Hindawi
Shock and Vibration
Volume 2021, Article ID 6627740, 12 pages
https://doi.org/10.1155/2021/6627740

Research Article

Multiclass Incremental Learning for Fault Diagnosis in Induction Motors Using Fine-Tuning with a Memory of Exemplars and Nearest Centroid Classifier

Magdiel Jiménez-Guarneros, Jonas Grande-Barreto, and Jose de Jesus Rangel-Magdaleno

National Institute for Astrophysics, Optics and Electronics, San Andrés Cholula, Mexico

Correspondence should be addressed to Jose de Jesus Rangel-Magdaleno; jrangel@inaoep.mx

Received 12 July 2021; Revised 23 September 2021; Accepted 13 October 2021; Published 27 October 2021

Academic Editor: Wenxian Yang

Early detection of fault events through electromechanical systems operation is one of the most attractive and critical data challenges in modern industry. Although these electromechanical systems tend to experiment with typical faults, a common event is that unexpected and unknown faults can be presented during operation. However, current models for automatic detection can learn new faults at the cost of forgetting concepts previously learned. This article presents a multiclass incremental learning (MCIL) framework based on 1D convolutional neural network (CNN) for fault detection in induction motors. The presented framework tackles the forgetting problem by storing a representative exemplar set from past data (known faults) in memory. Then, the 1D CNN is fine-tuned over the selected exemplar set and data from new faults. Test samples are classified using nearest centroid classifier (NCC) in the feature space from 1D CNN. The proposed framework was evaluated and validated over two public datasets for fault detection in induction motors (IMs): asynchronous motor common fault (AMCF) and Case Western Reserve University (CWRU). Experimental results reveal the proposed framework as an effective solution to incorporate and detect new induction motor faults to already known, with a high accuracy performance across different incremental phases.

1. Introduction

IMs support most of the production process in the modern industry's daily life due to their straightforward construction, reliability, and relatively low cost. However, IMs operate for long uninterrupted working periods, are exposed to the elements, and minimum preventive maintenance. These operative conditions raise unexpected faults that can show up at any time, causing lower productivity and economic losses. Thus, early motor failure detection and correction are challenging problems that catch many researchers' attention.

From a general overview, motor fault analysis methods split into signal processing and artificial intelligence approaches [1]. The first ones have been focused on analyzing diverse physical magnitudes to find features that help identify abnormal behavior in the motor's performance [2, 3]. For example, rotor vibrations [4], bearing faults [5], and broken rotor bar [2]. Meanwhile, artificial intelligence-based methods have been integrated to provide automatic fault detection using a data-driven approach. These methods base their performance on extracted features from raw signals to be used as inputs. In past years, deep learning (DL) architectures, such as autoencoders (AE) [6], convolutional neural network (CNN) [5, 7, 8], and capsule networks (CapsNet) [1], have been used in fault diagnosis due to their potential applicability for the automatic feature extraction, reported in several cases new state-of-the-art results. In the literature, most works combine DL architectures with different handcraft features and feature extractors (e.g., Fourier and Wavelet transform) [8]. Recently, some authors [9–12] have shown some promising advances to eliminate the requirement of the handcraft features, where CNN architectures have demonstrated high effectiveness. Despite this progress, classification models have been focused on detecting a set of known patterns that characterize typical faults on equipment from manufacturers. However, modifications in the operative conditions can generate patterns from new failures that differ from those detected by the current model. This issue forces existing methods to learn a new model considering unknown failure conditions.

To overcome the practical challenge mentioned above, multiclass incremental learning (MCIL) arises a promising solution by updating the current model on new data instead of training once on a whole dataset. Indeed, MCIL aims to learn new classes from previous ones, although none or a few samples of old classes are retained. Unlike the conventional classification setting, in MCIL, samples from different classes come in different time phases, whereas incremental classifiers aim to achieve a competitive performance overall seen classes [13]. Motivated by this, only a few works have been reported by traditional approaches to address multiclass incremental learning. For example, Saucedo-Dorantes et al. [14] trained a self-organizing map (SOM) every time that a new detection occurs. However, this model does not retain samples from previous classes, and the complexity of the model increases when new faults are incorporated. Incremental model transfer learning (IMTL) [15] follows a domain adaptation approach to allow a classification model to detect new faults but requires all samples from past faults during the subsequent incremental phases to achieve high performance. Overall, these works are still limited because they depend on an engineered data representation. In this direction, deep learning approaches have certain advantages by learning task-specific features and classifiers from raw signals. However, deep learning models can suffer from catastrophic forgetting [16] when they are trained incrementally, i.e., the tendency of a neural network to underfit past classes when new ones are learned.

This study presents an MCIL framework based on an 1D CNN for fault detection in IMs. To tackle the catastrophic forgetting problem, the presented framework employs a memory containing representative exemplars from past data and updates a 1D CNN model across incremental states, using a fine-tuning procedure [16]. The representative exemplars from past (known) faults are selected using the Herding method [17]. Next, nearest centroid classifier (NCC) is used to classify test samples in each incremental phase. By doing this, the proposed framework maintains a constant model complexity while new classes appear each time. We evaluated and validated the presented model over two different study cases: (1) motor common faults diagnosis and (2) bearing fault diagnostics. Experimental results show that the proposed MCIL framework effectively incorporates new faults on an 1D CNN, achieving a high accuracy performance across different incremental phases.

2. Convolutional Neural Network

Convolutional neural network (CNN) is a biologically inspired artificial neural network that processes data with a known grid-like topology [18]. CNN alternates convolution and pooling layers, followed by a fully connected layer to extract features and generate the desired output. Due to the inherent one-dimensional signals obtained from a vibration analysis in IMs, it has been preferable to deal with these signals using one-dimensional models [9, 11]. Thus, we first describe 1D convolution operators, which are used in the presented work. Then, we describe the complement layers that integrate a convolutional network.

In its standard approach, CNN performs a set of convolutions between an input signal and some finite impulse response (FIR) filters. The convolution operation ($*$) is described as a weighted average of an input signal $\mathbf{x}_i$:

$$\begin{aligned} s_i &= \mathbf{k}_i * x_i \\ &= \sum_{j=0}^{L-1} \mathbf{k}_{i-j} \cdot x_j, \end{aligned} \tag{1}$$

where s_i is called i-th feature map and $\mathbf{k}$ denotes a weighting factor, called filter or kernel, with length L. The kernels are built to identify spatial features on the input data. The output from a convolutional layer defines the next layer's activation value. Then, the output $s_i^{(l)}$ of a convolutional layer (l) at i-th feature map is defined as follows:

$$s_i^{(l)} = \sigma\left(\sum_{j=1}^{L} \mathbf{K}_{i,j}^{(l)} * \mathbf{x}_j^{(l-1)} + \mathbf{b}_i^{(l)}\right), \tag{2}$$

where $\mathbf{K}_{i,j}^{(l)}$ denotes each local weighting factor of the kernel, $\mathbf{x}_j^{(l-1)}$ represents the j-th feature map at the layer $l-1$, L is the number of filters applied over $\mathbf{x}_j^{(l-1)}$, $\mathbf{b}_i^{(l)}$ is the bias, and σ is the activation function.

Most of the time, raw data contain noise and undesirable spectral shapes that affect the feature extraction process [19]. Motivated from this issue, the SincNet layer [9, 20], an extension of the standard convolution, applies a set of temporal convolutions between a raw signal and digital filters to boost the first convolutional layer output.

2.1. Sinc Convolution.

Instead of learning the filters from the data, as the conventional CNN, the SincNet [9, 20] performs the convolution operation with a preset function g that requires only a reduced set of learnable parameters Θ_s, as defined in the following equation:

$$\mathbf{y}[n] = \mathbf{x}[n] * g[n, \Theta_s], \tag{3}$$

where g is a filter bank for band-pass filter in the frequency domain; it takes advantage of the Sinc function to convert to time domain through the inverse Fourier transform [19]. The use of rectangular filters represents a practical selection to define g. The magnitude Φ of a generic band-pass filter can be described as the difference between two low-pass filters.

$$\Phi[f, f_L, f_H] = \mathrm{rect}\left(\frac{f}{2f_H}\right) - \mathrm{rect}\left(\frac{f}{2f_L}\right), \tag{4}$$

where $\Theta_s = \{f_L, f_H\}$ the set of the trainable parameters; f_L and f_H represent low and high cutoff frequencies, respectively, of the band-pass filter learned by the Sinc filters. $\mathrm{rect}(\cdot)$ describes the rectangular function at the instant t as follows:

$$\mathrm{rect}(t) = \begin{cases} 1, & |t| \leq 0.5, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

Using the inverse Fourier transform, the reference function g becomes

$$g[n, f_L, f_H] = 2f_H \cdot \sin c(2\pi f_H n) - 2f_L \cdot \sin c(2\pi f_L n), \quad (6)$$

where the Sinc function is defined as $\mathrm{sinc}(\mathbf{x}) = \sin(\mathbf{x})/\mathbf{x}$. Finally, to achieve an approximation of the ideal band-pass filter, a winnowing procedure is applied. This procedure multiplies the truncated function g with a window function ω [21], intending to smooth out the abrupt discontinuities at the ends of g:

$$g_\omega[n, f_L, f_H] = g[n, f_L, f_H] \cdot \omega[n, f_L, f_H]. \quad (7)$$

Therefore, the succeeding layers learn the filter gain of each actual layer.

2.2. Pooling Layers. These layers perform a downsampling to reduce the spatial size of features, encouraging the input data's invariance to spatial translations. In particular, a max-pooling layer reporting the j-th maximum element within a rectangular frame for each feature map. Meanwhile, a global average pooling (GAP) layer replaces the fully connected layers in a CNN model [22], averaging the feature maps from previous convolutional layers. GAP aims to force correspondences between learned feature maps and classes in the previous convolutional layers.

3. MCIL Methodology

Let $\mathcal{X}$ and $\mathcal{Y}$ denote a feature and a label space, respectively. Let $\mathbf{D} = (\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^{N}$ be a labeled dataset with N samples, where $(\mathbf{X}, \mathbf{Y}) \in \mathcal{X} \times \mathcal{Y}$. In a classification problem, a task $\mathcal{T}$ consists in learning a labeling function G, such that $G\colon \mathcal{X} \longrightarrow \mathcal{Y}$. Notice that G represents a deep neural network with parameters Θ, so that $\mathbf{Y} = G(\mathbf{X}; \Theta)$. Likewise, G can be expressed as a composition of two functions, $G = G_y \circ G_f$, where $G_f\colon \mathcal{X} \longrightarrow \mathcal{Z}$ is a feature extractor and $G_y\colon \mathcal{Z} \longrightarrow \mathcal{Y}$ feature labeling with parameters Θ_f and Θ_y, respectively; here, $\mathcal{Z}$ is a latent feature space. The feature extractor G_f takes $\mathbf{X}$ and produces a latent feature dataset $\mathbf{Z}$. Then, G_y receives $\mathbf{Z}$ as input and produces label classifications $\mathbf{Y}$, i.e., $\mathbf{Y} = G_y(G_f(\mathbf{X}; \Theta_f); \Theta_y)$.

We focus on multiclass incremental learning (MCIL) where the model complexity is maintained constant during N incremental states, while a reduced number of samples is retained from past classes [13, 23]. We assume $N + 1$ phases, that is, N incremental phases and one initial phase S_0. A model G_0 is learned on a dataset $\mathbf{D}_0$ during the phase S_0. Due to this, we assume a memory limitation, all samples from $\mathbf{D}_0$ cannot be stored, so that exemplars $\mathbf{E}_0$ are selected and stored as a replacement of $\mathbf{D}_0$ with $|\mathbf{E}_0| \ll |\mathbf{D}_0|$. In the i-th incremental phase, dataset $\mathbf{D}_i$ from C_i classes is streamed, whereas exemplars $E_{0:\,i-1}$ from phases 0 to $i - 1$ are stored in memory. The aim of MCIL is to learn a model G using exemplars $E_{0:\,i-1}$ and data set $\mathbf{D}_i$.

Figure 1 shows the flowchart of the MCIL methodology for fault detection in IMs. In the initial phase, a model G, that is 1D CNN, is trained via cross-entropy loss $\mathcal{L}_{\mathrm{ce}}$ on the dataset $\mathbf{D}_0$, containing signals from different motor conditions. Next, exemplars $\mathbf{E}_0$ are selected using Herding method [17] over $\mathbf{D}_0$ in feature space, $\mathbf{Z}_0 = G_f(\mathbf{D}_0)$. The nearest centroid classifier (NCC) is used to classify test samples in the current phase using $\mathbf{E}_0$ as training set. In the i-th incremental phase, the output layer from the CNN is extended with randomly initialized weights for each new class. Then, the 1D CNN is fine-tuned over $\mathbf{D}_i$ and $\mathbf{E}_{0:\,i-1}$ using cross-entropy loss $\mathcal{L}_{ce}$; notice that imbalance data are produced in $\mathbf{D}_i \cup \mathbf{E}_{0:\,i-1}$ because $\mathbf{E}_{0:\,i-1}$ contains a reduced set of exemplars from past classes. This procedure updates all parameters Θ of 1D CNN. The resulting trained model G in phase i is used to extract features from $\mathbf{D}_i$ and $\mathbf{E}_{0:\,i-1}$. Herding method is used to select the exemplar set $\mathbf{E}_i$ over $\mathbf{D}_i$ in feature space. Then, NCC uses $\mathbf{E}_{0:\,i}$ as training set in feature space to classify test samples. This procedure is repeated over the different incremental phases.

3.1. CNN Architecture. The 1D CNN architecture is shown in Figure 2. 1D time-domain signals are used as inputs to the 1D CNN. One Sinc layer [9, 20] and two standard convolution layers were incorporated into the feature extractor. Conv $a \times b - c$ denotes the convolution layer of c filters with a size a and a stride b. For the lower layers, large size filters were employed to deal with high frequencies present in data. We added max-pooling layers to reduce the spatial feature dimensions. Likewise, a global average pooling layer is used to reduce the spatial dimensions of the learned features. The output layer is extended for each new class with a random initial value. The softmax activation function is used at the output layer to perform motor fault classification.

3.2. Exemplar Set Selection. The exemplar set $\mathbf{E}_i$ is adjusted in each incremental phase i using the Herding method [17], as shown in Algorithm 1. The exemplar selection is required when training data are available. Feature representation from dataset $\mathbf{D}_i$ is obtained using the feature extractor G_f (line 2). Each sample is normalized employing the L2 norm (line 3). Notice that m exemplars are selected and stored iteratively for each class (lines 5–7). One sample is added to the exemplar set in each iteration, prioritizing that sample that makes the average feature vector better approximate the mean vector.

3.3. Nearest Centroid Classifier. Nearest centroid classifier (NCC) [24] is a nearest-neighbor classifier, which is used to address the bias produced on new classes by training CNN over imbalanced data. The procedure followed by NCC is described in Algorithm 2. First, feature representations from exemplars $E_{0:\,i}$ are obtained using the feature extractor G_f

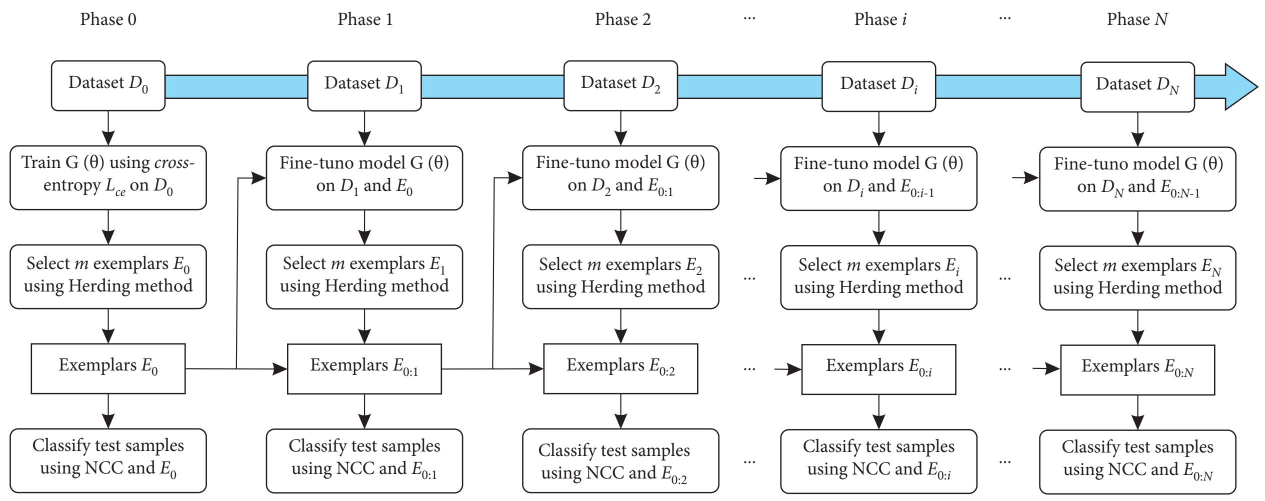
Phase 0
Phase 1
Phase 2
···
Phase i
···
Phase N
Dataset D_0
Dataset D_1
Dataset D_2
Dataset D_i
Dataset D_N
Train G (θ) using *cross-entropy* L_{ce} on D_0
Fine-tuno model G (θ) on D_1 and E_0
Fine-tuno model G (θ) on D_2 and $E_{0:1}$
Fine-tuno model G (θ) on D_i and $E_{0:i-1}$
Fine-tuno model G (θ) on D_N and $E_{0:N-1}$
Select m exemplars E_0 using Herding method
Select m exemplars E_1 using Herding method
Select m exemplars E_2 using Herding method
Select m exemplars E_i using Herding method
Select m exemplars E_N using Herding method
Exemplars E_0
Exemplars $E_{0:1}$
Exemplars $E_{0:2}$
Exemplars $E_{0:i}$
Exemplars $E_{0:N}$
Classify test samples using NCC and E_0
Classify test samples using NCC and $E_{0:1}$
Classify test samples using NCC and $E_{0:2}$
Classify test samples using NCC and $E_{0:i}$
Classify test samples using NCC and $E_{0:N}$

Figure 1: Flowchart of the presented methodology in order to train an incremental classifier for the fault detection in IMs.

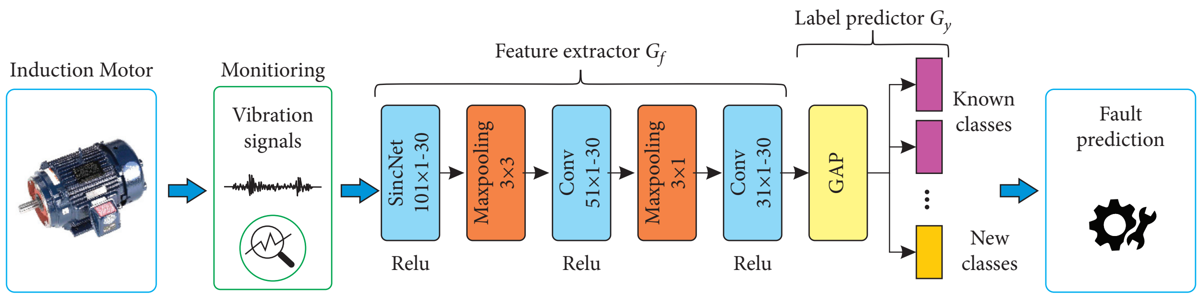

Figure 2: Architecture of the 1D CNN model.

```
    Inputs: $X_c = \{x_1, \ldots, x_{n_c}\}$: dataset of class $c$; $m$: number of exemplars to select; $G_f$: feature extractor
    Output: $\mathbf{E}$: exemplar set
(1) Initialize $\mathbf{E}$ to the empty set {}
(2) $\mathbf{Z}_c \leftarrow G_f(\mathbf{X}_c)$ \(▷\) Get feature representations
(3) $\widehat{\mathbf{Z}}_c \leftarrow \mathbf{Z}_c / \|\mathbf{Z}_c\|_2$ \(▷\) L2 norm
(4) $\overline{z} \leftarrow 1/n_c \sum_{\widehat{z} \in \widehat{Z}_c} \widehat{z}$ \(▷\) Get mean feature vector
(5) for $k = 1, \ldots, m$ do
(6)     $\mathbf{p}_k \leftarrow \arg\min_{\widehat{z} \in \widehat{Z}_c} \|\overline{z} - (1/k)[G_f(\widehat{z}) + \sum_{j=1}^{k-1} G_f(\mathbf{p}_j)]\|_2$
(7) end for
(8) $\mathbf{E} \leftarrow \{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$
```

Algorithm 1: Herding method.

(line 1). The centroid is computed as the point from which the sum of the distances of all exemplars that belong to that particular class are minimized (lines 2–4). NCC assigns the label of the most similar class centroid to the test sample $\mathbf{x}_t$ (line 5) as follows:

$$\mathbf{y}^* = \arg\min_{c \in 1,\ldots,C} d\left(\overline{p}_c, G_f(\mathbf{x}_t)\right), \tag{8}$$

where $\overline{p}_c$ is the centroid vector for the class c, obtained from exemplars $\mathbf{E}_{0:\,i}$; meanwhile, d is the Euclidean distance.

```
    Inputs: $\mathbf{E}_{0:\,i} = \{\mathbf{p}_1, \ldots, \mathbf{p}_t\}$: exemplar set from phase 0 to $i$; $G_f$: feature extractor; $\mathbf{x}_t$: sample to be classified
    Output: $\mathbf{y}^*$: one hot vector of the class label; $C$: number of classes
(1) $\widehat{Z} \leftarrow G_f(\mathbf{E}_{0:\,i})$ \(▷\) Get feature representations
(2) for $c = 1, \ldots, C$ do
(3) $\overline{p}_c \leftarrow \arg\min_{\mathbf{p}_1 \in \widehat{Z}_c} \sum_{\mathbf{p}_2 \in \widehat{Z}_c} d(\mathbf{p}_1, \mathbf{p}_2)$
(4) end for
(5) $\mathbf{y}^* \leftarrow \arg\min_{c \in 1,\ldots,C} \|\overline{p}_c - G_f(\mathbf{x}_t)\|_2$ \(▷\) nearest prototype
```

Algorithm 2: Nearest centroid classifier.

4. Experimental Setup

This section first describes data from the different cases of study used in multiclass incremental learning for motor fault diagnosis. Next, the experimental protocol is described. Finally, we present the implementation details of the MCIL model.

4.1. Cases of Study. Our experiments were conducted on two cases of study based on vibration analysis: (1) motor common fault diagnosis and (2) bearing fault diagnosis. For this, we used two public benchmark datasets: asynchronous motor common fault (AMCF) [1] and Case Western Reserve University (CWRU) [25]. Tables 1 and 2 present the description of the data acquisition and studied faults for the AMCF and CWRU datasets, respectively. The AMCF dataset is composed of 8,000 samples from 8 motor conditions (1,000 per class), where each sample contains 1,024 points. For the CWRU dataset, experiments were performed under 1 hp workload. This dataset contains three types of fault locations in bearing (balls, inner race, and outrace), showing fault diameters of 0.007, 0.014, and 0.021 inches. CWRU contains 10,000 samples from 10 different motor conditions (1,000 per class), including health bearings.

4.2. Experimental Protocol. For each dataset, we evaluated the proposed MCIL model starting from a pretrained CNN over initial data of motor faults; meanwhile, the rest of the data coming in different phases are used to train CNN in a class-incremental way. First, we fix the number of stored exemplars to the smallest memory size allowed, and after, the number of incremental phases is varied. Next, we fix the number of incremental phases to 6 and 8 for AMCF and CWRU, while the number of exemplars per fault is varied considering $m = 5$ and $m = 10$. In each incremental phase, faults are given in a fixed random order; 80% of the data samples in each class are used for training, and the remaining 20% for testing, performing a stratified sampling. The final model in each incremental phase is used to classify classes observed so far. Experiments were repeated five times using different random initial weights, different partitions of data, and a different fault order. We calculate the average accuracy and standard deviation only for incremental states, which are of interest for MCIL. Our comparison includes the results of CNN employing all previous data available (Full) and those using a fine-tuning procedure with a random selection of exemplars (FT + R).

4.3. Implementation Details and Model Parameter Selection. Table 3 presents the details of the 1D CNN model. We used filters with a large (101), medium (51), and small (11) size to learn features from raw signals. In addition, max-pooling layers were used with a size and stride of 3 to reduce spatial feature dimensions. The 1D CNN model employs a total of 56,281 trainable parameters. The 1D CNN model was implemented in Pytorch 1.7.0, whereas NCC was obtained from scikit-learn library (https://scikit-learn.org/stable/). Experiments were performed using a PC Intel(R) Core (TM) i7 with a graphic card GTX 1080 Nvidia on Ubuntu 20.04 LTS.

In our experiments, the 1D CNN model was trained by Adam algorithm [26] during 40 and 30 epochs for the AMCF and CWRU datasets, respectively. For both datasets, the initial learning rate was set to 0.0001 at the initial phase, whereas it was set to 0.001 for incremental phases. In addition, a learning decay of 0.1 was applied at 30 and 20 epochs. Likewise, a batch size of 30 was selected from {10, 30, 50} for both datasets. This hyperparameter setting was selected after comparing different configurations across 6 and 8 incremental phases on AMCF and CWRU; 5 exemplars from each past class (known faults) were stored in memory. For model parameter selection, we used coordinate descent [27], which changes only one hyperparameter at a time, aiming the best configuration. Fine-tuning and Herding selection (FT + H) were used for CNN retraining and exemplar selection in each incremental phase. Experimental results, as shown in Table 4, indicate that the batch size has a lower negative impact compared with learning rate. For both datasets, the 1D CNN model achieves its highest average accuracy when the learning rate is 0.001 and the batch size is 10 and 30. This last value of batch size was selected because it requires a lower number of iterations for data processing during training. Finally, as shown in Figure 3, the 1D CNN model stabilizes its training above 20 and 15 epochs on AMCF and CWRU for the different incremental phases. Using 40 and 30 epochs during training, we ensure a stabilization of the 1D CNN model.

5. Results

5.1. Case 1: Motor Common Fault Diagnostics. Table 5 shows the average accuracy and standard deviation (SD) on AMCF

Table 1: Description of the AMCF dataset.

Data acquisitions	
Motor	4 hp YE2-100L2-4
Sensor type	CT1020 L accelerometer
Signal description	Vibration signals were collected using an acquisition card PCI-1716 with a sampling frequency of 250 kS/s, high-resolution of 16 bits, and 16 SE/8 DI channels
Fault description	
Type of fault	The damages of the motors are as follows: short circuit of 2 turns (SC2T), short circuit of 4 turns (SC4T), short circuit of 4 turns (SC8T), air-gap eccentricity (AE), rotor bar broken (RBB), bearing cage broken (BCB), and bearing abrasion fault (BAF)

Table 2: Description of the CWRU dataset.

Data acquisitions	
Motor	2 hp reliance electric
Sensor type	Accelerometers
Signal description	Vibration signals were collected using a 16-channel DAT recorder. Digital data was collected at 12k samples/second
Fault description	
Bearing	6205-2RS JEM SKF, deep groove ball bearing
Fault location	Balls, inner race, and out race
Fault diameters	0.007, 0.014, and 0.021 inch

Table 3: Implementation details of the 1D CNN model.

Block	Layer name	Hyperparameters	Number of trainable parameters
Input	Sample input	—	
SincNet	SincNet	Filters = 30, size = 101, stride = 1	60
	Activation function	ReLU	—
	Max-pooling	Size = 3, stride = 3	—
Conv1	Convolution 1D	Filters = 30, size = 51, stride = 1	45,930
	Activation function	ReLU	—
	Max-pooling	Size = 3, stride = 3	—
Conv2	Convolution 1D	Filters = 30, size = 11, stride = 1	9,930
	Activation function	ReLU	—
GAP	Global average pooling	—	—
Output	Fully connected	C units	361
	Activation function	Softmax	—

Table 4: Accuracy results of 1D CNN on AMCF and CWRU datasets, using different batch sizes and learning rates.

	Batch size			Learning rate		
AMCF	10	30	50	0.01	0.001	0.0001
1D CNN (FT + H)	**94.97** ± 3.40	94.24 ± 3.96	91.20 ± 07.06	80.05 ± 08.33	**94.24** ± 03.96	58.67 ± 14.14
CWRU						
1D CNN (FT + H)	98.77 ± 0.83	**98.78** ± 0.83	98.68 ± 01.09	66.42 ± 09.66	**98.78** ± 00.83	87.58 ± 06.10

Best results are boldfaced for each setting.

using a different number of incremental phases and exemplars. We observed that the most challenging scenario is presented when one exemplar per fault is retained across different incremental phases. Inversely, we can see that the most straightforward scenario is presented when a greater number of exemplars per fault is stored (m = 5 and m = 10). Notice that the performance of FT + R (fine-tuning with a random selection) dropped when m = 1 and the number of incremental phases decreased. In this scenario, the proposed MCIL framework (FT + NCC + H) obtained average accuracies beyond 94%, outperforming to FT + R at least 20 percentage points (pp). Moreover, we can see that FT + NCC + H achieved average accuracies of 98.32% and 98.85% over 6 incremental phases and a number of exemplars m equal to 5 and 10, outperforming to FT + R by 5.07 and 3.79 pp.

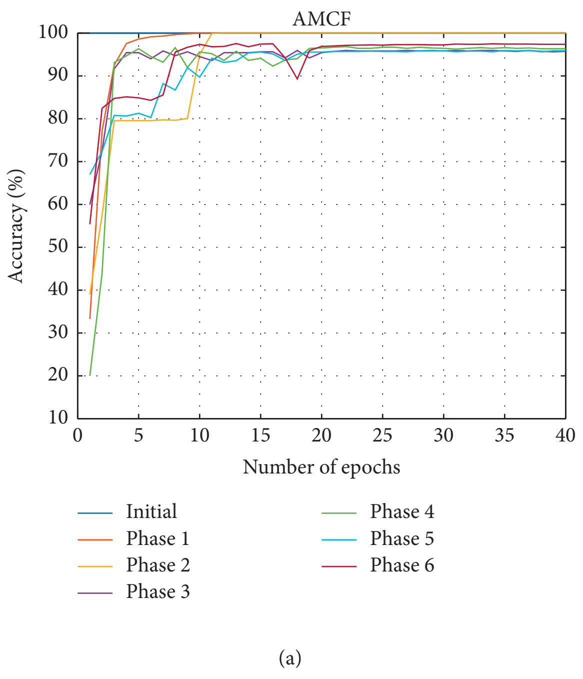
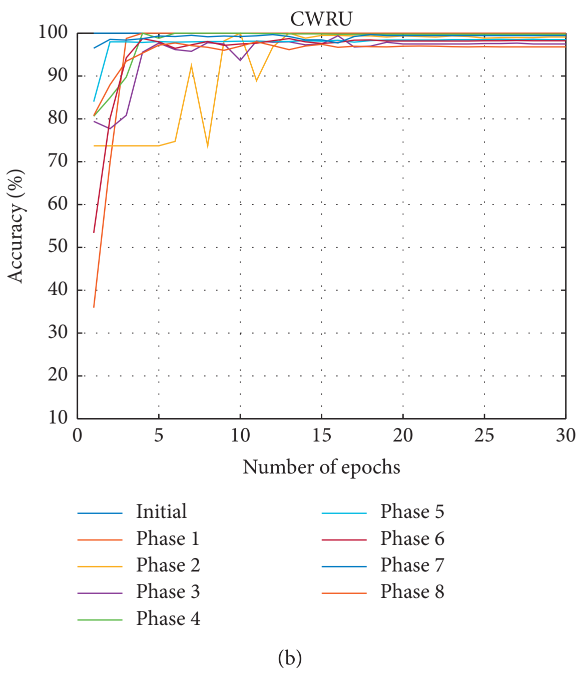
Figure 3: Accuracy performance of 1D CNN (FT + H) across different epochs. We show the accuracy performance for the different incremental phases.

Table 5: Accuracies and standard deviations (SD) over AMCF using a different number of incremental phases and exemplars.

	$m = 1$			Phases = 6	
	1 phase	3 phases	6 phases	$m = 5$	$m = 10$
FT + R	55.40 ± 03.85	68.96 ± 14.63	75.66 ± 08.37	93.25 ± 03.67	95.06 ± 03.53
FT + NCC + H	**96.70** ± 02.33	**94.73** ± 02.03	**96.22** ± 02.26	**98.32** ± 00.91	**98.85** ± 00.63
Full			99.38 ± 00.31		

Best results are boldfaced for each setting.

5.2. Case 2: Bearing Fault Diagnostics. Table 6 shows the average accuracies and standard deviations (SD) on CWRU using a different number of incremental phases and exemplars. The most challenging scenario is presented when $m = 1$, where the performance of FT + R dropped when the number of phases increased. In this scenario, FT + NCC + H achieved average accuracies beyond 93%, outperforming FT + R at least 6.27 pp. On the other hand, we observed that FT + NCC + H outperformed FT + R by only 0.48 and 0.39 pp across 8 incremental phases, while the number of exemplars is 5 and 10.

5.3. Ablation Studies

5.3.1. Effect of Each Component. We analyzed the impact of each component to determine its contribution over the final accuracy on AMCF and CWRU. Figure 4 shows the accuracy performance on AMCF and CWRU during 6 and 8 incremental phases, although one exemplar is retained in memory. Notice that accuracy results of FT without memory also were included. For both datasets, we observed that FT reduces its accuracy performance over incremental phases if memory is not available, suggesting the presence of the catastrophic forgetting problem. We can see that fine-tuning results significantly improved when memory is incorporated (FT + H), storing representative samples from past faults. Finally, notice that NCC also had a positive impact on the final accuracy (FT + NCC + H) by reducing the bias generated by incorporating new faults.

5.3.2. Effect of the Number of Exemplars. Figure 5 shows the impact on accuracy performance by varying the number of exemplars per fault. We observed that FT + R improved its results when the number of stored exemplars increased, while FT + NCC + H obtained results above 96% starting from 1 exemplar per class. We can see that FT + NCC + H achieved a competitive performance (98.32% vs. 99.38%) than training on full data, storing at least 5 exemplars per fault, while FT + R became competitive by using more than 20 exemplars.

Regarding the CWRU results, we can see that the worst performance is obtained when the number of stored exemplars per class is 1. Moreover, FT + R and FT + NCC + H

Table 6: Average accuracy and standard deviation (SD) for MCIL methods over CWRU, using a different number of incremental phases of exemplars.

	$m = 1$			Phases = 8	
	1 phase	4 phases	8 phases	$m = 5$	$m = 10$
FT + R	91.65 ± 03.40	83.90 ± 03.44	81.22 ± 04.22	98.65 ± 01.00	98.86 ± 00.96
FT + NCC + H	**97.92** ± 01.18	**94.77** ± 00.88	**93.36** ± 01.96	**99.13** ± 00.54	**99.25** ± 00.39
Full			100.0 ± 0.00		

Best results are boldfaced for each setting.

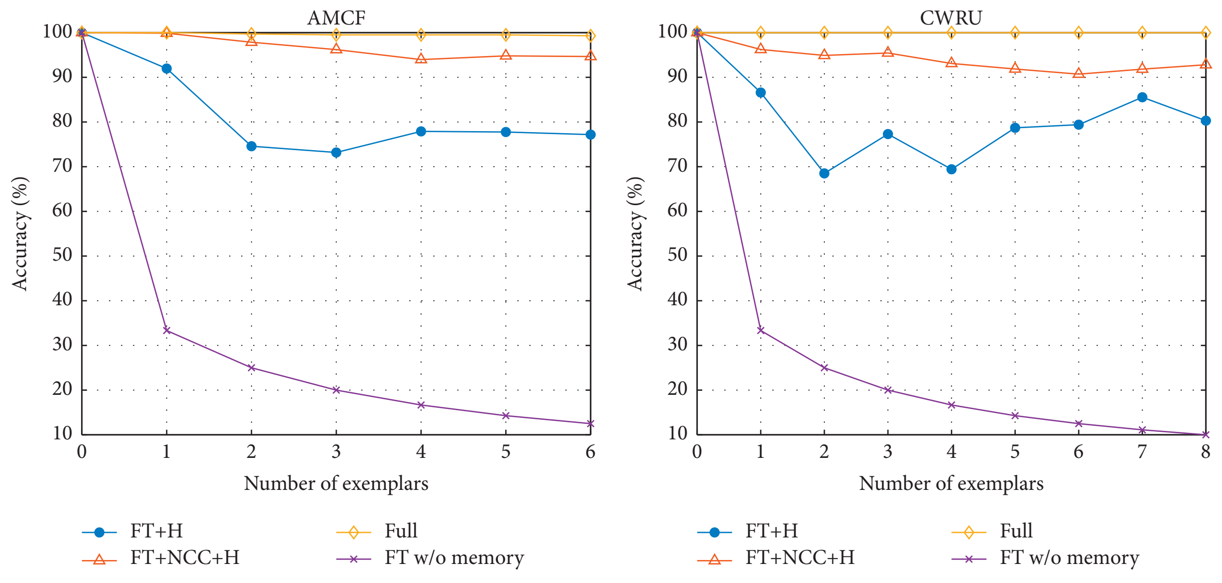

Figure 4: Impact of each component of FT + NCC + H on (a) AMCF and (b) CWRU across 6 and 8 incremental phases, retaining one exemplar per fault. Accuracy results of FT without (w/o) memory also were included.

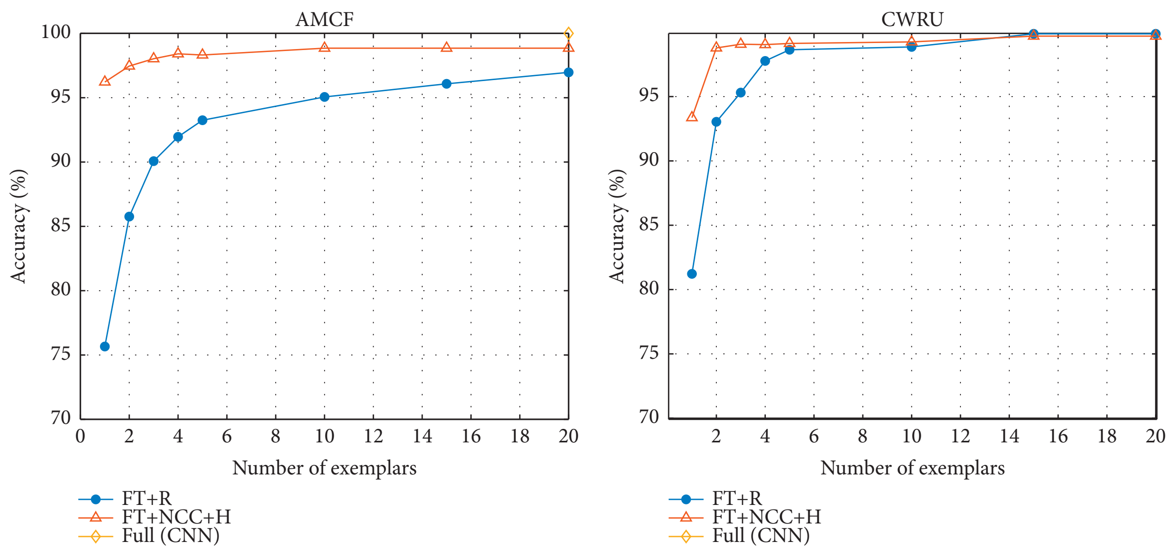

Figure 5: Impact of the number of exemplars stored in memory: (a) AMCF and (b) CWRU.

increased their accuracy performance starting from 2 samples per fault. FT + NCC + H obtained an average accuracy above 99%, storing at least 3 exemplars per fault, while FT + R achieved the same performance above 5 exemplars.

5.3.3. Effect of the Herding Method for Exemplar Selection. We studied the impact of the exemplar selection via Herding and a random selection over the accuracy performance on AMCF and CWRU. Figure 6 shows the accuracy performances on AMCF and CWRU across 6 and 8 incremental phases, while a different number of exemplars from past faults are retained. For AMCF, we observed that the Herding method (marked as + H) slightly improves the accuracy performance over random selection (marked as + R) when 1 to 10 exemplars are retained. For CWRU, only improvements can be seen when 2 to 5 exemplars are retained. We observed that random selection obtained a similar or even better performance than the herding method for the rest of the cases.

5.3.4. Effect of Noise over the Proposed Framework. In order to test the performance of the proposed framework under different noise conditions, we applied additive white Gaussian noise (AWGN) to the raw signals from the test set; 6 and 8 incremental phases were used on AMCF and CWRU, retaining 5 exemplars from past classes. Table 7 presents the classification results of FT + NCC + H under three different noise levels; accuracy results of FT + H (CNN trained incrementally) and the full model (CNN using all data) were included as reference. As expected, evaluated solutions reduced their average accuracy when a lower noise level is applied. However, we can see that FT + NCC + H obtained average accuracies beyond 92% and 94% when an SNR = 5 is applied on signals from AMCF and CWRU, respectively. For AMCF and CWRU, FT + NCC + H obtained the best average accuracies when SNR is 5 and 10, while it obtained a similar accuracy performance compared with the full model when SNR is 15.

5.3.5. Comparison of Classification Time. We analyzed the classification times of our proposed framework (FT + NCC + H) across different incremental phases; we included times of FT + H (CNN trained incrementally) as reference. For this experiment, 6 and 8 incremental phases were used for AMCF and CWRU, although 5 exemplars from each learned class were stored in memory. Table 8 shows the classification times for evaluated solutions. We can see that times increased when new classes are added to the 1D CNN classifier. Also, we observed that the times of FT + NCC + H did not significantly increase with respect to FT + H (CNN as classifier). From this, notice that NCC uses a reduced number of exemplars from past and current faults as training set.

6. Discussion

In experiments, we evaluated and validated the proposed MCIL framework on two different cases of study for motor fault detection in IMs. The evaluation was performed under scenarios where data from new faults are streamed in different time phases. From the results, we found that the MCIL framework allows the incorporation and detection of past and new motor faults from vibration signals with high accuracy across different incremental phases. Unlike previous works [14, 15], one or more faults can be added to the 1D CNN model in each incremental phase. Notice that computational requirements and memory should be bounded. In this sense, the proposed MCIL framework maintains a constant complexity while a few samples from past faults are retained. To the best of our knowledge, this is the first work that studies MCIL, based on a deep learning approach, for fault diagnosis in IMs from vibration signals.

From ablation studies, we observed that a neural network model tends to forget previously learned faults. This problem is known as catastrophic forgetting, which is produced by incorporating new faults into a pretrained model in a sequential way. In this direction, we found that the fine-tuning procedure with a memory of exemplars and the NCC classifier provides an effective solution to tackle the catastrophic forgetting problem [16] for fault diagnosis in IMs. As expected, the average accuracies of evaluated solutions significantly improved when the number of retained exemplars in memory increased. Notice that results on AMCF showed that at least 5 exemplars per fault are required across 6 incremental phases to achieve a competitive accuracy than training on full data. Also, we found that at least 3 exemplars were required across 8 incremental phases to obtain a similar performance using all data on CWRU. Notice that this amount of stored exemplars per fault represents approximately 1% of the size of the training set. Moreover, AMCF results showed that a greater number of incremental phases do not negatively impact the accuracy performance of the 1D CNN model; CWRU results showed that a greater number of incremental phases negatively impact the MCIL model's accuracy performance. Concerning to the exemplar selection, we found that the herding method slightly improved over the accuracy results than using a random selection when a few exemplars are retained, but similar or even worst results were obtained in other cases. Regarding noise conditions, we found that FT + NCC + H provides a robustness to disturbances in signals, outperforming to the full model in accuracy performance for SNRs with low values. In particular, we found that NCC helps to face such disturbances in signals. Finally, we found that NCC does not increase the classification time because a reduced number of samples are used as training set.

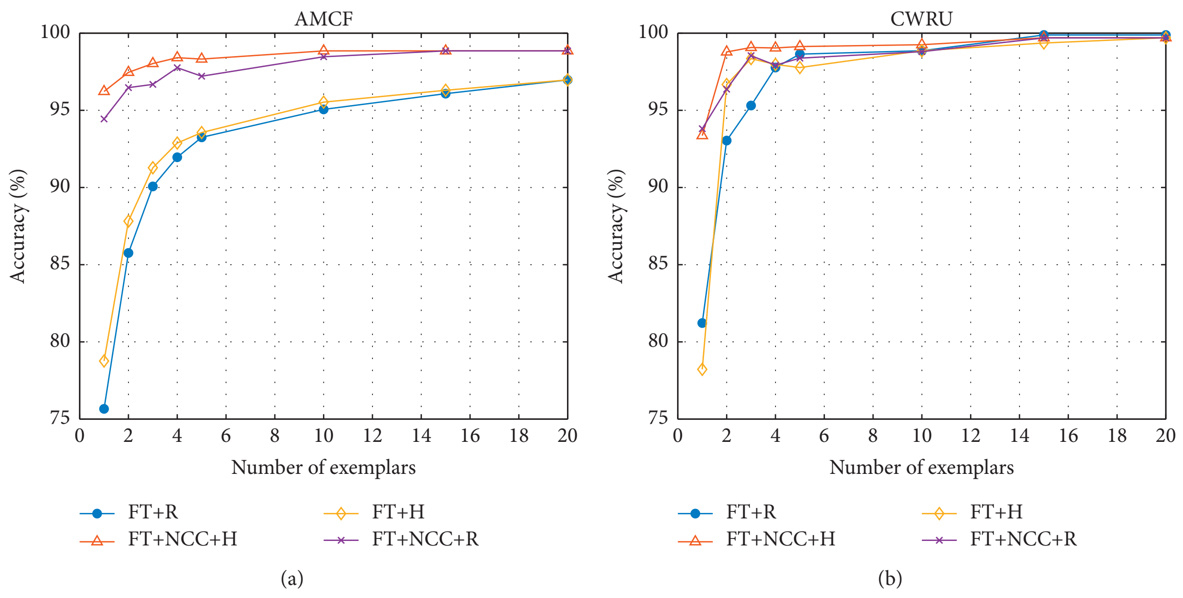

Figure 6: Impact of the herding method for exemplar selection on (a) AMCF and (b) CWRU. Herding method is marked as + H, whereas a random selection is marked as + R.

Table 7: Accuracy performances and standard deviations for evaluated solutions under different noise levels on AMCF and CWRU.

	SNR		
	5	10	15
AMCF			
Full	74.21 ± 00.91	94.41 ± 01.70	99.45 ± 00.21
FT + H	85.68 ± 03.37	93.55 ± 04.14	94.96 ± 03.32
FT + NCC + H	92.51 ± 03.22	98.15 ± 01.03	98.39 ± 00.89
CWRU			
Full	70.16 ± 06.18	96.20 ± 01.13	99.81 ± 00.07
FT + H	77.74 ± 07.08	93.25 ± 02.84	97.33 ± 01.62
FT + NCC + H	94.35 ± 04.49	98.77 ± 01.03	98.92 ± 00.89

Table 8: Classification time (seconds) across different incremental phases.

AMCF		
Incremental phases	FT + H	FT + NCC + H
Initial	0.1092	0.1084
1	0.1173	0.1386
2	0.1340	0.1443
3	0.1403	0.1588
4	0.1652	0.1704
5	0.1677	0.1862
6	0.1905	0.2034
CWRU		
Incremental phases	FT + H	FT + NCC + H
Initial	0.1087	0.1067
1	0.1129	0.1240
2	0.1308	0.1336
3	0.1462	0.1498
4	0.1555	0.1669
5	0.1590	0.1821
6	0.1836	0.2077
7	0.2011	0.2164
8	0.2093	0.2355

7. Conclusions

This study presents a MCIL framework based on fine-tuning with a memory of exemplars and the nearest centroid classifier (NCC) over an 1D convolutional neural network (CNN), to incorporate new motor faults from vibration signals to already known. Specifically, 1D CNN is fine-tuned over samples from new faults and exemplars from known (past) faults, whereas NCC is used during testing phase to classify samples from past and new faults. The proposed framework was evaluated over two datasets for motor fault diagnosis: AMCF and CWRU. Different experimental scenarios were considered, including different numbers of incremental phases and stored exemplars. Experiments showed that the proposed framework achieved an accuracy performance beyond 93% and 94% on AMCF and CWRU, retaining one exemplar per fault and varying the number of incremental phases. We found that 5 and 3 exemplars per fault across 6 and 8 phases on AMCF and CWRU are required to achieve competitive accuracy than training with full data (98.32% vs. 99.38% and 99% vs. 100.00%). These results suggest that the catastrophic forgetting problem can be reduced by the proposed framework over AMCF and CWRU. Another interesting finding is that NCC may help to obtain a robust classifier when noise is presented in data. Using this proposed framework, we showed that a classifier, based on a deep learning model, may be trained incrementally, achieving satisfactory diagnosis results for fault detection in IMs and maintaining a constant complexity of the model. As future work, we are interested in developing an end-to-end MCIL framework, where the feature extractor and the classifier can be trained jointly. Likewise, we are planning to extend our study for the diagnostic of incipient and electrical faults.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no conflicts of interest with respect to the research, authorship, and/or publication of this article.

Acknowledgments

This research was financially supported by National Institute For Astrophysics, Optics, and Electronics.

References

[1] Y. Liang, B. Li, and B. Jiao, “A deep learning method for motor fault diagnosis based on a capsule network with gate-structure dilated convolutions,” *Signal Processing*, vol. 17, no. 18, 2020.

[2] G. Adam, W. Glowacz, J. Kozik et al., “Detection of deterioration of three-phase induction motor using vibration signals,” *Measurement Science Review*, vol. 19, no. 6, pp. 241–249, 2019.

[3] A. Omar, F. Alkahatni, M. Masadeh et al., “Sounds and acoustic emission-based early fault diagnosis of induction motor: a review study,” *Advances in Mechanical Engineering*, vol. 13, no. 2, 2021.

[4] M. M. Rahman and M. N. Uddin, “Online unbalanced rotor fault detection of an im drive based on both time and frequency domain analyses,” *IEEE Transactions on Industry Applications*, vol. 53, no. 4, pp. 4087–4096, 2017.

[5] A. Y. Khodja, N. Guersi, M. N. Saadi, and N. Boutasseta, “Rolling element bearing fault diagnosis for rotating machinery using vibration spectrum imaging and convolutional neural networks,” *International Journal of Advanced Manufacturing Technology*, vol. 106, no. 5, pp. 1737–1751, 2020.

[6] F. Cipollini, L. Oneto, A. Coraddu, and S. Savio, “Unsupervised deep learning for induction motor bearings monitoring,” *Data-Enabled Discovery and Applications*, vol. 3, no. 1, 2019.

[7] J. Wu, T. Tang, M. Chen, Yi Wang, and K. Wang, “A study on adaptation lightweight architecture based deep learning models for bearing fault diagnosis under varying working conditions,” *Expert Systems with Applications*, vol. 160, Article ID 113710, 2020.

[8] J. R. R. Guillen, J. A. B. Hurtado, J. J. D. S. Perez, D. G. Lieberman, and J. P. A. Sanchez, “Convolutional neural network and motor current signature analysis during the transient state for detection of broken rotor bars in induction motors,” *Sensors*, vol. 20, no. 13, 2020.

[9] F. B. Abid, M. Sallem, and A. Braham, “Robust interpretable deep learning for intelligent fault diagnosis of induction motors,” *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 3506–3515, 2020.

[10] X. Li, J. Li, C. Zhao, Y. Qu, and D. He, “Gear pitting fault diagnosis with mixed operating conditions based on adaptive 1d separable convolution with residual connection,” *Mechanical Systems and Signal Processing*, vol. 142, Article ID 106740, 2020.

[11] C. Wu, P. Jiang, C. Ding, F. Feng, and T. Chen, “Intelligent fault diagnosis of rotating machinery based on one-dimensional convolutional neural network,” *Computers in Industry*, vol. 108, pp. 53–61, 2019.

[12] Y. Wang, J. Zhou, L. Zheng, and C. Gogu, “An end-to-end fault diagnostics method based on convolutional neural network for rotating machinery with multiple case studies,” *Journal of Intelligent Manufacturing*, vol. 32, pp. 1–22, 2020.

[13] Y. Liu and Y. Su, “An-an liu, bernt schiele and qianru sun. “Mnemonics training: multi-class incremental learning without forgetting”” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12245–12254, IEEE, Seattle, WA, USA, June 2020.

[14] J. J. S. Dorantes, M. D. Prieto, R. A. O. Rios, and R. D. J. T. Romero, “Industrial data-driven monitoring based on incremental learning applied to the detection of novel faults,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 5985–5995, 2020.

[15] X. Wang, X. Liu, and Y. Li, “An incremental model transfer method for complex process fault diagnosis,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 5, pp. 1268–1280, 2019.

[16] E. Belouadah, A. Popescu, and I. Kanellos, “A comprehensive study of class incremental learning algorithms for visual tasks,” *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 135, pp. 38–54, 2021.

[17] S. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “icarl: incremental classifier and representation learning,” in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, IEEE, Honolulu, HI, USA, July 2017.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[19] H. Zeng, Z. Wu, J. Zhang et al., “Eeg emotion classification using an improved sincnet-based deep learning model,” *Brain Sciences*, vol. 9, no. 11, 2019.

[20] M. Ravanelli and Y. Bengio, “Speaker recognition from raw waveform with sincnet,” in *Proceedings of the 2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 1021–1028, IEEE, Athens, Greece, December 2018.

[21] L. Rabiner and R. Schafer, *Theory and Applications of Digital Speech Processing*, Prentice Hall Press, NJ, USA, 2010.

[22] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, IEEE, Las Vegas, NV, USA, June 2016.

[23] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin, “Learning a unified classifier incrementally via rebalancing,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 831–839, IEEE, Long Beach, CA, USA, June 2019.

[24] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, “Diagnosis of multiple cancer types by shrunken centroids of gene expression,” *Proceedings of the National Academy of Sciences*, vol. 99, no. 10, pp. 6567–6572, 2002.

[25] 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, pp. 100–131, 2015.

[26] D. P. Kingma and B. Jimmy, “Adam: a method for stochastic optimization,” in *Proceedings of the International Conference on Learning Representations*, Cornell University Press, San Diego, California, December 2014.

[27] G. Montavon, B. O Genevieve, and M. K. Robert, “Neural Networks: Tricks Of the Trade,” *Lecture Notes In Computer Science*, Vol. 7700, Springer, Berlin/Heidelberg, Germany, Second edition, 2012.