Quick Learning Mechanism With Cross-Domain Adaptation for Intelligent Fault Diagnosis

Arun K. Sharma, *Student Member, IEEE*, and Nishchal K. Verma, *Senior Member, IEEE*

***Abstract*—The fault diagnostic model trained for a laboratory case machine fails to perform well on the industrial machines running under variable operating conditions. For every new operating condition of such machines, a new diagnostic model has to be trained which is a time-consuming and uneconomical process. Therefore, we propose a quick learning mechanism that can transform the existing diagnostic model into a new model suitable for industrial machines operating in different conditions. The proposed method uses the Net2Net transformation followed by a fine-tuning to cancel/minimize the maximum mean discrepancy between the new data and the previous one. The fine-tuning of the model requires a very less amount of labeled target samples and very few iterations of training. Therefore, the proposed method is capable of learning the new target data pattern quickly. The effectiveness of the proposed fault diagnosis method has been demonstrated on the Case Western Reserve University dataset, Intelligent Maintenance Systems bearing dataset, and Paderborn university dataset under the wide variations of the operating conditions. It has been validated that the diagnostic model trained on artificially damaged fault datasets can be used to quickly train another model for a real damage dataset.**

***Impact Statement*—The operating condition of the real-time machines in the industries may change depending on the loads or applications. For fault diagnosis of such machines, training a diagnostic model for every change in operating condition is not feasible in real-time. The quick learning mechanism proposed in this paper solves this problem by transforming the existing diagnostic model from laboratory case machines to real case machines running under any load conditions. The proposed methodology is capable to generate and train a suitable diagnostic model quickly for every new operating condition and therefore this method can be a very promising solution for the real-time monitoring of industrial case machines.**

***Index Terms*—Condition-based maintenance, domain adaptation, intelligent fault diagnosis, maximum mean discrepancy (MMD), Net2Net operation, transfer learning.**

I. Introduction

THE continuous monitoring of real-time industrial rotating machines plays a vital role in the safety and productivity of modern industries. It requires a diagnostic model for analysis of the measurement data generated by the continuous processes. For the fault diagnosis of real-time industrial machines, there are two main challenges: i) unavailability of the labeled dataset as some machines may not be allowed to run in a failure state, which may cause catastrophic accidents or critical breakdown and ii) distribution of the dataset may change due to change in the operating condition of the machine.

In literature, various data-driven approaches have been reported for intelligent health monitoring of the rotating machines, also called condition-based maintenance (CBM). The intelligent systems based on evolutionary algorithm [1]–[3], fuzzy systems [4], and deep neural networks (DNNs) [5], [6] are applied for learning the complex pattern from recorded data. It monitors specific changes in the machine signatures like vibration, acoustics, temperature, pressure, etc, [7]–[9] and notify the anomaly/fault in the various component of the machine.

Vibration-based CBM has gained much attention due to its effectiveness and flexibility in assessing the machine health conditions using time-domain and frequency-domain methods of signal analysis, [10]–[13]. But these approaches become ineffective if there is a contamination of extraneous frequencies and there is a change in the dynamic behavior of the machine. The recent advancement in machine learning techniques overcomes these problems and is capable to learn highly nonlinear and complex characteristics of a data pattern and diagnose the machine health condition [14]–[19]. These methods give accurate/commendable results for the test dataset exactly similar to the training data. If the operating condition of the machine is variable, the diagnostic model fails to perform well unless the model is retrained for the new data pattern. For such problems, knowledge transfer (also called transfer learning) gives a better solution [20]–[22], where training is accelerated by transferring the previous knowledge to the new similar task.

Pratt [20] formulated a discriminability-based transfer algorithm to introduce the concept of transfer learning. It uses previously learned knowledge to initialize the parameter of the target model to reduce the time of training. In most of the approaches of the transfer learning [21], the pretrained model on the source dataset is fine-tuned using a similar target dataset. If the distribution of the target data is different, fine-tuning the model requires a large amount of labeled target data and a long training process. For such cases, the transfer learning method fails to train the model using a small number of training samples from the target domain. To solve the problem of cross-domain fault analysis, domain adaptation (DA) techniques have been reported in various literature [23]–[29].

Manuscript received March 4, 2021; revised June 22, 2021 and August 26, 2021; accepted October 9, 2021. Date of publication November 12, 2021; date of current version May 25, 2022. This paper was recommended for publication by Associate Editor Dipti Srinivasan upon evaluation of the reviewers' comments. *(Corresponding author: Arun K. Sharma.)*

The authors are with the Department of Electrical Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, India (e-mail: arnksh@iitk.ac.in; nishchal@iitk.ac.in).

Digital Object Identifier 10.1109/TAI.2021.3123935

Pan *et al.* [23] proposed a learning mechanism for shared subspace using transfer component analysis via minimizing maximum mean discrepancy (MMD) of the source and the target features representations. Ganin *et al.* [25] proposed a domain adversarial training of DNN which uses labeled source data and unlabeled target data. Wen *et al.* [26] have suggested the minimization of combined objective function of classifier loss and the MMD term for the domain adaptation during fine-tuning of the model. Similarly, Lu *et al.* [27] use MMD term to model the loss due to change in probability distribution in a subspace and obtained a training law for DNN. These methods of transferring knowledge are not useful if the change in the data pattern needs a deep learning model with different architecture.

Chen *et al.* [30] has proposed Net2Net transformation to initialize a new (student) network from a previously trained (teacher) network based on the function preserving principle. The function-preserving principle allows us to quickly change the architecture of the network without changing the function mapping. If the student network with new architecture has to be used with a new dataset of a different probability distribution, it requires a sufficient amount of data and a large number of iterations to learn the new data pattern. Again, it becomes a challenge to quickly adapt with the change in the data pattern during the continuous process of experimentation.

To solve this problem, we propose a quick learning mechanism based on Net2Net transformation followed by fine-tuning with a domain adaptation algorithm. The process of fine-tuning minimizes the classification loss plus domain discrepancy using a few number of samples from the target domain. The key contributions of this work are summarized as follows.

1) The proposed mechanism is capable to train a DNN of user-provided architecture needed for the fault diagnosis of machines under variable working conditions.
2) The main novelty of the proposed work is that the desired architecture suitable for the industrial machine is generated from a trained DNN. The fine-tuning of the new DNN includes minimization of classification loss as well as MMD term. Therefore, the new DNN requires a less amount of target data and a few iterations of training. This makes the algorithm to learn the new pattern of the dataset faster.

The rest of this article is organized as follows. Section II briefly describes the theoretical background of DNN, domain adaptation, and function-preserving network transformation. Section III describes the development of quick learning system: the proposed approach. In Section IV, effectiveness of the proposed fault diagnosis approach has been demonstrated on i) Case Western Reserve University (CWRU) fault diagnosis bearing data [31], ii) Intelligent Maintenance Systems (IMS) bearing dataset [32], and iii) Paderborn university dataset [33]. Finally, Section V concludes the article.

II. Theoretical Background

A. Deep Neural Network

DNN is a multilayer neural network (NN) capable of highly nonlinear transformation through each layer depending on the types of activation function chosen to fit the problem. The training of the network includes obtaining optimal weight to map the input–output dataset. The DNN discussed in this section is a multilayered network called stacked autoencoder (SAE) [34], which is trained by the method of greedy layer unsupervised learning using unlabeled data followed by fine-tuning using labeled data. The formation of SAE by stacking autoencoders is depicted in Fig. 1.

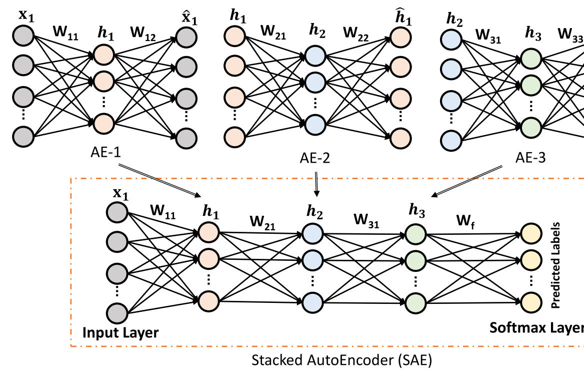

Fig. 1. Formation of stacked auto-encoder with three hidden layer.

Learning Mechanism of Autoencoder: The autoencoder is trained to learn the identity approximation [34] of the unlabeled dataset, $h_{W,b}(X) \approx X$, where W denotes the weight matrix and b is the bias vector of the hidden layer of the autoencoder. For an unlabeled dataset, the cost function J for sparse autoencoder in terms of weight W and bias b is defined as

$$J = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left\|h_{\mathbf{W},\mathbf{b}}(\mathbf{X}^l) - \mathbf{X}^l\right\|^2 + \frac{\lambda}{2}\sum_{i=1}^{d_l}\sum_{j=1}^{d_{l+1}}(W_{ji})^2 + \beta\sum_{j=1}^{d_l} KL(\rho/\hat{\rho}) \quad (1)$$

where N be the number of training samples, $\mathbf{X}^l$ be the input samples to the lth autoencoder, λ be the regularization parameter, d_l denotes the number of nodes in the lth layer, ρ be the sparsity parameter, β be the weight-penalizing deviation from ρ, and $KL(.)$ denotes the Kullback–Leibler divergence function. Sparsity is the mean of activation of a hidden layer averaged over the training set and is enforced to be equal to a given sparsity parameter. Therefore, sparsity constraint ensures the desired sparsity of the generated feature representation(s) at the hidden layer. Regularization parameter [35] is used to ensure the appropriate fitting of the DNN hyperparameters for the training dataset and avoid overfitting or underfitting for the testing on unseen data.

B. Domain Adaptation

Domain adaptation in transfer learning creates a self-taught system that learns the target data-mapping without the availability of labeled data or with partial availability of labeled datasets. Domain adaptation by minimizing MMD has gained

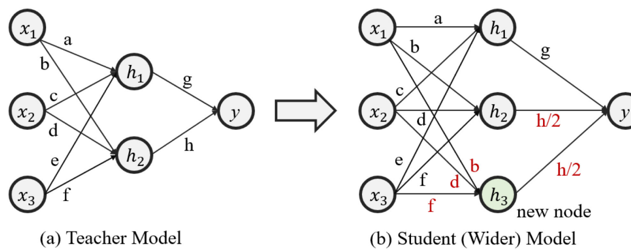

Fig. 2. Net2WiderNet transformation: h_3 be the new node.

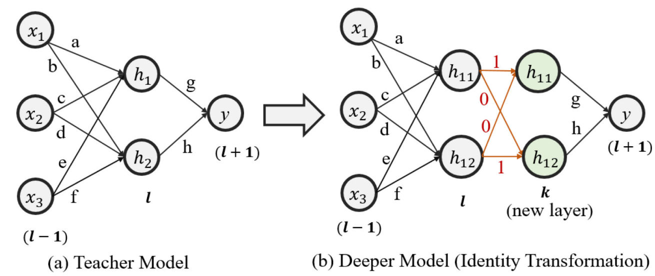

Fig. 3. Net2DeeperNet transformation: kth be the new layer.

much popularity due to its capability of domain shift in the shared subspace.

1) Maximum Mean Discrepancy (MMD): MMD gives the measure of nonparametric distance between mean of two distributions on reproducing kernel Hilbert space (RKHS) [36], [37]. For a universal RKHS, MMD asymptotically becomes zero if and only if the probability distribution of the two datasets is the same. Let $x_i^s \in \mathbf{X}^s$ and $x_j^t \in \mathbf{X}^t$ be the ith source data and the jth target data in the source space χ^s and the target space χ^t, respectively, then the MMD of the two distribution in RKHS is defined as

$$\mathrm{MMD}(\mathbf{X}^s, \mathbf{X}^t) = \left\| \frac{1}{n^s} \sum_{i=1}^{n^s} f\left(x_i^s\right) - \frac{1}{n^t} \sum_{j=1}^{n^t} f\left(x_j^t\right) \right\|_{\mathcal{H}} \quad (2)$$

where $f(.)$ denotes the kernel function $f : X, X \to \mathcal{H}, \mathcal{H}$ be the universal RKHS, and n^s and n^t be the number of samples in the source and the target datasets $\mathbf{X}^s$ and $\mathbf{X}^t$, respectively.

2) Function-Preserving Transformation: A function $y = f(x; \psi)$ represented by an NN model can equivalently be represented by another NN model $g(x; \psi')$ if a new set of parameter ψ' is chosen such that

$$\forall x, \quad f(x; \psi) = g(x; \psi') \quad (3)$$

where x is the input dataset, y is the output class label, and ψ and ψ' are the NN parameter vector. This concept allows us for two types of network transformations: Net2WiderNet and Net2DeeperNet.

3) Net2WiderNet: Net2WiderNet transformation is used to widen the network by adding nodes in any of the hidden layers of a network [30]. We consider a simple example to elaborate on the idea of replacing the network with a wider network as shown in Fig. 2.

Let the network in Fig. 2(a) be trained on a given dataset and be treated as the teacher network to train a wider network, called student network in Fig. 2(b). The knowledge of the network in Fig. 2(a) is transferred to initialize the parameters of a wider network in Fig. 2(b) such that it provides the same output as the network in Fig. 2(a). Random nodes are picked from the hidden layer and replicated as new nodes. Here, the weights of the h_2 node (b, d, and f) are replicated as the connections to the new node h_3. The outgoing weights are divided by 2 to compensate for the replication of h_2.

The above idea can easily be generalized with a recursive function for more than one layer and to add more than one node in a layer. Let us assume that a trained network has n_2 number of nodes in the lth layer and has to be widened to $n_2'(> n_2)$. To initialize the weight matrix in the new network, a random sampling function $\hbar : \{1, 2, \ldots, n_2'\} \to \{1, 2, \ldots, n_2\}$ is defined for the repetition of nodes as

$$\hbar(j) \triangleq \begin{cases} j & j \leq n_2 \\ \text{random sample from } \{1, \ldots, n_2\} & j > n_2 \end{cases} \quad (4)$$

where j is the node index from 1 to n_2' in the new network. Once random repetition of nodes is chosen, weights are assigned as demonstrated in Fig. 2.

4) Net2DeeperNet: Net2DeeperNet transformation allows us to insert a new layer to the network to transform it into a deeper network [30].

In Fig. 3, deeper model is created by inserting a layer k after the hidden layer l having the same number of nodes as in layer l. Let the output of the lth layer be $v^{(l)} = \phi(v^{(l-1)T} W^{(l)})$, where $\phi(.)$ represents the mapping by the lth layer and $W^{(l)}$ be the weight matrix in between $(l-1)$th and lth layers. The new kth layer is inserted such that its output is is given by

$$v^{(k)} = \phi\left(W^{(k)T} \phi\left(v^{(l-1)T} W^{(l)}\right)\right) \quad (5)$$

where $W^{(k)}$ is the weight matrix in between lth and kth layers and initialized to be identity matrix.

III. Quick Learning Mechanism: Proposed Methodology

We propose a network transformation method using the function-preserving principle of the NN model and minimization of MMD term from source to target dataset. The flowchart of the proposed method is depicted in Fig. 4. For the process shown in Fig. 4, $\mathrm{X}^{(\mathrm{s})}$ denotes the source dataset, $\mathrm{X}^{(\mathrm{t})}$ the target dataset, and W_{te}, b_{te} denote the weight and bias matrices of the teacher network and $\mathrm{W}_{\mathrm{stu}}$, $\mathrm{b}_{\mathrm{stu}}$ of student network. The whole learning mechanism includes two steps as described in the following subsections.

A. Step-1: Network Transformation

The knowledge transfer using the function-preserving concept allows us to transform the network to a new architecture without losing the function mapping (as described in Sections II-B3 and II-B4). Let W_{te}, b_{te} be the weight and bias matrices of the teacher network. Using Net2Net transformation, initial weight and bias $\mathrm{W}_{\mathrm{stu}}$ and $\mathrm{b}_{\mathrm{stu}}$ of student network are obtained, and then,

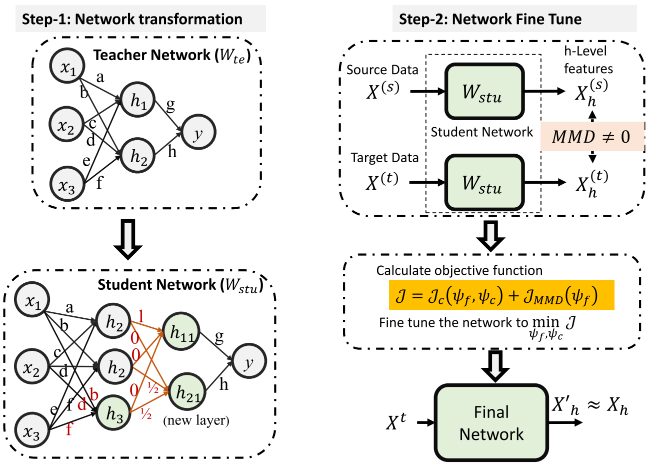

Fig. 4. Flowchart of the quick learning methodology (width and depth of the DNN shown here are for demonstration only).

Algorithm 1: Train the Teacher Network (DNN Model).

1: Train autoencoder (AE)
2: **for** i $\in$ Number of hidden layers **do**
  $AE_i \longleftarrow [h_{i-1},\ h_i,\ h_{i-1}]$ # Train the ith AE, $h_i = i$th hidden layer $i-1$ for $i=1$ indicates the input data
3: **end for**
4: DNN $\longleftarrow$ [Input, $h_1, h_2, \ldots, h_l$, Softmax layer] # Form DNN by stacking all hidden layers with input and output layers
5: Classification loss $J(W, b) \longleftarrow$ MSE [true label, predicted label] # Calculate mean-square error (MSE)
6: Fine-tune the network to find the optimal parameter (weight and bias vectors)
7: Teacher network $\longleftarrow$ Train DNN

the network weights are fine-tuned using the new (target) dataset by incorporating MMD term with the classifier loss.

B. Step-2: Fine-Tune With Domain Adaptation

The classification loss for the C class problem and the MMD term is defined using the h-level feature output of the new architecture for the source dataset and target dataset as follows:

$$\mathcal{J}_{\text{MMD}} = \sum_{i=1}^{C} \left\| \frac{1}{N_i^{(s)}} \sum_{p=1}^{N_i^{(s)}} f(x_{i,p}^{(s)}) - \frac{1}{N_i^{(t)}} \sum_{q=1}^{N_i^{(t)}} f(x_{i,q}^{(t)}) \right\|_{\mathcal{H}}^2 \tag{6}$$

$$\mathcal{J}_c = \frac{1}{N^{(s)}} \left[\sum_{p=1}^{N^s} \sum_{i=1}^{C} I[y_p = C] \log \frac{e^{(w_i^T f(x_p^s) + b_i)}}{\sum_{i=1}^{C} e^{(w_i^T f(x_p^s) + b_i)}} \right] \tag{7}$$

where $f(x) = \Phi(Wx + b)$ is the h-level features representation of DNN, and $N_i^{(s)}$ and $N_i^{(t)}$ are the number of samples in the ith of the source dataset $\mathrm{X}^{(s)}$ and the target dataset $\mathrm{X}^{(t)}$, respectively. $N^{(s)}$ be the batch size of the source data, y_p be the source label, and $[w_i,\ b_i]$ be the weight and bias connecting the ith node in the output (softmax) layer. Parameters of the student network are optimized by minimizing the loss function (8).

$$\mathcal{J} = \mathcal{J}_c + \lambda \mathcal{J}_{\text{MMD}}. \tag{8}$$

Let $\psi_f = [\mathrm{W}_{\text{stu}},\ \mathrm{b}_{\text{stu}}]$ and $\psi_c = [\mathrm{W}_{\text{c}},\ \mathrm{b}_{\text{c}}]$ be the parameters of the feature extractor (DNN) and classifier, respectively, then the loss function can be minimized to obtain the optimal network parameters by solving (9).

$$\min_{\psi_f,\, \psi_c} \mathcal{J} = \min_{\psi_f,\, \psi_c} \left[\mathcal{J}_c(\psi_f, \psi_c) + \lambda \mathcal{J}_{\text{MMD}}(\psi_f) \right]. \tag{9}$$

Algorithm 2: Algorithmic Steps for Quick Learning Mechanism: The Proposed Methodology.

Input:
  Teacher network parameters: W_{te}, b_{te}
  target datasets: $X^{(t)}$
Output:
  Fine-tuned network in the target domain.

1: **for** l $\in$ layers to be replicated **do**
  a) Add new layer k after layer l of size same as l
  b) Assign weight matrix to new layer $U^{(l)} = I_n$ (identity matrix) to satisfy (5)
2: **end for**
3: **for** l $\in$ layers to be widened **do**
  a) Let number of nodes in layers $l-1, l$, and $l+1$ be n_{l-1}, n_l, and n_{l+1} and number of nodes to be added $= n$.
  b) Generate random samples $\hbar \in [1,\ n_l]$ using equation (4)
  c) Let r_j denotes the number of repetition of j^{th} node in random samples $\hbar$
  d) Let $W_{j,i}^{(l)}$ denotes the weight connecting i^{th} node in $(l-1)^{th}$ layer to j^{th} node in l^{th} layer
  e) $W_{k,j}^{(l+1)}$ denotes the weight connecting j^{th} node in l^{th} layer to k^{th} node in $(l+1)^{th}$ layer
  f) **for** $j \in [n_l + 1,\ n_l + n]$ **do**
    for $i \in [1, n_{l-1}]$ **do**
      $\mathrm{W}_{j,i}^{(l)} = \mathrm{W}_{\hbar - n_l, i}^{(l)}$ & $\mathrm{b}_j^{(l)} = \mathrm{b}_{\hbar_{j-n_l}}^{(l)}$
    end for
    for $k \in [1, n_{l+1}]$ **do**
      Assign $\mathrm{W}_{k,j}^{(l+1)} = \mathrm{W}_{k,\hbar_{j-n_l}}^{(l+1)} / r_j$
      Update $\mathrm{W}_{k,\hbar_{j-n_l}}^{k} = \mathrm{W}_{k,\hbar_{j-n_l}}^{(l+1)} / r_j$
    end for
  g) **end for**
4: **end for**
5: Result: student network (weight matrix: W_{stu}, b_{stu}
6: Fine-tune the Network: Update the weight and bias using equations (10) and (11) to get the optima weight matrix
7: **end**

The cost function optimization objective stated in (9) is achieved by Limited-Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [38] algorithm or optimal weights can also be obtained using gradient descent as follows:

$$\psi_{\mathrm{f}} \leftarrow \psi_{\mathrm{f}} - \eta \left[\frac{\partial \mathcal{J}_c(\psi_f, \psi_c)}{\partial \psi_{\mathrm{f}}} + \lambda \frac{\partial \mathcal{J}_{MMD}(\psi_f)}{\partial \psi_{\mathrm{f}}} \right] \quad (10)$$

$$\psi_{\mathrm{c}} \leftarrow \psi_{\mathrm{c}} - \eta \frac{\partial \mathcal{J}_c(\psi_f, \psi_c)}{\partial \psi_{\mathrm{c}}} \quad (11)$$

where η is the learning rate for the parameter update.

The training process in the proposed method requires a pretrained DNN; therefore, the algorithmic steps for the training of DNN are given in Algorithm 1. The algorithmic steps for the proposed methodology is presented in Algorithm 2.

IV. Results and Discussion

Effectiveness of the proposed framework of network transformation with domain adaptation has been demonstrated using CWRU fault diagnosis bearing data [31], IMS bearing dataset [32], and Paderborn university dataset [33].

A. Dataset Description

1) CWRU Bearing Data [31]: The bearing dataset provided by CWRU was recorded on a ball-bearing testing platform. Using electrodischarge machining, motor bearings were seeded with faults of fault diameters 7, 14, and 21 mils (1 mil = 0.001 inches) at the inner raceway, rolling element (i.e., ball), and outer raceway. The vibration data were recorded with motor loads of 0–3 HP and motor speeds of 1730–1797 r/min in four different cases: i) *normal baseline data*, ii) *12k samples/s drive end (DE) fault data*, iii) *48k samples/s DE bearing fault data*, and iv) fan-end (FE) bearing fault data (recorded at 12k samples/s). The vibration signal represents four different states of the machine: i) healthy/normal (N), ii) inner race (IR), iii) outer race (OR), and iv) rolling element (ball: B).

2) IMS Bearing Dataset [32]: IMS bearing datasets [32] are provided by National Science Faundation Industry/ University Cooperative Research Center (NSFI/UCRC) of IMS. It consists of three datasets recorded using high-sensitivity quartz ICP accelerometers installed on four bearing housing (bearings-1 to 4: two accelerometers at each bearing for dataset-1, one accelerometer at each bearing for datasets-2 and 3). Each dataset contains a vibration signal recorded for 1; s at the sampling rate of 20 kHz. In the case of dataset-1, bearing-3 and bearing-4 get inner race defect and roller element defect, respectively, at the end of the test-to-failure experiment. Bearing-1 in dataset-2 and bearing-3 in dataset-3 get OR defect at the end of the test-to-failure.

3) Padeborn University Dataset [33]: The Paderborn university dataset is the best dataset for monitoring of bearing faults of electromagnetic rotating machines under a wide variety of operating conditions. It contains recorded signals of 32 different bearing experiments categorized as follows:

1) Six experiments on healthy bearings;
2) 12 experiments on artificially damaged bearings; and
3) 14 experiments on real damaged bearings by accelerated lifetime tests.

Dataset from each experiment has measurements of motor phase currents, vibration, speed, torque, bearing temperature, and radial force. Each dataset contains 20 measurements of 4 s under four different settings of speed, torque, and force. The four settings are i) **L1: N09_M07_F10** (speed = 900 r/min, torque = 0.7 Nm and radial force = 1000 N); ii) **L2: N15_M01_F10** (speed = 1500 r/min, torque = 0.1 Nm and radial force = 1000 N); iii) **L3: N15_M07_F04** (speed = 1500 r/min, torque = 0.7 Nm and radial force = 400 N); and iv) **L4: N15_M07_F10** (speed = 1500 r/min, torque = 0.7 Nm and radial force = 1000 N). There are two categories of faults: IR damage and OR damage. Each category of faults contains wide variety of damages and different level of damage represented by the extent of damage (details can be found in [33]).

B. Preprocessing

The sensor data are usually contaminated with noises and not well structured, which makes it unsuitable for the training of a network. Data preprocessing is required, which involves filtering, clipping, smoothing, and normalization to convert the dataset well structured. We have used the preprocessed data and rescaled into [0, 1] using min–max normalization before training the network.

$$\mathbf{x}_{\mathrm{normalized}} = \frac{\mathbf{x} - \mathbf{x}_{\mathrm{min}}}{\mathbf{x}_{\mathrm{max}} - \mathbf{x}_{\mathrm{min}}} \quad (12)$$

where $\mathbf{x}$ is the un-normalized data, $\mathbf{x}_{\mathrm{normalized}}$ is the normalized data, and $\mathbf{x}_{\mathrm{min}}$ and $\mathbf{x}_{\mathrm{max}}$ are minimum and maximum values of the data. Now, the data is is split into train & test datasets using five-fold cross-validation sampling technique for better generalization of the model.

C. Segmentation and Evaluation Scheme

The recorded data files have the time-series signals and there is a huge number of sample points (at least 121 265 points) in each class which may not be directly suitable for training the DNN. Therefore, source and target datasets are prepared by segmenting the recorded samples with the segment length of approximately 100 data points for the CWRU/IMS dataset and 400 data points for the Paderborn University dataset. For example, if a time-series signal contains 121 000 points, it can be converted into 1210×100 data samples per class.

Effectiveness of the domain adaptation of the proposed algorithm is studied in two cases.

1) Case-I: CWRU dataset + IMS dataset: The source and the target dataset represents four classes: N, IR, B, and OR as provided in Table I.

i) **Source Data:** Source dataset is created from *12 k Hz DE* fault with *fault dia. = 7 mil and load = 0 hp*. With the segment length of 100 data points, 1210×100 samples per class are created. Therefore, source data has the size of 4840×100.

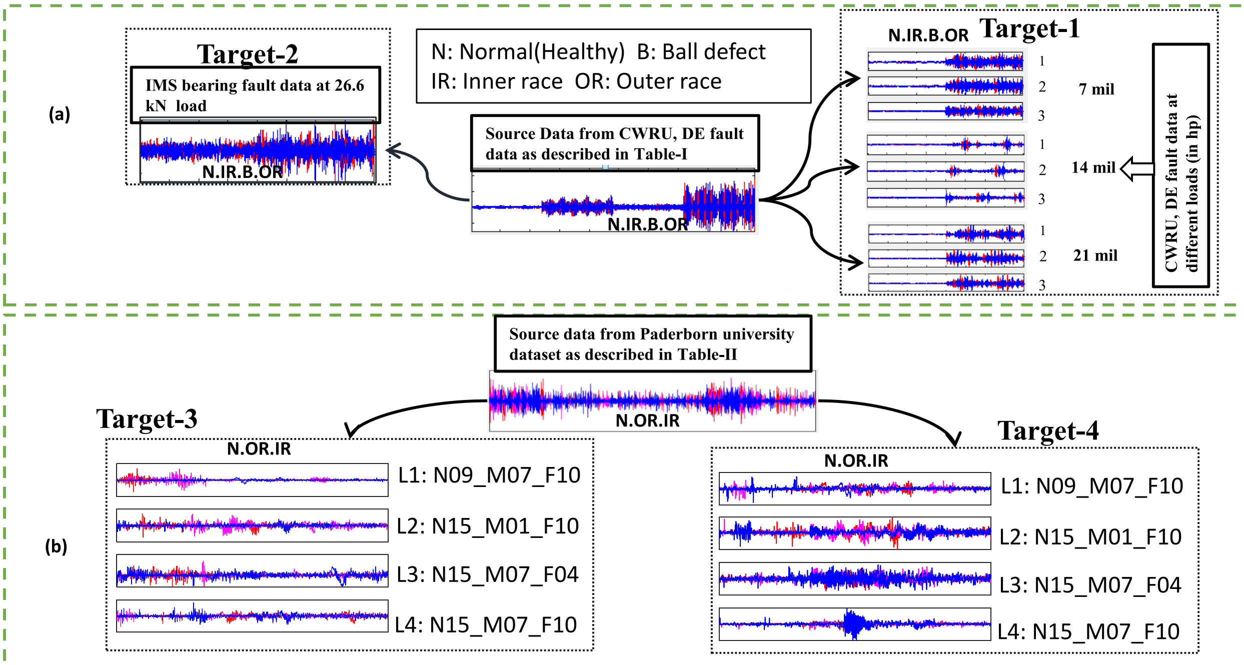

Fig. 5. Domain change from source to target for the two cases. (a) Source and target datasets as described in Table I. (b) Source and target data as described in Table II (the waveform in red, blue, and magenta colors is the vibration signals).

TABLE I
CWRU [31] AND IMS [32] DATASET DESCRIPTION

Class Name	Source (CWRU-DE)		Target-1 (CWRU-DE)		Target-2 (IMS Data)		Class Label
	Sample/ Class	Load	Sample/ Class	Load	Sample/ Class	Load	
N	1210	0 hp	400	1, 2 & 3 hp	400	26.6 kN	0
IR	1210	0 hp	400	1, 2 & 3 hp	400	26.6 kN	1
B	1210	0 hp	400	1, 2 & 3 hp	400	26.6 kN	2
OR	1210	0 hp	400	1, 2 & 3 hp	400	26.6 kN	3

ii) **Target 1:** Target 1 (T1) is prepared using the CWRU dataset for the *12 k Hz DE* fault with 7, 14, and 21 mil fault diameter and at 1, 2, and 3 hp motor load. For each case, 40 000 sample points from the time-series signal are used to create a dataset of 400×100 per class.

iii) **Target 2:** Target 2 (T2) is prepared using *IMS dataset* recorded at 26.6 kN motor load. A total of 40 000 data points are taken to create a dataset of 400×100 per class.

2) Case-II: Paderborn university dataset: The source and the target dataset represents three classes: N, OR, and IR.

i) **Source Data:** Source dataset is created using artificially damaged fault dataset with the extent of damage for OR and IR fault = 1. Each measurement file contains approximately 25 6001 sample points as a time-series signal. With the segment length of 400 data points, 10000×400 samples per class were created using 20 measurement data files from each class as mentioned in Table II.

ii) **Targets 3 and 4:** Target data [target 3 (T3) and target 4 (T4)] are selected from real damaged fault dataset as summarized in Table II. Samples for both the target datasets (T3 and T4) are considered under four load settings L1, L2, L3, and L4. 500×400 samples per class is created using one measurement file from each class.

The selection of the source and the target datasets as described above has also been pictorially shown in Fig. 5.

D. Training and Results

For the demonstration of the efficacy of the proposed method, first, the teacher model (DNN with hidden layers: $70 - 30 - 20$) is trained on the source data using Algorithm 1. The hyper-parameter of the DNN for its training is as follows: regularization parameter $(\lambda) = 0.05$, sparsity parameter $(\rho) = 0.1$, weight-penalizing deviation $(\beta) = 0.8$, method of parameter optimization = "lbfgs."

This model is used to train DNN with new architecture (here we have selected the new architecture as $70 - 50 - 30 - 20$) suitable for the machine running on different operating conditions. The target datasets from each case as described in Tables I and II are normalized and then split into train-test datasets using five-fold cross-validation sampling technique. Now, the new model is trained using Algorithm 2 on the training datasets from targets 1, 2, 3, and 4. Parameter optimization of the new model is done using backpropagation with softmax as classifier. The network is trained for 50 iteration with adaptive learning rate. Classification accuracies on the test datasets from each case are generated and presented in Tables III and IV.

To show the effectiveness of the proposed method, we have compared the results with the most advanced and similar methodologies of the fault diagnosis of rotating machines. As reported in literature review, the selected algorithms are DNN [5],

TABLE II
PADERBORN UNIVERSITY DATASET [33] DESCRIPTION

Class Name	Source (Artificial Damage)		Target-3 (T3)			Target-4 (T4)			Class label
	Bearing name (Extent of damage = 1)	Sample/Class	Bearing name	Extent of damage	Sample/Class	Bearing name	Extent of damage	Sample/Class	
N	K001+K002	10000	K001	None	500	K002	None	500	0
OR	KA01+KA05	10000	KA04	1	500	KA16	2	500	1
IR	KI01+KI05	10000	KI16	3	500	KI18	2	500	2

TABLE III
ACCURACY ON CWRU DE FAULT DATASET AND IMS DATASET

Target	Fault Dia.	Motor Load	DNN [5]	DANN [25]	DTL [26]	DAFD [27]	Net2Net	
							Without D. A.	With D. A.
T1	DE 7 mil	1hp	92.6	**99.7**	93.2	93.5	95.3	**99.7**
		2hp	89.4	90.6	92.4	90.4	97.6	**98.3**
		3hp	90.2	98.1	91.1	89.6	90.4	**99.3**
	DE 14 mil	1hp	72.9	33.1	71.0	71.7	84.9	**90.2**
		2hp	71.6	20.9	65.2	67.9	90.9	**97.9**
		3hp	72.3	31.9	67.3	69.0	87.4	**95.5**
	DE 21 mil	1hp	89.3	79.4	83.6	85.2	93.4	**97.2**
		2hp	90.9	52.2	87.8	88.0	90.6	**96.4**
		3hp	85.5	77.2	90.8	87.4	90.1	**97.7**
T2	IMS Dataset	26.6kN	83.81	85.63	82.94	81.59	87.59	**91.66**
Standard Deviation			8.4	29.7	10.8	9.5	3.8	3.15
T1 (10% samples)	DE 7 mil	1hp	62.1	90.1	83.1	80.8	89.8	**96.0**
		2hp	52.4	76.3	81.3	80.9	85.7	**92.9**
		3hp	62.2	87.4	81.4	82.7	85.2	**91.3**
	DE 14 mil	1hp	58.4	35.3	61.8	60.9	84.8	**91.5**
		2hp	46.2	25.2	59.5	57.5	80.2	**90.8**
		3hp	39.5	46.3	51.1	53.8	74.4	**87.0**
	DE 21 mil	1hp	71.7	58.2	80.6	79.4	88.0	**94.6**
		2hp	63.0	42.7	83.0	82.7	91.7	**96.1**
		3hp	62.9	88.7	83.7	82.6	89.4	**91.3**
T2 (10% samples)	IMS Dataset	26.6kN	59.2	81.4	80.0	78.2	85.6	**91.6**
Standard Deviation			8.7	24.5	12.2	11.6	4.3	2.02

domain adversarial neural network (DANN) [25], deep transfer learning (DTL) with classification loss and MMD term minimization [26], and DNN for domain adaptation in fault diagnosis (DAFD) [27]. All these models are also trained using the same dataset. The architecture of the DNN and the DTL is kept the same as the new model (student net): $70 - 50 - 30 - 20$.

DNN with the new architecture has been trained from scratch on the target datasets: T1, T2, T3, and T4. DANN has been trained using labeled source data and the unlabeled target data [25]. DTL with the new architecture is pretrained on unlabeled source data and fine-tuned on the target data based using the method in [26] and [27]. The high-level feature output of the deep learning algorithm is used to trained the softmax classifier (SC), and classification accuracies are presented in Tables III and IV. Average training time under the same computational condition for each algorithm is compared in Table V. The h-level feature visualization of all the methods for the CWRU DE fault dataset (T1, 7 mils, 1 hp) is presented in Fig. 6.

E. Discussion

Based on the results shown in Tables III and IV and the feature visualization in Fig. 6, the following points can observed.

1) The proposed method (Net2Net with DA) performs better for all operating conditions as compared to the state-of-the-art methods. Here, DNN fails to perform on the par because it is trained on the small target data (insufficient training data) from scratch. Similarly, DTL with DA overfits due to an insufficient amount of the target data. In the case of the proposed method, the knowledge of the source data is transferred via function-preserving principle to the new network with a suitable architecture. The transformed model is almost ready for the new data pattern. Further, the transformed model is fine-tuned using the target data to minimize the classification loss and the distribution discrepancy. Therefore, the proposed method performs better even if a very less amount of the target data is available.

TABLE IV
ACCURACY ON PADERBORN UNIVERSITY DATASET WITH DIFFERENT SPEED (N, M, F), LOAD, AND RADIAL FORCE

Target	Setting	DNN [5]	DANN [25]	DTL [26]	DAFD [27]	Net2Net	
						Without D. A.	With D. A.
T3	L1	88.27	92.45	93.6	94.27	95.7	**96.12**
	L2	78.73	90.23	85.6	88.93	91.2	**93.6**
	L3	84	88.45	87.13	84.6	89.12	**92.28**
	L4	78.4	87.6	88.4	87.13	93.12	**93.6**
T4	L1	89.53	**95.5**	94	94.47	90.5	90.1
	L2	78.2	92.4	90.93	92	92.2	**94.31**
	L3	84.4	89.52	91.47	91	91.1	**95.2**
	L4	80.07	91	87.73	88.27	90.02	**96.24**
Standard Deviation		4.5	2.5	3.1	3.5	2.06	2.05
T3 10% samples	L1	58.6	82.3	73.5	80.0	88.0	**92.3**
	L2	56.3	90.2	71.4	81.0	86.5	**93.0**
	L3	70.2	58.2	84.1	85.3	**90.0**	89.0
	L4	64.4	89.0	71.0	77.3	78.2	**85.0**
T4 10% samples	L1	65.3	55.0	76.0	76.0	78.0	**84.5**
	L2	58.5	48.1	76.0	80.0	85.2	**90.0**
	L3	64.8	73.2	78.2	88.0	84.2	**92.0**
	L4	60.0	90.0	79.3	83.4	87.0	**90.0**
Standard Deviation		6.4	17.3	4.4	4.0	4.4	**3.2**

TABLE V
AVERAGE TRAINING TIME (S) ON THE SAME MACHINE UNDER IDENTICAL CONDITION

Target Data Name	DNN [5]	DANN [25]	DTL [26]	DAFD [27]	Net2Net (Proposed)	
					Without D. A.	With D. A.
Target-1	165.84	2476.26	238.6	206.8	**90.4**	**55.0**
Target-2	160.96	2490.4	278.1	260.9	**80.2**	**61.8**
Target-3	156.96	2490.4	278.1	260.9	**80.2**	**61.8**
Target-4	426.96	4490.4	978.1	960.9	**215.2**	**191.8**

2) The comparison of the standard deviation (SD) over the various operating conditions shows the stability of the performance. It can be observed that the results by the Net2Net with DA are more stable with the variations in the operating conditions. Results of the DANN are the most unstable because DANN tries to adapt with an unlabeled target dataset which is very small in number.
3) For the Paderborn University dataset, source data is taken from the machine with artificially damage faults and the target datasets are from real damaged (run to failure case). From IV, it can be observed that the results with Net2Net are stable even under wide variations of the operating conditions and the extent of the damage.
4) Fig. 6 shows h-level feature visualization using the t-SNE. It can be observed that the h-level features from Net2Net with DA are more clearly separated from each other as compared to the state-of-the-art methods.
5) The performance of the proposed method under a reduced number of available labeled target samples (10% of the samples) is also presented in Tables III and IV. For the reduced number of the target samples, the effect of the domain adaptation [in (9)] is reduced, and therefore, the reduction in the performance. However, the performance of the proposed method is still better than the other state-of-the-art methods because of the knowledge transfer through network transformation.
6) If the number of training samples or the number of training iterations is reduced, the training time will be reduced, but it may result in overfitting of the network training leading to poor accuracy. On the other hand, if we wish to further improve the accuracy, more samples will be required, but this will increase the computational time.

F. Complexity of the Proposed Method

The training process of the proposed algorithm includes i) network transformation (initialization using function-preserving principle) and ii) the fine-tuning to obtain optimal parameters by solving the minimization objective in (9). The network transformation includes the conversion, insertion, and/or removal elements from parameter matrices. Therefore, the time complexity of the whole training process is mainly contributed by the

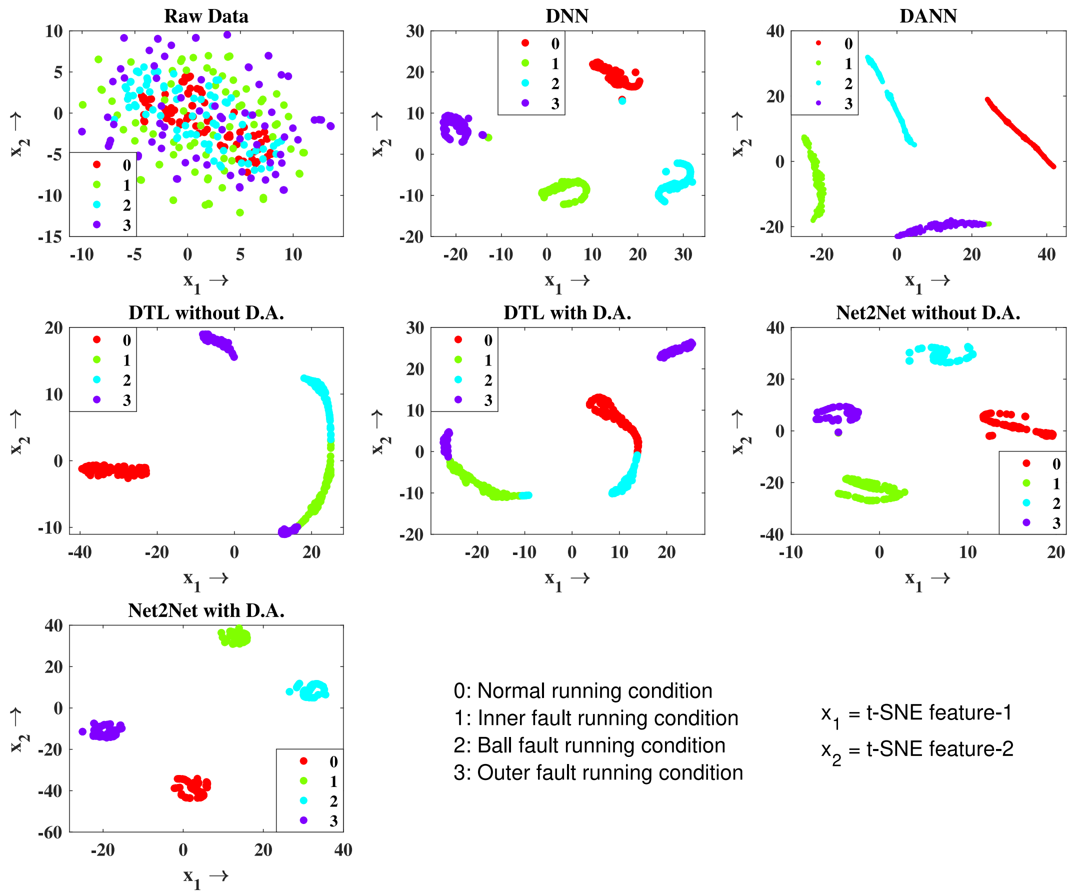

Fig. 6. Feature visualization for CWRU DE fault data (with 7 mil fault diameter and 1 hp motor load) using t-SNE.

fine-tuning process to get the optimality of (9), which is similar to training the DNN using the backpropagation algorithm. Let n denote the number of data points, d dimensionality of the input data, i, j, k are the number of nodes in the hidden layers in the network, c be the number output class, and N be the number of iterations required for the training. The time complexity analysis using $O(.)$ [39] for the process of fine-tuning can be expressed as $O(N * n * (d * i + i * j + j * k + k * c))$.

Since the fine-tuning requires a very less amount of labeled target data and only a few iterations, the training process takes very little time. The time complexity of each method listed is different because the training process in each method is different. However, assuming the same task of the fault diagnosis to be performed, the time comparison in Table V gives useful information about the quick adaptation of the proposed method under the variable operating conditions.

V. Conclusion

In this article, we have proposed a quick learning methodology based on Net2Net transformation followed by a fine-tuning to minimize classification loss and domain discrepancy. The proposed method is very effective for intelligent fault diagnosis under the variable operating conditions of industrial rotating machines. The fine-tuning process requires a very less amount of target data and a few number of iterations for fine-tuning; therefore, the method is capable for quick adaptation to the change in the operating condition. Also, the performance of the proposed method is stable under the variable operating conditions compared to the state-of-the-art methods. Therefore, the proposed method is very useful for continuous monitoring of the industrial machines.

The proposed method of network transformation has a broad scope in the future for developing lifelong learning systems. This work can be extended to develop a network optimization algorithm capable of adjusting its architecture depending upon the change in the operating conditions.

References

[1] W. Deng, S. Shang, X. Cai, H. Zhao, Y. Song, and J. Xu, "An improved differential evolution algorithm and its application in optimization problem," *Soft Comput.*, vol. 25, no. 7, pp. 5277–5298, 2021.

[2] W. Deng, J. Xu, X.-Z. Gao, and H. Zhao, "An enhanced MSIQDE algorithm with novel multiple strategies for global optimization problems," *IEEE Trans. Syst., Man, Cybern. Syst.*, doi: 10.1109/TSMC.2020.3030792.

[3] X. Cai *et al.*, "An improved quantum-inspired cooperative co-evolution algorithm with muli-strategy and its application," *Expert Syst. Appl.*, vol. 171, 2021, Art. no. 114629.

[4] A. K. Sharma, V. Singh, N. K. Verma, and J. Liu, "Condition based monitoring of machine using Mamdani fuzzy network," in *Proc. Prognostics System Health Manage. Conf.*, 2018, pp. 1159–1163.

[5] Y. Qi, C. Shen, D. Wang, J. Shi, X. Jiang, and Z. Zhu, "Stacked sparse autoencoder-based deep network for fault diagnosis of rotating machinery," *IEEE Access*, vol. 5, pp. 15066–15079, 2017.

[6] O. Janssens *et al.*, “Convolutional neural network based fault detection for rotating machinery,” *J. Sound Vib.*, vol. 377, pp. 331–345, 2016.

[7] R. Zhang, Z. Peng, L. Wu, B. Yao, and Y. Guan, “Fault diagnosis from raw sensor data using deep neural networks considering temporal coherence,” *Sensors*, vol. 17, no. 3, 2017, Art. no. 549.

[8] W. Fan, Q. Zhou, J. Li, and Z. Zhu, “A wavelet-based statistical approach for monitoring and diagnosis of compound faults with application to rolling bearings,” *IEEE Trans. Automat. Sci. Eng.*, vol. 15, no. 4, pp. 1563–1572, Oct. 2018.

[9] N. K. Verma, R. K. Sevakula, S. Dixit, and A. Salour, “Intelligent condition based monitoring using acoustic signals for air compressors,” *IEEE Trans. Rel.*, vol. 65, no. 1, pp. 291–309, Mar. 2016.

[10] F. Immovilli *et al.*, “Diagnosis of bearing faults in induction machines by vibration or current signals: A critical comparison,” *IEEE Trans. Ind. Appl.*, vol. 46, no. 4, pp. 1350–1359, Jul./Aug. 2010.

[11] O. Janssens, M. Loccufier, and S. Van Hoecke, “Thermal imaging and vibration-based multisensor fault detection for rotating machinery,” *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 434–444, Jan. 2019.

[12] G. Singh and S. A. K. S. Ahmed, “Vibration signal analysis using wavelet transform for isolation and identification of electrical faults in induction machine,” *Electric Power Syst. Res.*, vol. 68, no. 2, pp. 119–136, 2004.

[13] M. Z. Ali, M. N. S. K. Shabbir, X. Liang, Y. Zhang, and T. Hu, “Machine learning-based fault diagnosis for single-and multi-faults in induction motors using measured stator currents and vibration signals,” *IEEE Trans. Ind. Appl.*, vol. 55, no. 3, pp. 2378–2391, May/Jun. 2019.

[14] M. Ge, Y. Xu, and R. Du, “An intelligent online monitoring and diagnostic system for manufacturing automation,” *IEEE Trans. Automat. Sci. Eng.*, vol. 5, no. 1, pp. 127–139, Jan. 2008.

[15] S. Yin, S. X. Ding, X. Xie, and H. Luo, “A review on basic data-driven approaches for industrial process monitoring,” *IEEE Trans. Ind. Electron.*, vol. 61, no. 11, pp. 6418–6428, Nov. 2014.

[16] Y. Lei *et al.*, “An intelligent fault diagnosis method using unsupervised feature learning towards mechanical Big Data,” *IEEE Trans. Ind. Electron.*, vol. 63, no. 5, pp. 3137–3147, May 2016.

[17] N. K. Verma, R. K. Sevakula, and R. Thirukovalluru, “Pattern analysis framework with graphical indices for condition-based monitoring,” *IEEE Trans. Rel.*, vol. 66, no. 4, pp. 1085–1100, Dec. 2017.

[18] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, “Deep learning and its applications to machine health monitoring,” *Mech. Syst. Signal Process.*, vol. 115, pp. 213–237, 2019.

[19] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi, “Applications of machine learning to machine fault diagnosis: A review and roadmap,” *Mech. Syst. Signal Process.*, vol. 138, 2020, Art. no. 106587.

[20] L. Y. Pratt, “Discriminability-based transfer between neural networks,” in *Advances in Neural Information Processing Systems*, Morgan-Kaufmann, vol. 5, 1993.

[21] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[22] R. K. Sevakula, V. Singh, N. K. Verma, C. Kumar, and Y. Cui, “Transfer learning for molecular cancer classification using deep neural networks,” *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 16, no. 6, pp. 2089–2100, Nov./Dec. 2019.

[23] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, “Domain adaptation via transfer component analysis,” *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.

[24] M. Long, J. Wang, G. Ding, S. J. Pan, and P. S. Yu, “Adaptation regularization: A general framework for transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1076–1089, May 2014.

[25] Y. Ganin *et al.*, “Domain-adversarial training of neural networks,” *J. Mach. Learn. Res.*, vol. 17, no. 59, pp. 1–35, 2016.

[26] L. Wen, L. Gao, and X. Li, “A new deep transfer learning based on sparse auto-encoder for fault diagnosis,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 1, pp. 136–144, Jan. 2019.

[27] W. Lu, B. Liang, Y. Cheng, D. Meng, J. Yang, and T. Zhang, “Deep model based domain adaptation for fault diagnosis,” *IEEE Trans. Ind. Electron.*, vol. 64, no. 3, pp. 2296–2305, Mar. 2017.

[28] X. Li, W. Zhang, and Q. Ding, “Cross-domain fault diagnosis of rolling element bearings using deep generative neural networks,” *IEEE Trans. Ind. Electron.*, vol. 66, no. 7, pp. 5525–5534, Jul. 2019.

[29] L. Guo, Y. Lei, S. Xing, T. Yan, and N. Li, “Deep convolutional transfer learning network: A new method for intelligent fault diagnosis of machines with unlabeled data,” *IEEE Trans. Ind. Electron.*, vol. 66, no. 9, pp. 7316–7325, Sep. 2019.

[30] T. Chen, I. J. Goodfellow, and J. Shlens, “Net2net: Accelerating learning via knowledge transfer,” in *Proc. 4th Int. Conf. Learning Representations*, San Juan, Puerto Rico, May 2016.

[31] W. A. Smith and R. B. Randall, “Rolling element bearing diagnostics using the case western reserve university data: A benchmark study,” *Mech. Syst. Signal Process.*, vol. 64, pp. 100–131, 2015.

[32] J. Lee, H. Qiu, G. Yu, and J. Lin, and Rexnord Technical Services, *Bearing Data Set*, *IMS*, Univ. Cincinnati. NASA Ames Prognostics Data Repository 2007.

[33] C. Lessmeier, J. K. Kimotho, D. Zimmer, and W. Sextro, “Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification,” in *Proc. Eur. Conf. Prognostics Health Manage. Soc.*, Bilbao, Spain, 2016. [Online]. Available: http://mb.uni-paderborn.de/kat/datacenter

[34] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, vol. 19, pp. 153–160.

[35] C. Wu, M. J. F. Gales, A. Ragni, P. Karanasou, and K. C. Sim, “Improving interpretability and regularization in deep learning,” *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 2, pp. 256–265, Feb. 2018.

[36] Y. Chen, S. Song, S. Li, and C. Wu, “A graph embedding framework for maximum mean discrepancy-based domain adaptation algorithms,” *IEEE Trans. Image Process.*, vol. 29, pp. 199–213, 2020, doi: 10.1109/TIP.2019.2928630.

[37] N. Aronszajn, “Theory of reproducing Kernels,” *Trans. Amer. Math. Soc.*, vol. 68, no. 3, pp. 337–404, 1950.

[38] J. Nocedal and S. J. Wright, *Large-Scale Unconstrained Optimization*. New York, NY, USA: Springer, 2006, pp. 164–192.

[39] M. Sipper, “A serial complexity measure of neural networks,” in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, vol. 2, pp. 962–966.