*. Ten different kinds of faults are denoted by ten different colors respectively. Features of the source domain B extracted by M_S are represented by square symbols in all subgraphs. Features of the target domain C extracted by M_S (i.e., $M_S(x_T)$) are shown in (a) and features extracted by the fine-tuning M_T after 1000 and 2000 iterations are shown in (b) and (c).

TABLE 6: Health conditions of the 2009 PHM data challenge of gearboxes. [42]

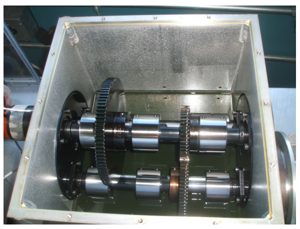
Category labels	Gear		Bearing			Shaft	
	24 T	Others	Input Shaft : Output Side	Idler Shaft : Output Side	Others	Input	Output
1	Good	Good	Good	Good	Good	Good	Good
2	Chipped	Good	Good	Good	Good	Good	Good
3	Broken	Good	Combination	Inner	Good	Bent Shaft	Good
4	Good	Good	Combination	Ball	Good	Imbalance	Good
5	Broken	Good	Good	Inner	Good	Good	Good
6	Good	Good	Good	Good	Good	Bent Shaft	Good



(a) Gearbox Schematic



(b) Gearbox Apparatus



(c) Gearbox Inside

FIGURE 12: The gearbox used in PHM 2009 Challenge Data [41].

Vibration signals are divided into data segments at first and 6144 sampling points are selected as a segment [42]. Then, the fast Fourier transform (FFT) is implemented on each data segment and the first 4097 coefficients are used in each sample. The samples collected from 30, 35, 40, 45 and 50 Hz shaft speed form five domains. There are six classes under each working condition and each class consists of 800 samples. Therefore, each domain contains 4800 samples of six classes collected from the corresponding working condition.

We randomly choose two from the five domains to construct domain adaptation problems, where one is considered as the source domain and the other is considered as the target domain. Therefore, we construct twenty (P_5^2) domain adaptation problems. Take the domain adaptation problem $30Hz \rightarrow 40Hz$ as an example. The examples collected at 30 Hz shaft speed are used as the source domain data D_S, and the examples collected at 40 Hz shaft speed are used as the target domain data D_T.

B. EXPERIMENTAL SETUP

1) Baseline Methods

We compare our methods with the convolutional neural networks (CNN) system with frequency features [42] proposed by Jing et al. in 2017. This CNN consists of one convolutional layer, one pooling layer and a fully-connected layer with softmax. Table 7 lists the parameters of the CNN. Neither overlapping of the convolutional window nor padding is used in their experiments. The input of the networks is the 4097 Fourier coefficients transformed from the data segments with length 6144 of the raw temporal signals using FFT.

According to the experimental setting in [42] , the CNN is trained by 50% of samples randomly selected from the dataset composed of 6 different health conditions of the gearbox under low load and 30, 40, 50 Hz speed. PCA is implemented on the FFT features, and their first three PCs are shown in Figure 13. As illustrated in Figure 13, without considering the working conditions of the samples, most samples of the same health condition are gathered in the corresponding cluster and most samples of the different health conditions are separated. The testing accuracy is 99.33% with a small standard deviation. The accuracy's and loss's trends of CNN [42] are visualized in Figure 14.

However, we can find that the mean and variance of the data collected at different speeds have changed significantly. Then, 50% of samples collected from the 30 Hz speed are randomly selected to train the other CNN model and tested by all of the samples collected from the 40 Hz speed. Figure 15 shows the corresponding accuracy's and loss's trends of CNN [42] . The validating accuracy is nearly 100%, but the testing accuracy is merely 27%.

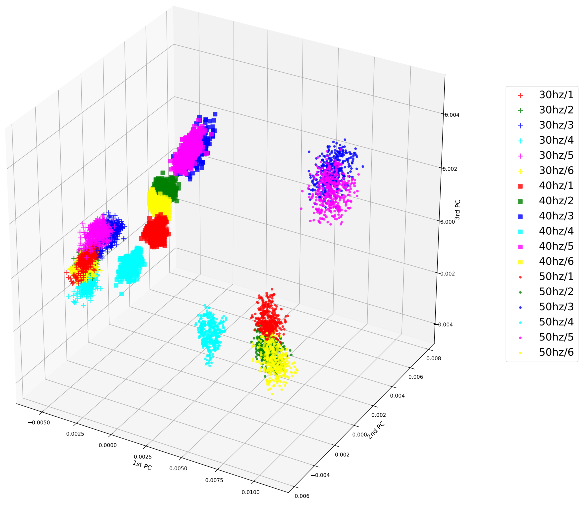


FIGURE 13: Scatter plots of PCs for the FFT features. Six different kinds of health conditions are denoted by six different colors respectively, where cross symbols represent the 30 Hz speed, square symbols represent the 40 Hz speed and dot symbols represent the 50 Hz speed.

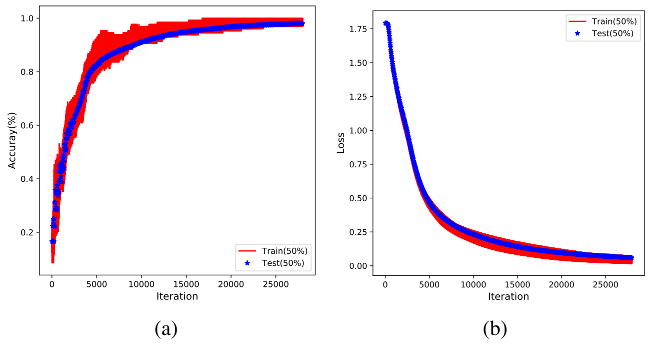


FIGURE 14: The CNN model trained by 50% of samples randomly selected from the dataset composed of 6 different health conditions of the gearbox under 30, 40, 50 Hz speed. (a) The training accuracy's and test accuracy's trends. (b) The training loss's and test loss's trends.

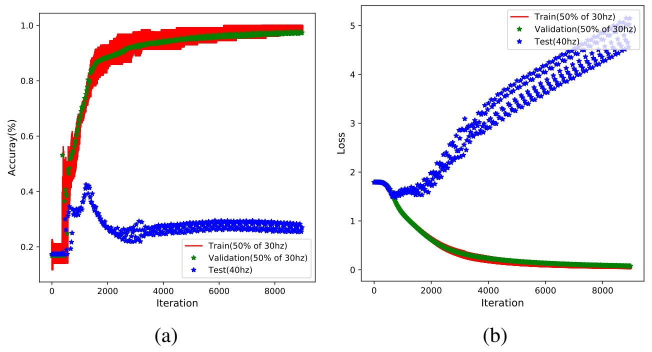


FIGURE 15: The CNN model trained by randomly selecting 50% of samples collected from the 30 Hz speed and tested by all of the samples collected from the working condition the 40 Hz speed. (a) The training accuracy's, validating accuracy's and test accuracy's trends. (b) The training loss's, validating loss's and and test loss's trends.

2) Parameters of the proposed DACNN

Similar to Case Study 1, the feature extractors M_S and M_T are composed of five convolutional layers and pooling layers followed by two fully-connected hidden layers. The pooling type is max pooling and the activation function is ReLU. The parameters of the convolutional and pooling layers are detailed in Table 8. Also, we apply a sum of m multiple Gaussian kernels $\{k_i(x, x') = exp(-\frac{\|x-x'\|^2}{2\sigma_i^2})\}$ as the kernel function $k(x, x')$ and exponentially growing sequences of the parameter σ_i is used to estimate the largest MMD distance between domains during the domain adaptive fine-tuning.

By comparing Figure 13 and Figure 1, we can find that the changes in the mean and variance of the gearbox dataset are more obvious than the bearing dataset. In order to make the examples of the gearbox dataset more or less look like standard normally distributed data and accelerate the convergence of the training of the feature extractors M_S and

TABLE 7: Parameters of the CNN used by [42].

No.	Layer type	Layer name	Kernel	stride	Channel	Output	Padding
1	Convolution	Conv1	64×1	64×1	10	64×10	No
	&MaxPool		2×1	2×1	10	32×10	No
2	Fully-connected	FC1	30		1	30	
3	Fully-connected	FC2	6		1	6	

TABLE 8: Details of the feature extractor M_S and M_T used in experiments.

No.	Layer type	Layer name	Kernel	stride	Channel	Output	Padding
1	Convolution	Conv1	32×1	2×1	8	2033×8	Yes
	&MaxPool		2×1	2×1	8	1016×8	No
2	Convolution	Conv2	16×1	2×1	16	501×16	Yes
	&MaxPool		2×1	2×1	16	250×16	No
3	Convolution	Conv3	8×1	2×1	32	122×32	Yes
	&MaxPool		2×1	2×1	32	61×32	No
4	Convolution	Conv4	8×1	2×1	32	27×32	Yes
	&MaxPool		2×1	2×1	32	13×32	No
5	Convolution	Conv5	3×1	2×1	64	6×64	Yes
	&MaxPool		2×1	2×1	64	3×64	No
6	Fully-connected	FC1	100		1	100	
7	Fully-connected	FC2	6		1	6	

M_T, the examples collected from each working condition are standardized by z-score firstly.

C. ACCURACY ACROSS DIFFERENT DOMAINS

Each subgraph of Figure 16 represents the accuracy of four domain adaptive problems when the examples collected at the corresponding shaft speed are used as the source domain data. The solid line in each subgraph represents the difference of the shaft speed Δ_{speed} between the domains. We can find that the accuracy of fault diagnosis generally decreases with the increase of Δ_{speed}.

As Table 9 shows, CNN [42] performed poorly in domain adaptation, with average accuracy in the twenty domain problems being around 33.21%. Also, when the Δ_{speed} is 5 Hz, its average accuracy in the corresponding eight domain problems is merely 50.66%. By contrast, DACNN achieves 82.62% accuracy in average in the twenty domain problems. And, its average accuracy is 91.22% in the eight domain problems with the Δ_{speed} being 5 Hz.

TABLE 9: Average Accuracy (%) on Δ_{speed}. The numbers in parentheses represent the amount of corresponding domain adaptive problems.

Δ_{speed}	5 hz (8)	10 hz (6)	15 hz (4)	20 hz (2)	AVG
CNN [42]	50.66%	26.03%	17.33%	16.67%	33.21%
DACNN$_S$	76.06%	66.29%	56.61%	38.46%	65.48%
DACNN	**91.22%**	**81.55%**	**74.29%**	**68.02%**	**82.62%**

In addition, by comparing CNN with DACNN$_S$, we can find that in every scenario, the performance of DACNN$_S$ is superior to CNN. The use of z-score in the training phase which makes the examples of the gearbox dataset look like standard normally distributed data plays an important role.

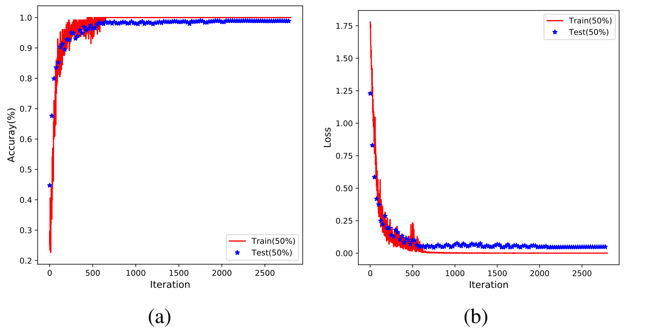


FIGURE 17: The DACNN$_S$ model trained by 50% of samples randomly selected from the dataset composed of 6 different health conditions of the gearbox under low load and 30, 40, 50 Hz speed. (a) The training accuracy's and test accuracy's trends. (b) The training loss's and test loss's trends.

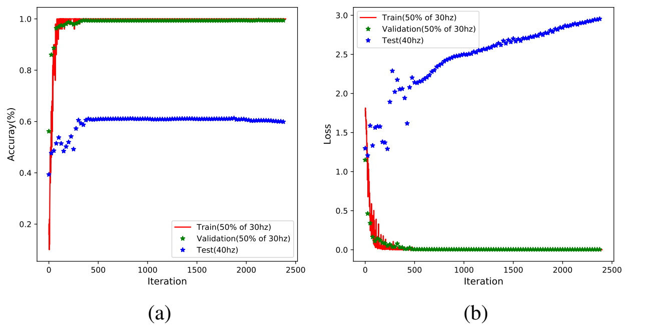
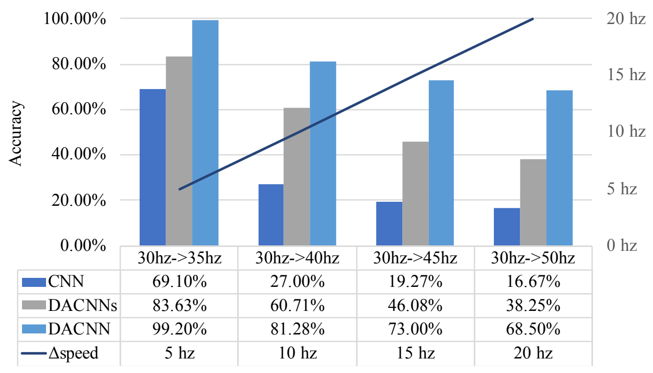
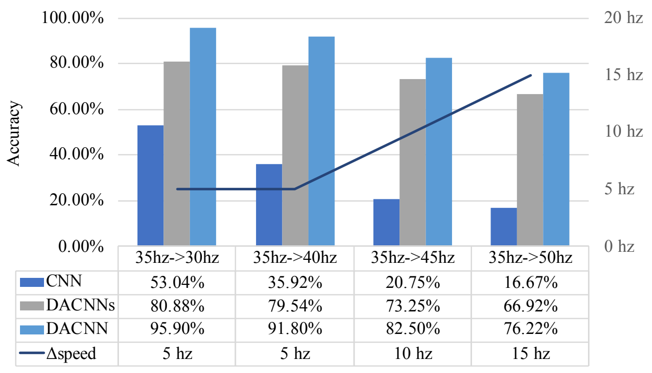


FIGURE 18: The DACNN$_S$ model trained by randomly selecting 50% of samples collected from the 30 Hz speed and tested by all of the samples collected from the working condition the 40 Hz speed. (a) The training accuracy's, validating accuracy's and test accuracy's trends. (b) The training loss's, validating loss's and and test loss's trends.



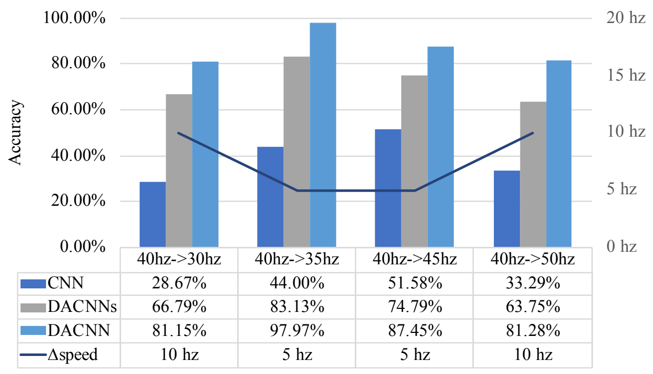
	30hz->35hz	30hz->40hz	30hz->45hz	30hz->50hz
■ CNN	69.10%	27.00%	19.27%	16.67%
■ DACNNs	83.63%	60.71%	46.08%	38.25%
■ DACNN	99.20%	81.28%	73.00%	68.50%
— Δspeed	5 hz	10 hz	15 hz	20 hz

(a) 30 Hz shaft speed



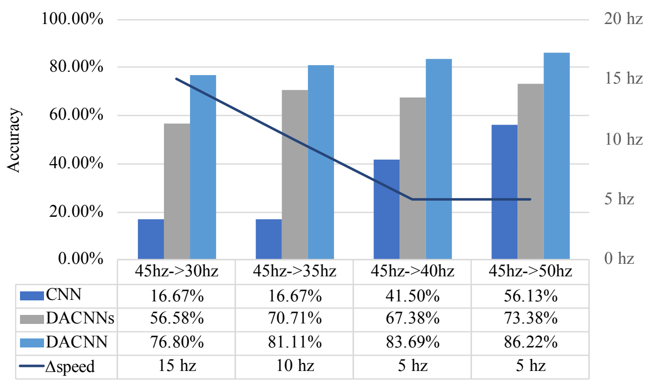
	35hz->30hz	35hz->40hz	35hz->45hz	35hz->50hz
■ CNN	53.04%	35.92%	20.75%	16.67%
■ DACNNs	80.88%	79.54%	73.25%	66.92%
■ DACNN	95.90%	91.80%	82.50%	76.22%
— Δspeed	5 hz	5 hz	10 hz	15 hz

(b) 35 Hz shaft speed



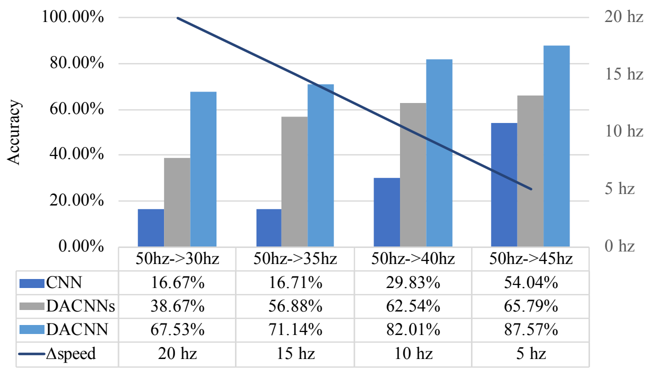
	40hz->30hz	40hz->35hz	40hz->45hz	40hz->50hz
■ CNN	28.67%	44.00%	51.58%	33.29%
■ DACNNs	66.79%	83.13%	74.79%	63.75%
■ DACNN	81.15%	97.97%	87.45%	81.28%
— Δspeed	10 hz	5 hz	5 hz	10 hz

(c) 40 Hz shaft speed



	45hz->30hz	45hz->35hz	45hz->40hz	45hz->50hz
■ CNN	16.67%	16.67%	41.50%	56.13%
■ DACNNs	56.58%	70.71%	67.38%	73.38%
■ DACNN	76.80%	81.11%	83.69%	86.22%
— Δspeed	15 hz	10 hz	5 hz	5 hz

(d) 45 Hz shaft speed



	50hz->30hz	50hz->35hz	50hz->40hz	50hz->45hz
■ CNN	16.67%	16.71%	29.83%	54.04%
■ DACNNs	38.67%	56.88%	62.54%	65.79%
■ DACNN	67.53%	71.14%	82.01%	87.57%
— Δspeed	20 hz	15 hz	10 hz	5 hz

(e) 50 Hz shaft speed

FIGURE 16: Accuracy (%) on 20 domain adaptation problems.

Similar to the experimental setting in [42] , the DACNN$_S$ is trained by 50% of samples randomly selected from the dataset composed of 6 different health conditions of the gearbox under low load and 30, 40, 50 Hz speed. The accuracy's and loss's trends of the DACNN$_S$ are visualized in Figure 17.

Then, 50% of samples collected from the 30 Hz speed are randomly selected to train the other CNN model and tested by all of the samples collected from the 40 Hz speed. Figure 18 shows the corresponding accuracy's and loss's trends of the DACNN$_S$. From Figure 17 and Figure 18, we can find that the training of the DACNN$_S$ can converge faster than CNN [42] and its accuracy's and loss's trends are more stable.

VII. CONCLUSION

This paper proposes a domain adaptive model based on CNN named DACNN to address the fault diagnosis problem under varying working condition. DACNN contains three parts, a source feature extractor, a target feature extractor, and a label classifier. Unlike other existing domain adaptation models, the layers between the source and target feature extractor in the proposed DACNN are partially untied during the training stage to ensure both training efficiency and effective domain adaptation. In order to obtain strong fault-discriminative and domain-invariant capacity, we adopt a two-stage training process. First, to get the fault-discriminative features, the source feature extractor is pre-trained with labeled source training examples to minimize the label classifier error. Then, during the domain adaptive fine-tuning stage, the target feature extractor is initialized and trained to minimize the squared MMD between the output of the source and target feature extractor, such that the instances sampled from the source and target domains have similar distributions after the mapping.

Results on the classic CWRU bearing fault data in Section V and the 2009 PHM gearbox fault data in Section VI demonstrate that, compared with the state-of-the-art intelligent fault diagnosis models, the proposed DACNN achieves higher *accuracy* under different working conditions. Besides the commonly used fault diagnostic *accuracy*, we also employed in this work two additional evaluation metrics, namely *precision* and *recall*, to analyze the sensitivity of the proposed for each type of fault detection. A *precision* score of 1.0 for a fault type means that there is no false alarm, while a *recall* of 1.0 means that there is no missing alarm. Compared with *accuracy*, *precision* and *recall* can evaluate the reliability of a model for certain type of fault recognition in more details.

Finally, through visualizing the feature maps learned by our model, we explored the inner mechanism of our proposed model in fault diagnosis and domain adaptation and verified that partially untying of the layers between the source and target feature extractor can lead to effective adaptation.

ACKNOWLEDGMENT

Thanks to Professor Ming Zhao of Xi'an Jiaotong University for providing the labeled 2009 PHM gearbox fault dataset.

REFERENCES

[1] G. J. Vachtsevanos, F. Lewis, A. Hess, and B. Wu, Intelligent fault diagnosis and prognosis for engineering systems. Wiley Online Library, 2006.

[2] Z. Qiao, Y. Lei, J. Lin, and F. Jia, "An adaptive unsaturated bistable stochastic resonance method and its application in mechanical fault diagnosis," Mechanical Systems and Signal Processing, vol. 84, pp. 731–746, 2017.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV). IEEE, 2017, pp. 2980–2988.

[4] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 4, pp. 834–848, 2018.

[5] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "The microsoft 2016 conversational speech recognition system," in Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017, pp. 5255–5259.

[6] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017, pp. 4845–4849.

[7] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data," Mechanical Systems and Signal Processing, vol. 72, pp. 303 – 315, 2016.

[8] Z. Huijie, R. Ting, W. Xinqing, Z. You, and F. Husheng, "Fault diagnosis of hydraulic pump based on stacked autoencoders," in Proceedings of the 12th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), vol. 1. IEEE, 2015, pp. 58–62.

[9] L. Guo, H. Gao, H. Huang, X. He, and S. Li, "Multifeatures fusion and nonlinear dimension reduction for intelligent bearing condition monitoring," Shock and Vibration, vol. 2016, 2016.

[10] N. K. Verma, V. K. Gupta, M. Sharma, and R. K. Sevakula, "Intelligent condition based monitoring of rotating machines using sparse autoencoders," in Proceedings of the 2013 IEEE Conference on Prognostics and Health Management (PHM). IEEE, 2013, pp. 1–7.

[11] V. T. Tran, F. AlThobiani, and A. Ball, "An approach to fault diagnosis of reciprocating compressor valves using teager–kaiser energy operator and deep belief networks," Expert Systems with Applications, vol. 41, no. 9, pp. 4113–4122, 2014.

[12] M. Gan, C. Wang, and C. Zhu, "Construction of hierarchical diagnosis network based on deep learning and its application in the fault pattern recognition of rolling element bearings," Mechanical Systems and Signal Processing, vol. 72-73, pp. 92 – 104, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327015005312

[13] J. Tao, Y. Liu, and D. Yang, "Bearing fault diagnosis based on deep belief network and multisensor information fusion," Shock and Vibration, vol. 2016, 2016.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS). Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-d convolutional neural networks," IEEE Transactions on Industrial Electronics, vol. 63, no. 11, pp. 7067–7075, 2016.

[17] X. Guo, L. Chen, and C. Shen, "Hierarchical adaptive deep convolution neural network and its application to bearing fault diagnosis," Measurement, vol. 93, pp. 490–502, 2016.

[18] F. Shen, C. Chen, R. Yan, and R. X. Gao, "Bearing fault diagnosis based on svd feature extraction and transfer learning classification," in Proceedings of the 2015 Prognostics and System Health Management Conference (PHM). IEEE, 2015, pp. 1–6.

[19] W. Lu, B. Liang, Y. Cheng, D. Meng, J. Yang, and T. Zhang, "Deep model based domain adaptation for fault diagnosis," IEEE Transactions on Industrial Electronics, vol. 64, no. 3, pp. 2296–2305, 2017.

[20] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang, "A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals," Sensors (Basel, Switzerland), vol. 17, no. 2, p. 425, 02 2017.

[21] W. Zhang, C. Li, G. Peng, Y. Chen, and Z. Zhang, "A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load," Mechanical Systems and Signal Processing, vol. 100, pp. 439–453, 2018.

[22] L. Wen, L. Gao, and X. Li, "A new deep transfer learning based on sparse auto-encoder for fault diagnosis," IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp. 1–9, 2017.

[23] "Case western reserve university bearings vibration dataset. available from: http://csegroups.case.edu/bearingdatacenter/home."

[24] Y. Lei, F. Jia, J. Lin, S. Xing, and S. X. Ding, "An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data," IEEE Transactions on Industrial Electronics, vol. 63, no. 5, pp. 3137–3147, May 2016.

[25] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, Oct 2010.

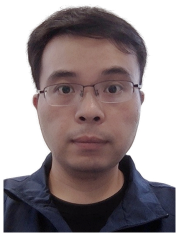
[26] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," Machine Learning, vol. 79, no. 1, pp. 151–175, May 2010.

[27] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS). MIT Press, 2006, pp. 137–144.

[28] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," Journal of Machine Learning Research, vol. 13, pp. 723–773, Mar. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2188385.2188410

[29] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in Proceedings of the 32nd International Conference on International Conference on Machine

Learning (ICML). JMLR.org, 2015, pp. 97–105. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045130
[30] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI). AAAI Press, 2015, pp. 4119–4125.
[31] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," Journal of Statistical Software, Articles, vol. 33, no. 1, pp. 1–22, 2010.
[32] Y. Ganin and V. S. Lempitsky, "Unsupervised domain adaptation by backpropagation," in Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML). JMLR.org, 2015, pp. 1180–1189.
[33] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV). IEEE Computer Society, 2015, pp. 4068–4076. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.463
[34] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2962–2971.
[35] A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2018.
[36] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," IEEE Transactions on Neural Networks, vol. 22, no. 2, pp. 199–210, Feb 2011.
[37] P. Albrecht, J. Appiarius, E. Cornell, D. Houghtaling, R. McCoy, E. Owen, and D. Sharma, "Assessment of the reliability of motors in utility applications," IEEE Transactions on Energy Conversion, vol. EC-2, no. 3, pp. 396–406, Sept 1987.
[38] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," Mechanical Systems and Signal Processing, vol. 20, no. 7, pp. 1483–1510, OCT 2006.
[39] W. A. Smith and R. B. Randall, "Rolling element bearing diagnostics using the case western reserve university data: A benchmark study," Mechanical Systems and Signal Processing, vol. 64-65, pp. 100 – 131, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327015002034
[40] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," Journal of Machine Learning Research, vol. 9, no. 2605, pp. 2579–2605, 2008.
[41] "Phm data challenge 2009. available from: https://www.phmsociety.org/competition/phm/09."
[42] L. Jing, M. Zhao, P. Li, and X. Xu, "A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox," Measurement, vol. 111, pp. 1 – 10, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0263224117304517



BO ZHANG received his PhD from the Institute of Computing Technology, the Chinese Academy of Sciences, Beijing, China, in 2014.

He is currently a lecturer at the School of Computer Science and Technology, China University of Mining and Technology. His research interests include machine learning, transfer learning, and fault diagnosis.

WEI LI received his PhD degree from University of Duisburg-Essen, Duisburg, Germany, in 2009.

Now he is a professor of mechatronic engineering at School of Mechatronic Engineering, China University of Mining and Technology, Xuzhou, P.R. China. His current research interests include fault diagnosis, remaining useful life prediction of rotating machinery.

XIAO-LI LI received his PhD from the Institute of Computing Technology, the Chinese Academy of Sciences, Beijing, China.

Now he is currently a department head (Data Analytics department, consisting of 70+ data scientists, which is Singapore largest data analytics group) and a senior scientist at the Institute for Infocomm Research, A*STAR, Singapore. He also holds adjunct associate professor positions at Nanyang Technological University. He has been serving as the (Senior) PC members/workshop chairs/session chairs in the leading data mining/machine learning related conferences (including KDD, ICDM, SDM, PKDD/ECML, ACML, PAKDD, WWW, IJCAI, AAAI, ACL, and CIKM), bioinformatics related book editors (4 book editions) and PC members.

SEE-KIONG NG obtained both his BS and PhD in computer science from Carnegie Mellon University(CMU), with an MS from University of Pennsylvania.

Now he is currently Professor of Practice at the Department of Computer Science of the School of Computing, National University of Singapore (NUS), and Director, Translational Research for the university's Institute of Data Sciences. His diverse and cross-disciplinary research interests - Bioinformatics, Smart Cities, Text Mining, Social Network Mining, and Privacy-Preserving Data Mining - revolve around advancing the science of data. His primary research objective is to unravel the underlying functional mechanisms of dynamic real-world networks and complex systems through the practice of data science.