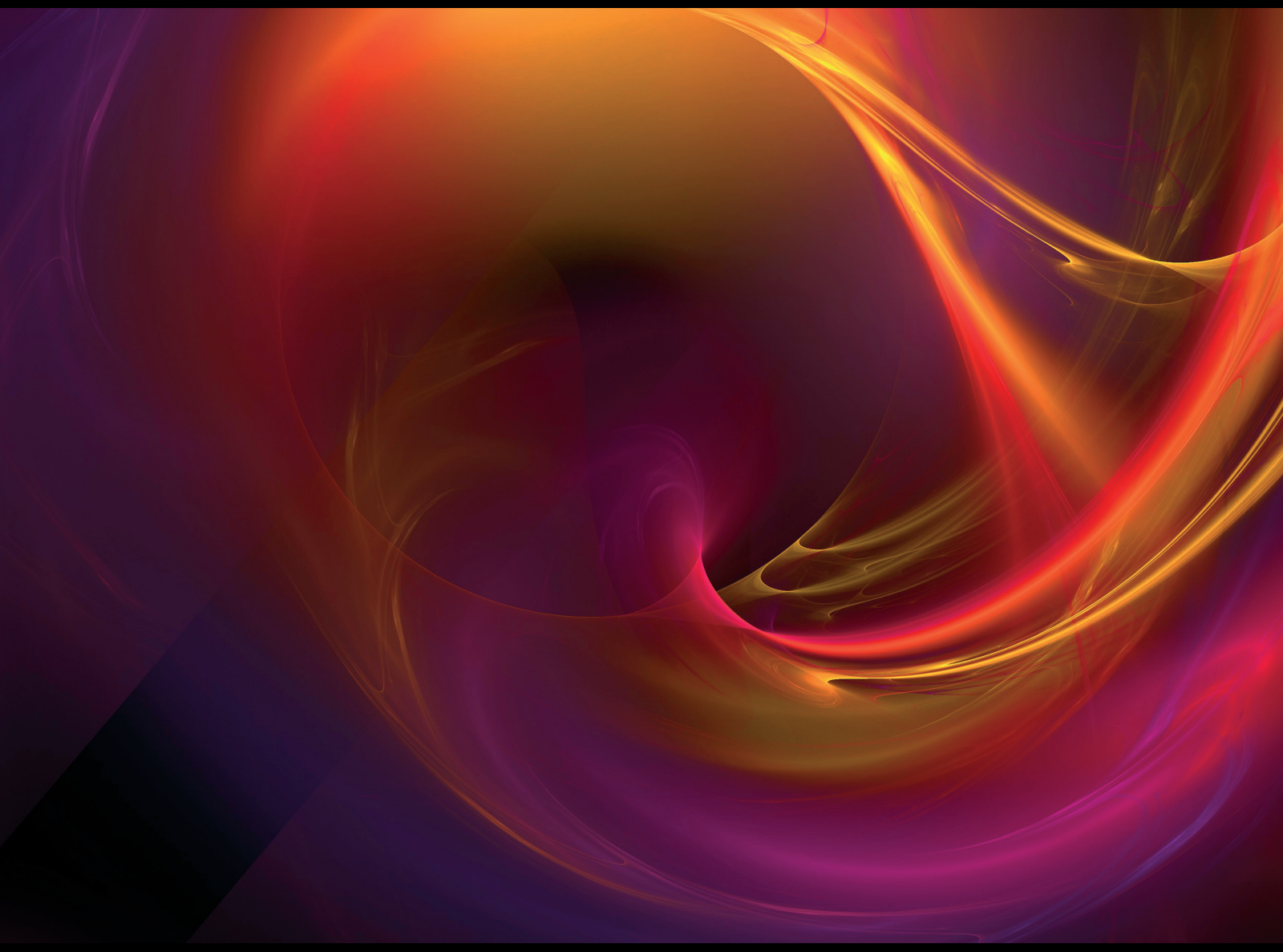
Complexity

Deep Neural Networks for Prognostics and Health Management in Complex and Nonlinear Industrial Systems

Lead Guest Editor: Kiyong Oh
Guest Editors: Taeseong Kim and Minchul Shin



Deep Neural Networks for Prognostics and Health Management in Complex and Nonlinear Industrial Systems

Complexity

Deep Neural Networks for Prognostics and Health Management in Complex and Nonlinear Industrial Systems

Lead Guest Editor: Kiyong Oh
Guest Editors: Taeseong Kim and Minchul Shin

Contents

Hindawi
Complexity
Volume 2022, Article ID 4598725, 15 pages
https://doi.org/10.1155/2022/4598725

WILEY | Hindawi

Research Article

A Diagnosis Framework for High-reliability Equipment with Small Sample Based on Transfer Learning

Jinxin Pan, Bo Jing, Xiaoxuan Jiao, Shenglong Wang, and Qingyi Zhang

[1]*Air Force Engineering University, Xi'an/710038, China*

Correspondence should be addressed to Jinxin Pan; panjinxin_sensor@126.com

Received 17 August 2021; Revised 14 September 2021; Accepted 2 December 2021; Published 23 February 2022

Academic Editor: Kiyong Oh

Conventional methods for fault diagnosis typically require a substantial amount of training data. However, for equipment with high reliability, it is arduous to form a large-scale well-annotated dataset due to the expense of data acquisition and costly annotation. Besides, the generated data have a large number of redundant features which degraded the performance of models. To overcome this, we proposed a feature transfer scenario that transfers knowledge from similar fields to enhance the accuracy of fault diagnosis with small sample. To reduces the redundant information, data were filtered according to manifold consistency. Then, features were extracted based on CNN and feature transfer was conducted. For adequate fitness, the joint adaptation of conditional distribution and marginal distribution was used between the two domains. Minimum structural risk and MMD of adaptation were two indicators weighted for training the model. To test the efficiency of the model, we built an airborne fuel pump testbed, and contributed a new dataset that contained 15 categories of fault data, which serves as the small sample dataset in this research. Then the proposed model was applied in our experimental data. As a result, the fault diagnosis rate increases by 28.6% through our proposed model, which is more precise than other classical methods. The results of feature visualization further demonstrate that the features are more distinguished through the proposed method. All code and data are accessible on my GitHub.

1. Introduction

In recent years, data-driven methods have been widely applied in the fields of prognostics and health management, which have become a hot method in building complex diagnosis models [1–3]. Most data-driven methods are based on a large number of samples, from which the corresponding relationship between input and output is extracted to establish a model. However, for some equipment with high reliability and long lifespan, it is arduous to obtain sufficient fault data. Along with that, the generated data have a large number of redundant features which degraded the performance of models. A model trained by small samples has limited generalization ability, which will lead to low accuracy when applied to other fields [4].

Some popular methods have been proposed to solve the problem of small sample. To our best knowledge, these methods are of three categories. The first is based on resample, which resampling small samples to generate more data, such as Random Under-sampling [5] and Synthetic Minority Over-sampling [6]. The second is based on Generative Adversarial Net [7, 8], whose principle is to make the generated network sample as realistic as possible by antagonizing generative network and discriminant network, so as to enlarge small samples. The third is based on few-shot learning [9, 10], which decompose the small samples into different meta tasks to learn the generalization ability of the model. Therefore, few-shot learning has adaptive capacity to an unseen dataset. Although these methods achieved success to some extent, their sources of knowledge are only from the small samples. From the perspective of information theory, these methods do not change the nature of a small sample with little knowledge.

Transfer learning is a new machine learning method that uses existing knowledge to solve problems in different but related fields [11]. Transfer learning has been applied to some tasks such as hand gesture recognition [12], sentiment analysis [13], fraud detection [14], and hyperspectral image analysis [15]. Besides, many advanced transfer learning theories have been proposed. For example, Liu [16]

proposed a one-step approach towards classifiers have to be trained with noisy data. Chen [17] proposed a boundary based Out-of-Distribution (OOD) classifier which classifies the unseen and seen domains by only using seen samples for training. Teshima [18] proposed a meta-distributional scenario in which a data generating mechanism is invariant among domains.

Due to the advantages of transfer learning in domain generalization, it is also widely applied in fault diagnosis. Wang [19] proposed an LDA-based deep transfer learning framework for fault diagnosis in industrial chemical processes, Singh [20] utilized minimum redundancy maximum relevance (mRMR) for intelligent fault diagnosis of rotating machines. Deng [21] proposed a double-layer attention based adversarial network (DA-GAN) for partial transfer learning in machinery fault diagnosis. To draw a conclusion from the existing literatures, there are two aspects that few researchers had considered. Firstly, most proposed transfer learning methods focus on the adaptation of the diagnosis algorithm in different fields but pay less attention to the condition of small samples. Secondly, most literatures use the raw signal in the source domain, which is feasible for many fields. However, some high-reliability equipment has a long lifespan, which results in redundant features of monitoring data. If the raw signal is used directly, negative migration may occur.

Given that high-reliability equipment has characteristics of small sample size and redundant features, we proposed a feature transfer framework. To reduces the redundant information, data were filtered according to manifold consistency. Then, features were extracted based on CNN and feature transfer was conducted. For adequate fitness, the joint adaptation of conditional distribution and marginal distribution was used between the two domains. Minimum structural risk and MMD of adaptation were two indicators weighted for training the model to enhance the generalization ability of the model. To test the efficiency of the model, we built an airborne fuel pump testbed, and contributed a new dataset that contained 15 categories of fault data, which serves as the small sample dataset in this research. Then the proposed model was applied in our experimental data. As a result, the fault diagnosis rate increases by 28.6% through our proposed model, which is more precise than other classical methods. The results of feature visualization further demonstrate that the features are more distinguished through the proposed method.

2. Problem Setup

2.1. Definition of transfer learning. In most cases, a domain D consists of two components: a feature space χ and a marginal probability distribution $P(X)$, where $X = \{x_1, ..., x_n\} \in \chi$. For example, given a concrete domain $D = \{X, P(X)\}$, a task can be expressed as two components: a label space Υ and an objective predictive function $f(\cdot)$ (denoted by$T = \{Y, f(\cdot)\}$), which is not observed but can be trained from the data, which consist of pairs $\{x_i, y_i\}$, where $x_i \in \chi$ and$y_i \in \Upsilon$. The function $f(\cdot)$ can be used to predict from the data to the corresponding label$f(x)$. From a probabilistic viewpoint, $f(x)$ can be written as$P(y \mid x)$.

Definition of transfer learning: Given a source domain D^s and learning taskT^s, a target domain D^t and learning taskT^t, transfer learning aims to help improve the learning of the target predictive function $f^t(\cdot)$ inD^t using the knowledge in D^s andT^s, where $D^s \neq D^t$, or $T^s \neq T^t$[22].

2.2. Fault diagnosis with small sample. For some highly reliable products, the small sample data set can be represented as $\{x_i^t, y_i^t\}_{i=1}^{n_t}$, which contains n_t samples. In the form of transfer learning, the data set is described as target domain $D^t = \{x^t, P(x^t)\}$ and the target task $T^t = \{y^t, f^t(\cdot)\}$, where $P(x^t)$ is the Marginal Distribution of x^t, y^t is the label space of the target domain, $f^t(\cdot)$ is a function that maps the sample x^t to the tag space y^t in the target domain.

Another dataset with rich samples is represented as $\{x_i^s, y_i^s\}_{i=1}^{n_s}$, which contains n_s samples ($n_s \gg n_t$). In the form of transfer learning, the data set is described as the source domain $D^s = \{x^s, P(x^s)\}$ and the source task $T^s = \{y^s, f^s(\cdot)\}$ in transfer learning, where $P(x^s)$ is the marginal distribution of x^s, y^s is the label space of the source domain, $f^s(.)$ is a function that maps the sample x^s to the tag space y^s in the source domain.

The goal of transfer learning is to acquire and apply the knowledge from the source domain [23]. More specifically, it is to establish the nonlinear mapping relationship from the equipment monitoring data to the health label space in the source domain, then transfer it to the target domain. In the given situation, the labels of the source domain and target domain are accessible. Such problem is concluded as multitask learning. Aiming at this problem, feature transfer is often used to transfer knowledge from source domain to target domain [22].

2.3. Feature transfer. The idea of feature transfer is to learn a pair of mapping functions $\{\varphi^s(\cdot), \varphi^t(\cdot)\}$ to extract diagnostic features respectively from the source domain and the target domain. Then adapt features extracted by mapping functions, and the target domain could extract features according to the paradigm of the source domain. The fault classification is carried out in the target domain based on the features, and the diagnosis knowledge of the source domain is transferred to the target domain through the feature adaptation [24].

Aiming for learning transferable features between a given target domain and source domain, a common approach of feature adaptation is to minimize inter-domain differences between source feature and target feature. During the adaptation, the extracted features from the target domain act as a template, that the source domain could learn from the template. The schematic diagram of domain adaptation is shown in Figure 1.

3. General Framework

The main issues of transfer learning can be concluded as the following three: when to transfer, what to transfer and how to transfer. Aiming at these three issues, this paper designed a framework of LLE-CNN-JDA. Aiming at when to transfer, we designed a data filtering method based on manifold

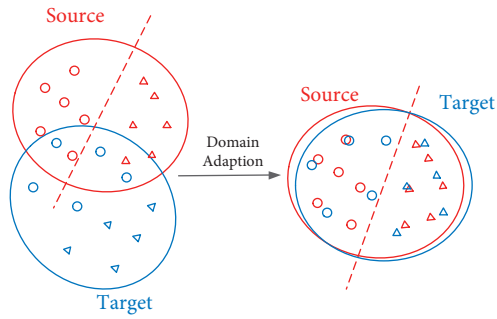


Figure 1: Schematic diagram of domain adaptation.

consistency. We mapped the high-dimensional data into low dimensional space, to analyze the similarity between the source domain and target domain. To ensure the availability of transfer data, we filtered data in the source domain based on the Euclidean distance between the source domain and the target domain in manifold space. Aiming at what to transfer, the method of feature transfer is adopted, the convolutional neural network is used to extract the deep feature of the data in the source domain and target domain. By adapting the feature in the source domain and target domain, the knowledge may be transferred in the feature layer. Aiming at how to transfer, we designed a term that jointly adapt conditional distribution and marginal distribution of feature layer, and the network is trained based on structural risk minimization. The framework diagram of the proposed model is shown in Figure 2.

3.1. Data filtered based on manifold consistency. The so-called manifold is a general term for geometric objects, such as curves and surfaces of various dimensions. Manifold learning maps data from higher-dimensional space into lower-dimensional space. Unlike other dimensionality reduction methods, manifold learning assumes that data is sampled from a potential manifold. If we can find the laws of the data in the manifold space, we may find the potential laws of the data in high dimensions to mine the essential characteristics of data [25, 26].

For the given situation, the sample size of the source domain is large, so the prognostics model in the source domain can be trained well. However, the data in the target domain is not abundant, we may not excavate enough information from the target domain. The source domain has sufficient data, but it is necessary to explore whether the source data can be effectively applied to the target domain. Based on the idea of manifold learning, we consider that if we can explore the relationship between the source domain and the target domain in the low-dimensional manifold space, and filter the data transferable to the target domain, source data may be applicable to the target domain.

Locally linear embedding (LLE) is a method of manifold learning, which enables the data to maintain the original manifold structure well after dimension-reduction. The manifold of LLE is an unclosed surface, which has features of relatively uniform and dense distribution. Every data point can be constructed by the linear weighted combination of its nearest points. LLE transfers manifolds from higher dimensions to lower dimensions and preserves some features of manifolds in higher dimensions as much as possible [27, 28]. The steps of the LLE algorithm are as follows:

Step1. :Calculate K adjacent points for each sample point. Adopting the KNN strategy, K points with the smallest Euclidean distance to the sample point are taken as the K adjacent points of the sample.

Step2. :Calculate the local reconstruction weight matrix W of the sample. The reconstruction error is defined as $\varepsilon(W) = \sum_i |\overrightarrow{X_i} - \sum_j W_{ij}\overrightarrow{X_j}|^2$, and the local covariance matrix C is defined as $C_{jk} = (\overrightarrow{x} - \overrightarrow{\eta}_j) \cdot (\overrightarrow{x} - \overrightarrow{\eta}_k)$, where X represents a specific point, and its K adjacent points are denoted by η. Then Minimize $\varepsilon(W)$, then get $W_j = \sum_k C_{jk}^{-1} / \sum_{lm} C_{lm}^{-1}$.

Step3. :Map all sample points to a low-dimensional space, where the mapping function satisfies $\min_Y \Phi(Y) = \sum_i \overrightarrow{Y_i} \sum_j | - \sum_j W_{ij}\overrightarrow{Y_j}|^2$. The formula can also be represented as $\Phi(Y) = \sum_{ij} M_{ij}(\overrightarrow{Y_i} \cdot \overrightarrow{Y_j})$, where $M = (I - W)^T(I - W)$. Combining with the restrictive conditions $\sum_i Y = \overrightarrow{0}$ and $1/N\sum_i Y_i \overrightarrow{Y_i}^T = I$, the problem is transformed into MY = λY, take M eigenvectors of matrix M to form column vectors, that matrix Y = N ∗ M, where N is the size of data.

In this paper, we adopted the LLE algorithm to evaluate the similarity of the data from the source domain and target domain. We adopted the bearing failure data from Case Western Reserve University as the source domain, the failure data from our airborne fuel pump test-bed as the target domain. Two kinds of data were mapping to manifold space respectively. As the neighbouring points number K change, the mapping results in manifold space are shown in Figure 3. Through analysis, we found that when K = 100, the mapping manifold of the two types of data is most close. Therefore, the K = 100 was chosen to filter data in the low dimension. The failure data from the airborne fuel pump testbed worked as the target domain, which was in a small sample size. The bearing failure data from Case Western Reserve University worked as the source domain. The bearing failure data is in big sample size, but it is difficult to ensure the validity of the data. Therefore, we proposed to filter data of the source domain in the manifold space by calculating the Euclidean distance to the target domain. We adopted the tactics of KNN, computed the distance of each sample point from the source domain to all sample points from the target domain, then chose the minimum distance $d_i = \min|X_i^s - X_j^t|\,(j = 1 \cdots n_t)$ as an indicator of the sample point. We got an indicator set $\{d_i\}(i = 1 \cdots n_s)$, then sorted the indicator set, and chose the **n** sample points with the smallest distance as the target domain.

3.2. Deep feature extraction. The source domain and target domain data both contain prognostics information related to the equipment. Therefore, we adopt a convolutional

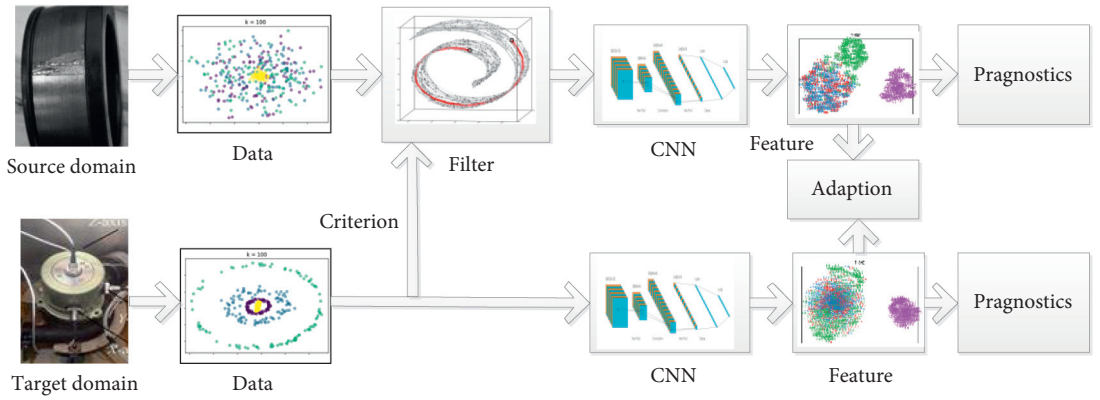


Figure 2: The framework of the proposed model.

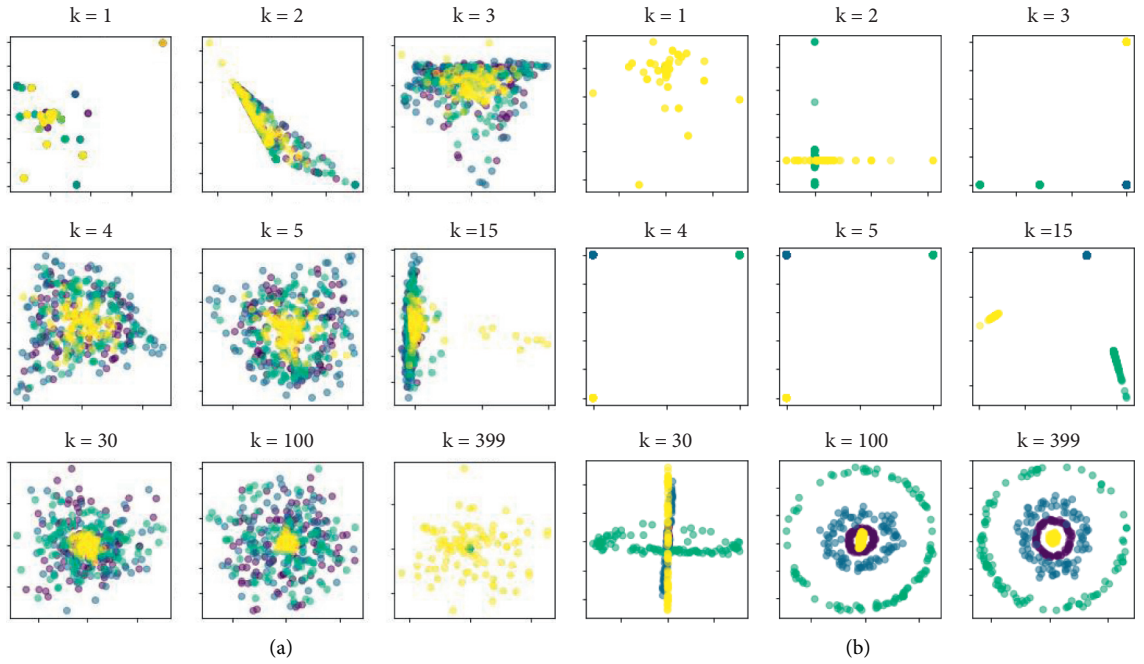


Figure 3: Results of LLE under the different value of K; (a) LLE of bearing fault data; (b) LLE of airborne fuel pump fault data.

neural network to extract fault features. The source domain data and target domain data were extracted separately by the neural network, which contains convolution layer, pooling layer, flatten layer and full connection layer. The network parameters are trained based on structural risk minimization and feature adaptation between the source domain and target domain.

Convolution layer operation: Assume that $x_i^{s,l-1}$ and $x_i^{t,l-1}$ represent vectors in the L-1 layer of the network. Through a convolution kernel function $k \in R^{16\times3}$, the feature vector of layer L is $x_i^{s,l} = F(x_i^{s,l-1}; \theta^l) = \sigma_l(x_i^{s,l-1} * k^l + b^l)$, where σ_l is the activation function of layer L, $\theta^l = \{k^l, b^l\}$ is the parameters to be trained of the convolution layer.

Pooling layer operation: The processing logic of the pooling layer is to compress the input matrix. The formula of the pooling layer is $v_{m,n}^{s,l} = \max\{x_i^{s,l} | m \le i \le n\}$, where $v_{m,n}^{s,l}$ represents the result of pooling the features from sequence M to N in the convolution layer L, $x_i^{s,l}$ represents the vector whose sequence is i in the convolution layer L, s represents that the operation is for the source domain.

After multiple convolution and pooling processes, the feature extraction layer output, namely the flatten layer input vector $x_i^{s,C}$, is obtained. The vector fed into flatten function, and the flatten layer output $x_i^{s,F} = \text{flatten}(x_i^{s,C})$ was obtained. The output of the flattening layer is then used as the input vector of the full connection layer, where the vector is mapped to the label space through the neural network of the full connection layer. The function is $y_i^{s,o} = g(x_i^{s,F}; \theta^F) = \sigma(w^F \cdot x_i^{s,F} + b^F)$, where σ is the activation function, $\theta^F = \{w^F, b^F\}$ is the parameter set to be trained.

3.3. Structural risk minimization. For the target domain data with a small size, overfitting is easy to occur in the training process. To prevent overfitting, it is necessary to avoid the excessive complexity of network structure in the process of

training. Therefore, complexity and accuracy are important indicators impacting the efficiency of the network. In this paper, the Convolutional Neural Network was trained based on structural risk minimization. A penalty term (regularization term) for the complexity of the model is added to the empirical risk to reduce the risk of data overfitting [29]. The formula of structural risk minimization is expressed as follows:

$$\min_{f\in F} \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f\left(x_i\right)\right) + \lambda J\left(f\right). \tag{1}$$

In the formula, the more complex the model f is, the greater the value of $J(f)$ will be. λ indicates the importance of model complexity. As the empirical risk convergence to a certain degree, when the empirical risk decreases, model complexity will increase sharply. Model complexity may make the model fit the data in the source domain over exactly, and the model would be difficult to generalize to the target domain. Thus, we added the penalty term for model complexity to inhibits the excessive increase of model complexity.

For a conditional probability distribution, the loss function is logarithmic, and the model complexity is determined by the prior probability of the model. Therefore, the structural risk minimization is equal to the maximum posterior probability estimate. Given a sample set $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$, it is assumed that the prior distribution of parameter θ is$g(\theta)$, and the probability of T is$\prod_{i=1}^{N} P(y_i \mid x_i; \theta)g(\theta)$. Maximize the probability as $\max_{\theta} \prod_{i=1}^{N} P(y_i \mid x_i; \theta)g(\theta)$, then take the log of the result, $\max_{\theta}\left\{\sum_{i=1}^{N} \log P(y_i \mid x_i; \theta) + \log g(\theta)\right\}$ will be obtained. Take the complex number of the above equation, it is transformed into the minimization problem $\min_{\theta}\left\{\sum_{i=1}^{N} -\log P(y_i \mid x_i; \theta) + \log 1/g(\theta)\right\}$. Define the loss function as $L(x_i, P(y_i \mid x_i; \theta)) = -\log P(y_i \mid x_i; \theta)$, the coefficient as $\lambda = 1/N$, the penalty term as $J(f) = \log 1/g(\theta)$, the equation is equal to form (2), which is structural risk minimization [30].

$$\min_{\theta}\left\{\frac{1}{N}\sum_{i=1}^{N} L\left(y_i, P\left(y_i \mid x_i; \theta\right)\right) + \lambda J\left(f\right)\right\}. \tag{2}$$

3.4. Joint adaption of conditional distribution and marginal distribution. Feature extraction and network training are conducted separately in the source domain and target domain. To transfer knowledge, the adaptation between the source domain and the target domain is to be conducted at the feature layer.

As for the source domain, the size of data is large, so the extracted feature contains a lot of information related to the equipment. Thus, the pattern how deep features be extracted in source domain is an empirical model for fault identifying, which may be appliable to other related fields. For the target domain, the sample size is small, which may lead to poor generalization ability of the network in feature extraction. Therefore, the feature extraction of the source domain is a reference to the target domain, and feature transfer will be available in this way.

Marginal distribution and conditional distribution reflect domain distribution [31]. Therefore, feature adaptation is to adapt marginal distribution and conditional distribution. According to probability theory $J = P \cdot Q$, we seek to minimize the distribution distance (1). between the marginal distributions P^s andP^t, and (2). between the conditional distributions Q^s andQ^t simultaneously. For source domain $D^s = \{X^s, P(X^s)\}$ and target domain $D^t = \{X^t, P(X^t)\}$, assume that the features extracted through CNN network are $x_i^{s,C}$ and $x_i^{t,C}$. If the features of the two domains are to be adapted, the marginal distribution and conditional distribution of feature vectors must be adapted.

3.4.1. Adaption of the marginal distribution. We try to minimize the distance between marginal distributions P^s andP^t. Since directly estimating probability densities is nontrivial, we resort to explore nonparametric statistics. We adopt empirical Maximum Mean Discrepancy (MMD) to measure the distance, which compares different distributions based on the distance between the sample means of two domains in a reproducing kernel Hilbert space (RKHS) [32].

Specifically, the statistical approach of MMD is conducted in the following manner. Based on the samples of the two distributions, look for a continuous function $f(\cdot)$ in the sample space, get the function values corresponding to the two distributions, and calculate the mean of the function values of each distribution. By making difference between the two mean values, the mean discrepancy of the two distributions will be obtained corresponding to$f(\cdot)$. Look for an $f(\cdot)$ that causes the mean discrepancy to have a maximum value, the value is MMD. Thus, MMD is taken as a test statistic to determine whether the two distributions were close. If this value is small enough, the two distributions are considered the same; otherwise, they are not. This value is also used to determine the degree of similarity between two distributions [33]. If F is used to represent a continuous set of functions in the sample space, then MMD can be expressed as follows:

$$\mathrm{MMD}[F, p, q] = \sup_{f\in F}\left(E_{x\sim p}[f(x)] - E_{y\sim p}[f(y)]\right). \tag{3}$$

Assume that X and Y are two data sets obtained by independent identical distribution sampling from distribution p and q respectively, and the sizes of the data sets are M and N. The empirical estimate of MMD based on X and Y is as follows:

$$\mathrm{MMD}[F, p, q] = \sup_{f\in F}\left(\frac{1}{m}\sum_{i=1}^{m} f\left(x_i\right) - \frac{1}{n}\sum_{i=1}^{n} f\left(y_i\right)\right). \tag{4}$$

Given two distributions of observation sets X, Y, this result will depend heavily on a given set of functions F. To express the properties of MMD, if and only if P and Q are of the same distribution, MMD is 0. Thus, F is required to be rich enough. On the other hand, the empirical estimation of MMD should rapidly converge to its expectation as the size

of the observation set increases. Thus, F must be sufficiently restrictive. To satisfy the two requirements, we adopt the reproducing kernel Hilbert Spaces.

In reproducing kernel Hilbert Spaces, F space is a complete inner product space, and each F corresponds to a feature map. Based on the feature map, we defined a mean embedding of p for a distribution p that satisfies the following properties: $\mu_p \in H$ such that $E_x(f) = \langle f, \mu_p \rangle_H$ for all $f \in H$. Mean embedding exists with constraints. Under the existence of the mean embedding of P and Q, the MMD squared can be expressed as follows:

$$\begin{aligned} \mathrm{MMD}^2[F, p, q] &= \left[\sup_{\|f\|_H \le 1} \left(E_x[f(x)] - E_y[f(y)] \right) \right]^2 \\ &= \left\| \mu_p - \mu_q \right\|_H^2. \end{aligned} \tag{5}$$

If F is a unit ball in a universal RKHS, such as Gaussian and Laplace RKHSs, the square of this MMD can be expressed as:

$$\begin{aligned} \mathrm{MMD}^2[F, p, q] = E_{x,x'}[k(x, x')] - 2E_{x,y}[k(x, y)] \\ + E_{y,y'}[k(y, y')]. \end{aligned} \tag{6}$$

X and X 'are two random variables that obey p, and y and y' are two random variables that obey q. One of the above statistical estimates can be expressed as:

$$\mathrm{MMD}[F, X, Y] = \left[\frac{1}{m^2} \sum_{i,j=1}^{m} k(x_i, x_j) - \frac{2}{\mathrm{mn}} \sum_{i,j=1}^{m,n} k(x_i, y_j) + \frac{1}{n^2} \sum_{i,j=1}^{n} k(y_i, y_j) \right]^{1/2}. \tag{7}$$

Marginal distribution adaptation is to maximize mean discrepancies that for minimizing the marginal distributions of the features from the source domain and target domain [31], which is:

$$\min \mathrm{MMD}\left(x_i^{s,C}, x_i^{t,C}\right). \tag{8}$$

3.4.2. Adaption of the conditional distribution. We try to minimize the distance between conditional distributions Q^s andQ^t. Since calculating the nonparametric statistics of $Q^s(y^s|x^s)$ and$Q^t(y^t|x^t)$ are difficult, we resort to explore the quasi-conditional distributions as $Q^s(x^s|y^s)$ and $Q^t(x^t|y^t)$ instead, which can well approximate $Q^s(y^s|x^s)$ and$Q^t(y^t|x^t)$ when sample sizes are large [34, 35].

Conditional distribution adaptation is to maximize mean discrepancies. Simultaneously, the quasi-conditional distributions of the features from the source domain and target domain will reach a minimum, namely:

$$\min \mathrm{MMD}\left(x_i^{s,C}|y_k, x_i^{t,C}|y_k\right). \tag{9}$$

4. Training of the network

Overall, manifold consistency was used to filter data from the source domain. The deep features of the source domain and the target domain were extracted by CNN, then the features were adapted, and the classifier was constructed through the full connection layer. After network construction, network parameters need to be trained in the following aspects: (1). structural risk minimization; (2). marginal distribution adaptation; (3). conditional distribution adaptation. The network structure of the proposed model is shown in Figure 4. The joint optimization formula is as follows:

$$\begin{aligned} \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L\left(y_i^t, f\left(x_i^t\right)\right) + \lambda_t J\left(f^t\right) + \alpha \frac{1}{N} \sum_{i=1}^{N} L\left(y_i^s, f\left(x_i^s\right)\right) \\ + \lambda_s J\left(f^s\right), \\ + \beta \cdot \sum_{i=1}^{N} \mathrm{MMD}\left(x_i^{s,C}, x_i^{t,C}\right) + \delta \cdot \sum_{i=1,k=1}^{N,K} \mathrm{MMD} \\ \cdot \left(x_i^{s,C}|y_k, x_i^{t,C}|y_k\right). \end{aligned} \tag{10}$$

Due to the addition of the adaptive function in the feature layer, the parameter training and backpropagation of the entire convolutional neural network will be affected. For a single fault diagnosis with convolutional neural network, the error is from the difference between the expected label and the real label. However, when the joint distribution adaptation function is added to the feature layer, the adaptation error of the feature layer will also affect the parameter training of the whole network. Therefore, to search how the network is trained, the error backpropagation formula of the network is derived in this paper.

4.1. Error backpropagation in the full connection layer. The error of the output layer comes from the difference between the expected label and the real label, which is usually expressed by the two-norm of the difference between the two labels. The formula is as follows:

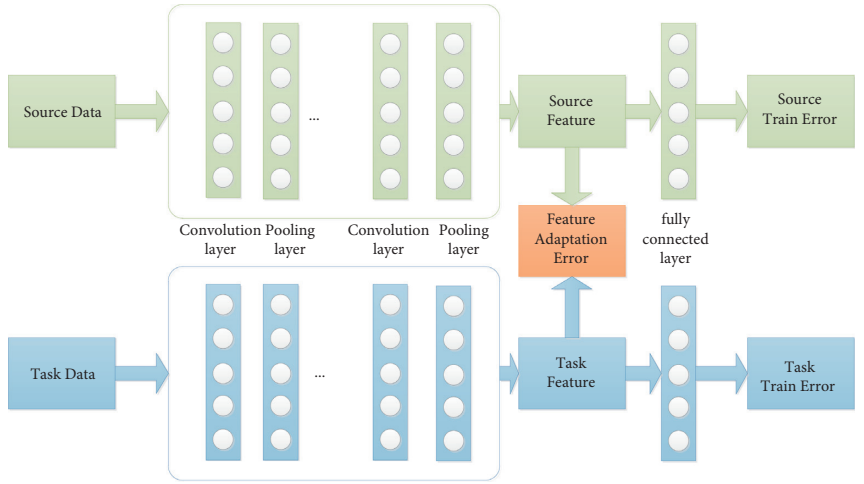


Figure 4: The network structure of the proposed model.

$$J(W,b,x,y)=\frac{1}{2}\left\|a^L-y\right\|_2^2, \tag{11}$$

Where a^L is output of the network, y is the label of the data set. According to the functional relationship, a^L can be expressed as: $a^L=\sigma(z^L)=\sigma(W^La^{L-1}+b^L)$, so we can get:

$$\begin{aligned}\frac{\partial J(W,b,x,y)}{\partial W^L}&=\left[\left(a^L-y\right)\odot\sigma'\left(z^L\right)\right]\left(a^{L-1}\right)^T,\\ \frac{\partial J(W,b,x,y)}{\partial b^L}&=\left[\left(a^L-y\right)\odot\sigma'\left(z^L\right)\right],\end{aligned} \tag{12}$$

Where $\odot$ is Hadamard product, σ is activation function, W^L and b^L are the weights and bias of the output layer.

In this paper, a single-layer full connection layer is used as a feature classifier. To enhance the extensibility of the algorithm, a more general case of multi-layer full connection is considered in the derivation of the backpropagation formula. Suppose an error propagation variable is:

$$\begin{aligned}\delta^l&=\frac{\partial J(W,b,x,y)}{\partial z^l}\\&=\left(\frac{\partial z^L}{\partial z^{L-1}}\frac{\partial z^{L-1}}{\partial z^{L-2}}\cdots\frac{\partial z^{l+1}}{\partial z^l}\right)^T\frac{\partial J(W,b,x,y)}{\partial z^L}.\end{aligned} \tag{13}$$

If we can figure out δ^l , according to the formula of layer l: $z^l=W^la^{l-1}+b^l$, the gradient formula of layer l can be obtained as follows:

$$\begin{aligned}\frac{\partial J(W,b,x,y)}{\partial W^l}&=\delta^l\left(a^{l-1}\right)^T,\\ \frac{\partial J(W,b,x,y)}{\partial b^l}&=\delta^l.\end{aligned} \tag{14}$$

So, the whole point of the problem is to figure out δ^l. In this paper, δ^l was deduced by mathematical induction. As for the output layer, $\delta^L=\partial J(W,b,x,y)/\partial z^L=(a^L-y)\odot\sigma'(z^L)$. For layer l, δ^l can be figured out throughδ^{l+1} from layer$l+1$, the formula is as follows:

$$\begin{aligned}\delta^l&=\left(\frac{\partial z^{l+1}}{\partial z^l}\right)^T\frac{\partial J(W,b,x,y)}{\partial z^{l+1}},\\&=\operatorname{diag}\left(\sigma'\left(z^l\right)\right)\left(W^{l+1}\right)^T\delta^{l+1},\\&=\left(W^{l+1}\right)^T\delta^{l+1}\odot\sigma'\left(z^l\right).\end{aligned} \tag{15}$$

Therefore, δ^l of each layer can be obtained by continuous backward recursion from the output layer, and then the updated formula of weight and bias of each layer can be calculated as:

$$\begin{aligned}W^l&=W^l-\alpha\sum_{i=1}^m\delta^{i,l}\left(a^{i,l-1}\right)^T,\\ b^l&=b^l-\alpha\sum_{i=1}^m\delta^{i,l}.\end{aligned} \tag{16}$$

4.2. Error backpropagation in pooling layer. There is no need to optimize and update W and B in the pooling process. However, in the process of backpropagation, the error will change in the pooling layer. Similar to the full connection layer, we still use δ^l as the bridge to calculate the backpropagation formula for the pooled layer.

For the pooling layer, during the backpropagation, we firstly restore all of the submatrix matrix sizes of δ^l to their pre-pooling sizes. If the pooling operation is Max, the values of each pooled locality of all submatrices of δ^l are placed at the position where the previous forward propagation algorithm obtained the maximum value. If the pooling operation is Average, then the values of each pooling locality of all the submatrices of δ^l are averaged and placed at the reduced submatrix position. This process is usually called the Upsample, and the formula is as follows:

$$\delta^l=\left(\frac{\partial a^l}{\partial z^l}\right)^T\frac{\partial J(W,b)}{\partial a^l}=\operatorname{upsample}\left(\delta^{l+1}\right)\odot\sigma'\left(z^l\right), \tag{17}$$

Where the Upsample function completes the logic of enlarging the pool error matrix and redistributing the error.

4.3. Error backpropagation in the convolution layer. Based on the analysis of the full connection layer, we can get that the recursion relation of δ^l in the convolution layer is:

$$\begin{aligned}\delta^l &= \frac{\partial J(W,b,x,y)}{\partial z^l} = \left(\frac{\partial z^{l+1}}{\partial z^l}\right)^T \frac{\partial J(W,b,x,y)}{\partial z^{l+1}} \\ &= \left(\frac{\partial z^{l+1}}{\partial z^l}\right)^T \delta^{l+1}.\end{aligned} \tag{18}$$

The key to the problem is to solve for $\partial z^{l+1}/\partial z^l$. According to the matrix relation: $z^{l+1} = a^l \cdot W^{l+1} + b^{l+1} = \sigma(z^l) \cdot W^{l+1} + b^{l+1}$, it can be obtained that: $\delta^l = (\partial z^{l+1}/\partial z^l)^T \delta^{l+1} = \delta^{l+1} \cdot \mathrm{rot180}(W^{l+1}) \odot \sigma'(z^l)$. According to the matrix relation: $z^l = a^{l-1} \cdot W^l + b^l$, the gradient of W and b can be obtained as follows:

$$\begin{aligned}\frac{\partial J(W,b,x,y)}{\partial W^l} &= a^{l-1} \cdot \delta^l, \\ \frac{\partial J(W,b,x,y)}{\partial b^l} &= \sum_{u,v} \left(\delta^l\right)_{u,v}.\end{aligned} \tag{19}$$

Thus, the updated formula of weight and bias of each layer can be calculated as follows:

$$\begin{aligned}W^l &= W^l - \alpha \sum_{i=1}^{m} a^{i,l-1} \cdot \delta^{i,l}, \\ b^l &= b^l - \alpha \sum_{i=1}^{m} \sum_{u,v} \left(\delta^{i,l}\right)_{u,v}.\end{aligned} \tag{20}$$

4.4. Error backpropagation in feature adaption layer. For the model, the error of the feature adaption layer will affect both the source domain and the target domain. Taking the target domain as an example, we calculated the gradient update error of the feature adaption layer.

In the feature adaption layer, the output is set as a_t^l. In the process of backpropagation, the error of this layer has two sources. One is the error gradient δ^{l+1} returned by the next layer over the network, and the other is the maximize mean discrepancies between the featurea_t^l and a_s^l. As for the feature adaption layer, the error can be expressed as:

$$\begin{aligned}J_t^l(W,b,x,y,\mathrm{MMD}) &= \frac{1}{2}\left\|a_t^l - y\right\|_2^2 + \gamma \mathrm{MMD}\left(a_t^l, a_s^l\right) \\ &\quad + \eta \sum_{k=1}^{K} \mathrm{MMD}\left(a_t^l | y_k, a_s^l | y_k\right).\end{aligned} \tag{21}$$

The gradient of W and b can be obtained as follows:

$$\begin{aligned}\frac{J_t^l(W,b,x,y,\mathrm{MMD})}{\partial W^l} &= \delta^l \left(a^{l-1}\right)^T + \gamma \frac{\partial \mathrm{MMD}\left(a_t^l, a_s^l\right)}{\partial W^l} \\ &\quad + \eta \sum_{k=1}^{K} \frac{\mathrm{MMD}\left(a_t^l | y_k, a_s^l | y_k\right)}{\partial W^l}, \\ \frac{J_t^l(W,b,x,y,\mathrm{MMD})}{\partial b^l} &= \delta^l + \gamma \frac{\partial \mathrm{MMD}\left(a_t^l, a_s^l\right)}{\partial b^l} \\ &\quad + \eta \sum_{k=1}^{K} \frac{\mathrm{MMD}\left(a_t^l | y_k, a_s^l | y_k\right)}{\partial b^l},\end{aligned} \tag{22}$$

Thus, the updated formula of weight and bias of each layer can be calculated as follows:

$$\begin{aligned}W^l &= W^l - \alpha \sum_{i=1}^{m} \left[a^{i,l-1} \cdot \delta^{i,l} + \gamma \frac{\partial \mathrm{MMD}\left(a_t^{i,l}, a_s^{i,l}\right)}{\partial W^{i,l}} \right. \\ &\quad \left. + \eta \sum_{k=1}^{K} \frac{\mathrm{MMD}\left(a_t^{i,l} | y_k, a_s^{i,l} | y_k\right)}{\partial W^{i,l}}\right], \\ b^l &= b^l - \alpha \sum_{i=1}^{m} \left[\sum_{u,v} \left(\delta^{i,l}\right)_{u,v} + \gamma \frac{\partial \mathrm{MMD}\left(a_t^{i,l}, a_s^{i,l}\right)}{\partial b^{i,l}} \right. \\ &\quad \left. + \eta \sum_{k=1}^{K} \frac{\mathrm{MMD}\left(a_t^{i,l} | y_k, a_s^{i,l} | y_k\right)}{\partial b^{i,l}}\right].\end{aligned} \tag{23}$$

After updating W and b of the feature adaption layer, it is also necessary to deduce the transfer of the errors of this layer to the previous layer. Referring to the backpropagation methods of the convolution layer, the error propagation variable of the feature adaption layer is defined as:

$$\begin{aligned}\delta^l &= \frac{\partial J(W,b,x,y,\mathrm{MMD})}{\partial z^l} \\ &= \left(\frac{\partial z^{l+1}}{\partial z^l}\right)^T \frac{\partial J(W,b,x,y,\mathrm{MMD})}{\partial z^{l+1}}.\end{aligned} \tag{24}$$

According to the matrix relation between two layers, we can figure out:

$$\begin{aligned}\frac{\partial z^{l+1}}{\partial z^l} &= \left(W^{l+1}\right)\mathrm{diag}\left(\sigma'\left(z^l\right)\right), \\ \frac{\partial J(W,b,x,y,\mathrm{MMD})}{\partial z^{l+1}} &= \delta^{l+1} + \gamma \frac{\partial \mathrm{MMD}\left(a_t^l, a_s^l\right)}{\partial z^{l+1}} \\ &\quad + \eta \sum_{k=1}^{K} \frac{\mathrm{MMD}\left(a_t^l | y_k, a_s^l | y_k\right)}{\partial z^{l+1}}.\end{aligned} \tag{25}$$

Thus, the recursive relation of the feature adaption layer can be obtained as follows:

$$\delta^l = \operatorname{diag}\left(\sigma'\left(z^l\right)\right)\left(W^{l+1}\right)^T \cdot \left[\delta^{l+1} + \gamma \frac{\partial \mathrm{MMD}\left(a_t^l, a_s^l\right)}{\partial z^{l+1}} + \eta \sum_{k=1}^{K} \frac{\mathrm{MMD}\left(a_t^l | y_k, a_s^l | y_k\right)}{\partial z^{l+1}}\right]. \tag{26}$$

5. Construction of testbed and application of the model

To verify the effectiveness of the method, an airborne fuel pump test platform was built to obtain the fault test data of the airborne fuel pump. Firstly, based on the FMECA analysis of the statistics data of the airborne fuel pump over recent years, the common fault modes of the airborne fuel pump were obtained to guide the test. Secondly, based on the analysis of the physical model of the airborne fuel pump, the fault injection test was carried out in the testbed to collect the fault data of the relevant fault modes of the airborne fuel pump. After obtaining the data, the bearing fault data of Western Reserve University combined with our experimental data was used to carry out transfer learning research.

5.1. FMECA of the airborne fuel pump. The airborne fuel pump is a core component of the fuel system, and the pump is responsible for the fuel supply and fuel transfer of the aircraft. The structure of the airborne fuel pump is shown in Figure 5. Searching the degradation law of the airborne fuel pump is the basis for the life prediction of the airborne fuel pump. The time sequence of the degradation data of the airborne fuel pump is the key to predict the trend of breaking down, then estimate the life span of the airborne fuel pump [36, 37].

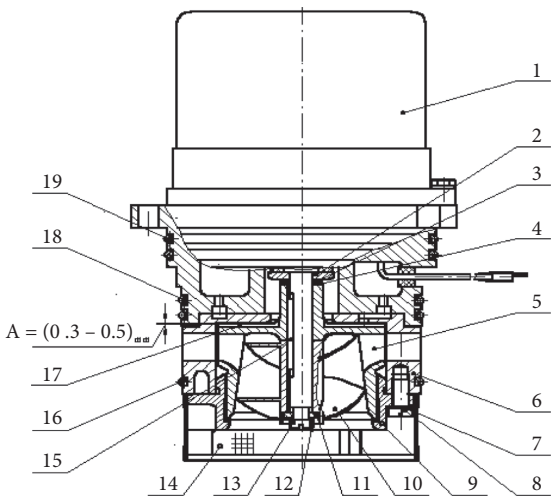
Through Failure Mode, Effects, and Criticality Analysis (FMECA) of the airborne fuel pump, six typical faults as blade damage, diffusion pipe damage, leakage, diffusion pipe and impeller rub, pump port and impeller rub, and bearing wear were selected, which is shown in Figure 6. Further analysis of the working principle and failure mechanism showed that when the fuel pump failure or performance declines, it will cause an abnormal vibration signal of the shell. However, in the military airfield, it is usually dismantled or returned to the factory for maintenance, without effective data monitoring and recording measures. Thus, it is difficult to quickly locate the fault, resulting in a reduction of the maintenance support level and waste of airborne equipment. Therefore, we considered selecting the vibration signal of the airborne fuel pump as the monitoring signal and carried out the time-frequency analysis and statistical characteristics analysis to extract the fault feature [38]. Our goal is to realize the intelligent and effective diagnosis of the airborne fuel pump.

5.2. Construction of airborne fuel pump testbed. A centrifugal AC electric pump provided by Nanjing Engineering Institute Centre is selected as the experiment object, as shown in Figure 7. This type of fuel pump is mainly used for the thermal subsystem and oil tanks. The fuel pump uses aviation fuel RP-3 as the working medium, whose temperature range from minus 60°C to 85°C. The other working parameters are shown in Table 1.

The experimental platform of the airborne fuel pump is shown in Figure 8. The platform mainly includes an oil storage tank, oil feeding tank, centrifugal test fuel pump, electric diaphragm pump, air-cooled radiator, pressure transducer, flow transducer, temperature transducer, liquid level transducer, data acquisition equipment, etc. In the main loop, the fuel pump pumps the oil from the oil feeding tank to the oil storage tank. For cycling, the oil in the storage tank returns to the feeding tank by gravity through the valve [39]. Through cycling, the working environment of the test pump is stable. In the second loop, an electric diaphragm pump is used to ensure the uniform distribution of particles in the impurity experiment. An air-cooled radiator is used to cool the oil and keep the oil temperature near room temperature. As shown in Figure 9, in the oil feeding tank, three vibration sensors are adopted to monitor the vibration of the airborne fuel pump, among which the vibration sensors are installed at three mutually perpendicular positions on the motor housing in the form of magnetic suction seats.

When testing, open the valve, fill the oil storage tank with fuel and connect the pump power supply, and make sure the pump continues to work. When the pump runs stably, collect the pump vibration signal, the outlet pressure signal, and the outlet flow signal. After the signal collection, close the power supply and the valve. Then, similar to the normal fuel pump, the other six typical fault signals are obtained by replacing different fault parts. The typical fault parts are shown in Figure 6. As shown in Table 2, vibration and pressure signals under normal state and 14 kinds of fault state were measured respectively in the experiment. Each group of data contained 4 channels, with a sampling frequency of 6000Hz and a sampling time of 5s for each channel.

5.3. Model application in the airborne fuel pump. In this paper, bearing fault data from Case Western Reserve University is selected as the source domain of transfer learning. Bearing faults in Case Western Reserve University are mainly caused by bearing wear, and the degree of bearing wear are 0.1778mm, 0.3556mm, 0.5334mm, and 0.7112mm. As for the target domain of airborne fuel pump, 1 impeller blade damage, diffusion tube damage, leakage, and bearing wear of 0.02mm were selected as the target domain for transfer learning. There are 4 types of ^{*3496} (vector number) ^{*246} (vector length) fault data available for the airborne fuel pump, and only 2^*246 data are selected for each type of fault. The remaining large amount of airborne fuel pump fault data are used as verification data for the transfer learning effect. In other words, the network trained by 2 sets of data was verified by 3494 sets of validation data to judge the diagnostic accuracy of the network. After the pre-processing, the small target samples combined with their labels are available for training.



1 Motor	2 Washer	3 Disc spring	4 Adjusting washer
5 Impeller	6 Diffusion tube	7 Washer	8 Cylinder head screw
9 Oil pump port	10 Axial flow impeller	11 Adjusting shim	12 Shim
13 Screws	14 Filters	15 Keys	16 Seals
17 Countersunk head screws	18 Seals	19 seals	

Figure 5: The structure of airborne fuel pump.

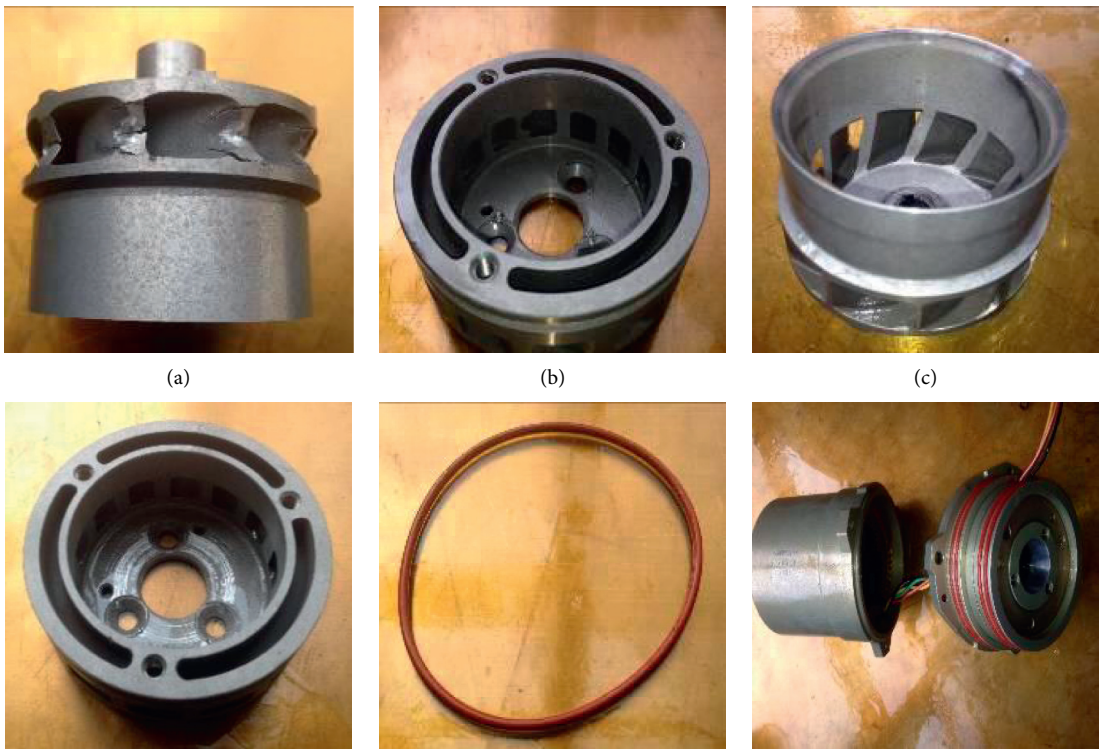


Figure 6: Typical faults of airborne fuel pump. (a) Blade damage; (b) Diffusion pipe damage; (c) Pump port and impeller rub; (d) Diffusion pipe and impeller rub; (e) Sealing ring aging; (f) Bearing wear.

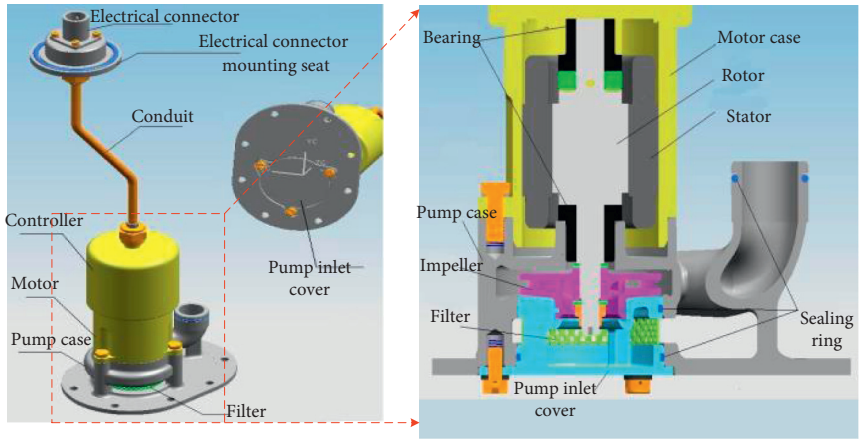


Figure 7: Structure diagram of a certain type centrifugal airborne fuel pump.

Table 1: Main working parameters of the airborne fuel pump.

Pump flow (L/h)	*Output pressure (Kpa)*	*Voltage (V)*	*Current (A)*	*Oil leakage (ml/min*
12000	≥73	115	≤5.5	0

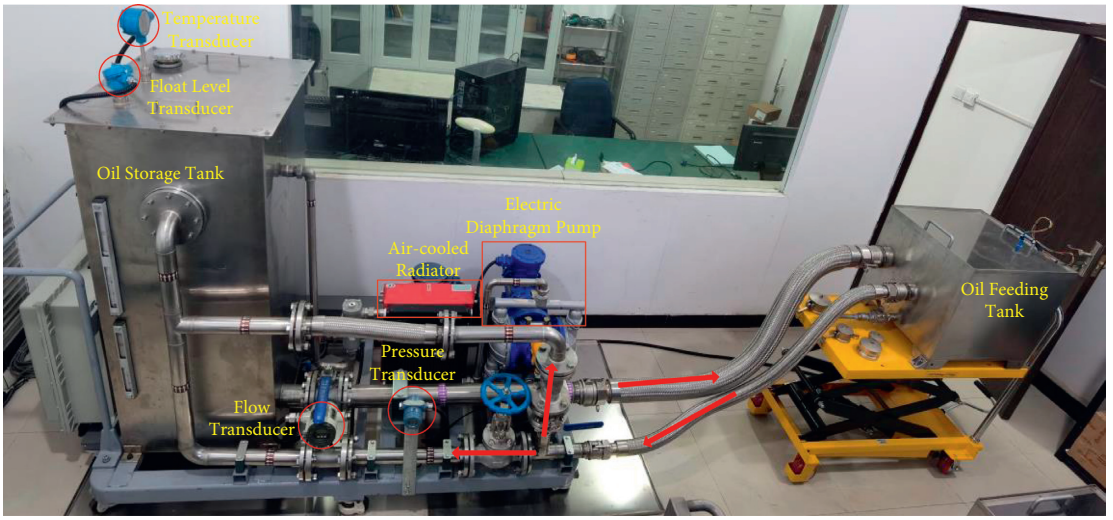


Figure 8: Testbed of the airborne fuel pump.

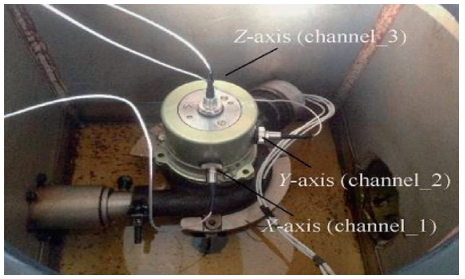


Figure 9: The installation of the airborne fuel pump in the testbed.

For the proposed model, there are hyper-parameters like learning rate, regularization parameter, cost function to be determined. In this research, we adopted randomized search to seek the optimized hyper-parameters. Nevertheless, the proposed model was complex that the process of hyper-parameters optimization cost too much time. Therefore, we sampled a smaller dataset to fed into the model. Although the accuracy of hyper-parameters decreased, the time cost reduced greatly. Through this way, the hyper-parameters of the model were determined [40].

To test the efficiency of the proposed model, firstly, fault diagnosis without transfer learning was carried out for small sample fault data of airborne fuel pump, and the confusion matrix of diagnosis results was shown in Figure 10. Then, the feature transfer model was used, bearing fault data from Case Western Reserve University was taken as the source domain, and the small sample fault data of airborne fuel pump was taken as the target domain. The results of the confusion matrix of the transfer diagnosis were shown in Figure 10. According to the results, the fault diagnosis

Table 2: Scheme of fault injection test.

NO.	Statement		Sample size (group)	Channel number	Time(second)	Frequency (Hz)
1		Normal	30	4	5	6000
2	foreign object damage	1 impeller blade	30	4	5	6000
3		2 impeller blades	30	4	5	6000
4		all impeller blades	30	4	5	6000
5		Diffusion tube	30	4	5	6000
6		all impeller blades & Diffusion tube	30	4	5	6000
7	scratch	back of impeller scrapes with diffusion tube	2	4	60	6000
8		edge of impeller inlet scrapes with oil pump mouth ring	2	4	60	6000
9		leakage	30	4	5	6000
10	bearing wear	Wear 0mm	15	4	5	6000
11		Wear 0.02mm	15	4	5	6000
12		Wear 0.05mm	15	4	5	6000
13		Wear 0.08mm	15	4	5	6000
14		Wear 0.10mm	15	4	5	6000
15		Wear 0.12mm	15	4	5	6000

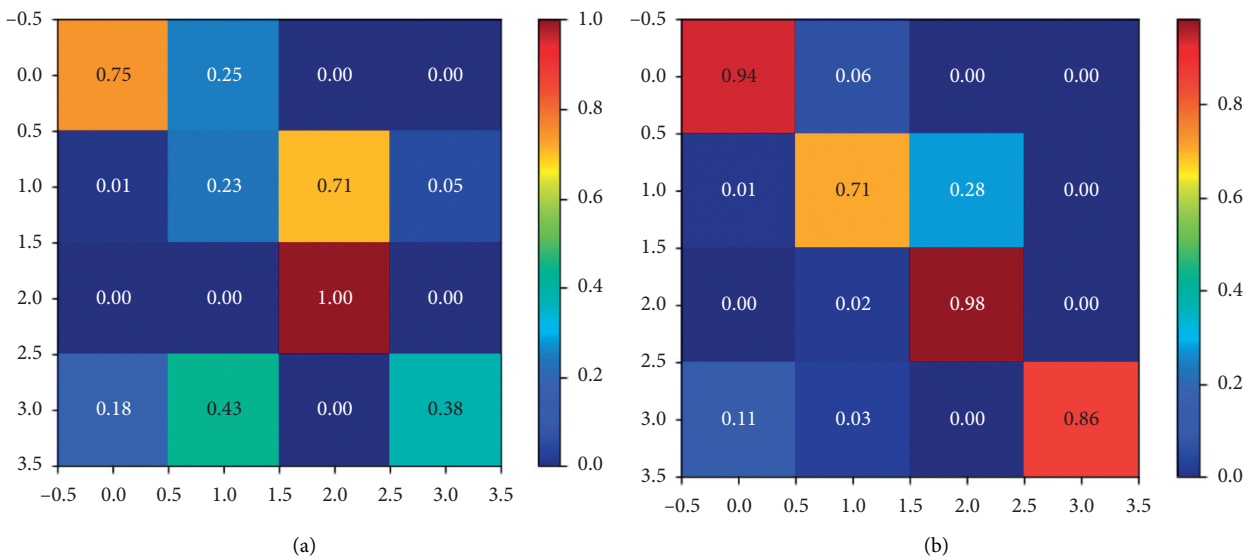


Figure 10: Confusion matrix of transfer learning results. (a) Confusion matrix without transfer learning; (b) Confusion matrix after transfer learning.

accuracy of each type of airborne fuel pump is improved by 28.25% on average through feature transfer learning.

Some other classical transfer learning algorithms such as ResNet-50, DANN, ADDA, JAN, MADA, CBST, CAN, CDAN+E, DM-ADA, 3CATN, ALDA are chosen as comparisons. The accuracy of diagnosis is given in Table 3. The results show that the diagnostic accuracy of the proposed model is remarkably higher than other algorithms. To further explore the capacity of the proposed model to filter the transferable data, some data filtering methods such as PCA, K-Means, DBSCAN, GMM, BIRCH are selected as comparisons. The accuracy of diagnosis is given in Table 4, the proposed model gets a higher score than models with other data filtering methods. The results show that the proposed model is more capable to filter available data in the source domain, which further proves that the proposed algorithm may prevent negative transfer to some extent.

5.4. Validation of feature transfer. To further verify the effectiveness of the model, this paper uses the method of feature visualization to show the learning effect of feature transfer. Since the features extracted from the network are in high-dimensional that cannot be directly visualized, the T-Distribution Stochastic Neighbour Embedding method is adopted to visualize the high-dimensional data.

T-distribution Stochastic Neighbour Embedding is often used to visualize high-dimensional data. The main advantage of T-SNE is the ability to preserve local structures. The

Table 3: Accuracy of diagnosis under different transfer learning approaches.

Model	Accuracy
Proposed model	87.3±0.5%
ResNet-50 (He et al., 2016)	63.2±0.8%
DANN (Ganin et al., 2016)	80.1±0.2%
ADDA (Tzeng et al., 2017)	53.8±1.5%
JAN (Long et al., 2017)	82.3±0.6%
MADA (Pei et al., 2018)	76.3±1.2%
CBST (Zou et al., 2018)	56.7±0.9%
CAN (Zhang et al., 2018)	75.1±0.5%
CDAN+E (Long et al., 2018b)	79.5±0.7%
DM-ADA (Xu et al., 2020)	58.7±1.1%
3CATN (Li et al., 2019)	83.2±0.6%
ALDA (Chen et al., 2020)	73.6±0.5%

Table 4: Accuracy of diagnosis with different data filtering methods.

Model	Accuracy
Proposed model (LLE+CNN+JDA)	87.3±0.5%
PCA+CNN+JDA	65.7±0.7%
K-Means+CNN+JDA	68.6±0.4%
DBSCAN+CNN+JDA	77.6±0.4%
GMM+CNN+JDA	85.2±0.6%
BIRCH+CNN+JDA	71.4±0.7%

Figure 11: Feature visualization by T-SNE. Red, blue, green, and purple represent sort 1-4 separately. (a) Feature visualization without transfer learning; (b) Feature visualization after transfer learning.

T-SNE algorithm models the distribution of the nearest neighbours of each data point. We model the high-dimensional space as a Gaussian distribution, while model the two-dimensional output space as a T-distribution. The goal of this process is to find the transformation that maps a higher-dimensional space to a two-dimensional space and to minimize the gap between these two distributions of all points [41].

For the network without transfer learning, the convolutional neural network for feature extraction was trained by 2 sets of data, and 3,494 sets of data were used as validation data. Feature extraction was carried out in the Convolutional Neural Network from validation data, and the extracted features were visualized using T-Distribution Stochastic Neighbour Embedding. The obtained results are shown in Figure 11(a). Similarly, for the model with transfer learning, the features extracted from the verification data were visualized using T-Distribution Stocking Neighbour Embedding, as shown in Figure 11(b). As can be seen from the figure, for networks without feature transfer, the boundaries of feature categories 1, 2, and 4 extracted from verification data are not clear, which is not conducive to the next step of feature classification and diagnosis. In the network with feature migration, the four categories of features extracted from the verification data have clear boundaries, which is easy to carry out classification and diagnosis. The effectiveness of the model for feature transfer is further proved from the feature visualization.

6. Conclusions

In this research, we proposed a feature transfer scenario that transfers knowledge from similar fields to enhance the accuracy of fault diagnosis with small sample. To reduces the

redundant information, data were filtered according to manifold consistency. Then, features were extracted based on CNN and feature transfer was conducted. For adequate fitness, the joint adaptation of conditional distribution and marginal distribution was used between the two domains. Minimum structural risk and MMD of adaptation were two indicators weighted for training the model. To test the efficiency of the model, we built an airborne fuel pump testbed, and contributed a new dataset that contained 15 categories of fault data, which serves as the small sample dataset in this research. Then the proposed model was applied in our experimental data. As a result, the fault diagnosis rate increases by 28.6% through our proposed model, which is more precise than other classical methods. The results of feature visualization further demonstrate that the features are more distinguished through the proposed method. All code and data are accessible on my GitHub.

Data Availability

Data and code of this research are accessible, please visit: https://github.com/ppqweasd/Diagnosis-for-High-reliability-Equipment-with-Small-Sample-Based-on-Transfer-Learning-A-General-Fra.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] Y. Wang, Z. Pan, X. Yuan, C. Yang, and W. Gui, "A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network," *ISA Transactions*, vol. 96, pp. 457–467, 2020.

[2] N. Md Nor, C. R. Che Hassan, and M. A. Hussain, "A review of data-driven fault detection and diagnosis methods: applications in chemical process systems," *Reviews in Chemical Engineering*, vol. 36, no. 4, pp. 513–553, 2020.

[3] Y. Wang, H. Yang, X. Yuan, Y. A. W. Shardt, C. Yang, and W. Gui, "Deep learning for fault-relevant feature extraction and fault classification with stacked supervised auto-encoder," *Journal of Process Control*, vol. 92, pp. 79–89, 2020.

[4] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: methods and applications," *Journal of Manufacturing Systems*, vol. 48, pp. 144–156, 2018.

[5] T. Kinoshita, K. Fujiwara, Y. Sumi, M. Matsuo, M. Kano, and H. Kadotani, "Development of spindle detection algorithm by wavelet synchrosqueezed transform and random under sampling," *Sleep Medicine*, vol. 64, p. S121, 2019.

[6] P. Soltanzadeh and M. Hashemzadeh, "RCSMOTE: range-Controlled synthetic minority over-sampling technique for handling the class imbalance problem," *Information Sciences*, vol. 542, pp. 92–111, 2021.

[7] N.-T. Tran, V.-H. Tran, N.-B. Nguyen, T.-K. Nguyen, and N.-M. Cheung, "On data augmentation for gan training," *IEEE Transactions on Image Processing*, vol. 30, pp. 1882–1897, 2021.

[8] X. Zhou, X. Liu, G. Lan, and J. Wu, "Federated conditional generative adversarial nets imputation method for air quality missing data," *Knowledge-Based Systems*, vol. 228, 2021.

[9] T. Lee and S. Yoo, "Augmenting few-shot learning with supervised contrastive learning," *Ieee Access*, vol. 9, pp. 61466–61474, 2021.

[10] R.-Q. Wang, X.-Y. Zhang, and C.-L. Liu, *Meta-Prototypical Learning for Domain-Agnostic Few-Shot Recognition*, IEEE Trans. Neural Netw. Learn. Syst., 2021.

[11] F. Zhuang, Z. Qi, K. Duan, D. Xi, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 99, pp. 1–34, 2020.

[12] Z. P. Yu, J. H. Zhao, Y. C. Wang, L. L. He, and S. N. Wang, "Surface EMG-based instantaneous hand gesture recognition using convolutional neural network with the transfer learning method," *Sensors*, vol. 21, no. 7, p. 21, 2021.

[13] S. Smetanin and M. Komarov, "Deep transfer learning baselines for sentiment analysis in Russian," *Information Processing & Management*, vol. 58, no. 3, p. 19, 2021.

[14] A. Langevin, T. Cody, S. Adams, and P. Beling, "Generative adversarial networks for data augmentation and transfer in credit card fraud detection," *Journal of the Operational Research Society*, p. 28, 2021.

[15] N. Wu, F. Liu, F. J. Meng, M. Li, C. Zhang, and Y. He, "Rapid and accurate varieties classification of different crop seeds under sample-limited condition based on hyperspectral imaging and deep transfer learning," *Frontiers in Bioengineering and Biotechnology*, vol. 9, p. 14, 2021.

[16] F. Liu, J. Lu, B. Han, G. Niu, G. Zhang, and M. Sugiyama, "Butterfly: one-step approach towards wildly unsupervised domain adaptation," 2019.

[17] X. Chen, X. Lan, F. Sun, and N. Zheng, "A boundary based out-of-distribution classifier for generalized zero-shot learning," 2020.

[18] T. Teshima, I. Sato, and M. Sugiyama, "Few-shot domain adaptation by causal mechanism transfer," 2020.

[19] Y. L. Wang, D. Z. Wu, and X. F. Yuan, "LDA-based deep transfer learning for fault diagnosis in industrial chemical processes," *Computers & Chemical Engineering*, vol. 140, pp. 13-14, 2020.

[20] V. Singh and N. K. Verma, "mRMR-DNN with transfer learning for IntelligentFault diagnosis of rotating machines," 2019.

[21] Y. Deng, D. Huang, S. Du, G. Li, C. Zhao, and J. Lv, "A double-layer attention based adversarial network for partial transfer learning in machinery fault diagnosis," *Computers in Industry*, vol. 127, Article ID 103399, 2021.

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

[23] R. Yan, F. Shen, C. Sun, and X. Chen, "Knowledge transfer for rotary machine fault diagnosis," *IEEE Sensors Journal*, vol. 20, no. 15, p. 1, 2019.

[24] Z. Yue, L. Meng, L. Liangzhen, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data arXiv," vol. 13, p. 13, 2018.

[25] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, pp. 2399–2434, 2006.

[26] Juba and Brendan, "Estimating relatedness via data compression," in *Proceedings of the 23rd international conference on Machine learning*, pp. 441–448, USA, June 2006.

[27] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[28] M. Rezaei-Ravari, M. Eftekhari, and F. Saberi-Movahed, "Regularizing extreme learning machine by dual locally linear embedding manifold learning for training multi-label neural network classifiers," *Engineering Applications of Artificial Intelligence*, vol. 97, p. 15, 2021.

[29] J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony, "Structural risk minimization over data-dependent hierarchies," *IEEE Transactions on Information Theory*, vol. 44, no. 5, pp. 1926–1940, 1998.

[30] J. Liu, M. Bai, N. Jiang, and D. Yu, "Structural risk minimization of rough set-based classifier," *Soft Computing*, vol. 24, no. 3, pp. 2049–2066, 2020.

[31] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," *Proceedings*, in *Proceedings of the IEEE International Conference on Computer Vision*, Sydney, NSW, Australia, December 2013.

[32] A. Smola, A. Gretton, L. Song, and B. Schölkopf, *A Hilbert Space Embedding for Distributions*, pp. 13–31, Springer, Berlin, Germany, 2007.

[33] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schlkopf, and A. J. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, pp. 723–773, 2012.

[34] M. Long, J. Wang, G. Ding, S. J. Pan, and P. S. Yu, "Adaptation regularization: a general framework for transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1076–1089, 2014.

[35] M. Long, J. Wang, G. Ding, D. Shen, and Q. Yang, "Transfer learning with graph Co-regularization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 7, pp. 1805–1818, 2014.

[36] X. Jiao, B. Jing, X. Qiang, Q. Liu, J. Li, and W. Zhou, "Fault diagnosis of airborne fuel pump and experimental platform research," vol. 36, no. 01, pp. 120–128, 2017.

[37] J. X. Pan, B. Jing, Z. Li, S. Wang, and X. X. Jiao, "Modeling and experimental design of electrical stress accelerated degradation of airborne fuel pump," *Journal of Electronic Measurement and Instrument*, vol. 33, no. 11, pp. 50–56, 2019.

[38] X. Jiao, B. Jing, Y. Huang, J. Li, and G. Xu, "Research on fault diagnosis of airborne fuel pump based on EMD and probabilistic neural networks," *Microelectronics Reliability*, vol. 75, no. aug, pp. 296–308, 2017.

[39] J. X. Pan, B. Jing, X. X. Jiao, and S. L. Wang, "Analysis and application of grey wolf optimizer-long short-term memory," *IEEE Access*, vol. 8, pp. 121460–121468, 2020.

[40] M. Abd Elaziz, A. Dahou, L. Abualigah et al., "Advanced metaheuristic optimization techniques in applications of deep neural networks: a review," *Neural Computing & Applications*, vol. 33, no. 21, pp. 14079–14099, 2021.

[41] V. D. M. Laurens and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 2605, pp. 2579–2605, 2008.

Hindawi
Complexity
Volume 2021, Article ID 3005276, 16 pages
https://doi.org/10.1155/2021/3005276

WILEY | Hindawi

Research Article

Online Semisupervised Learning Approach for Quality Monitoring of Complex Manufacturing Process

Weng Weiwei,[1] Mahardhika Pratama ,[1] Andri Ashfahani,[1] and Edward Yapp Kien Yee[2]

[1]*School of Computer Science and Engineering, Nanyang Technological University, Singapore*
[2]*Singapore Institute of Manufacturing Technology, A∗STAR, Singapore*

Correspondence should be addressed to Mahardhika Pratama; pratama@ieee.org

Received 15 April 2021; Accepted 10 August 2021; Published 2 September 2021

Academic Editor: Taeseong Kim

Data-driven quality monitoring is highly demanded in practice since it enables relieving manual quality inspection of the product quality. Conventional data-driven quality monitoring is constrained by its offline characteristic thus being unable to handle streaming nature of sensory data and nonstationary environments of machine operations. Recently, there have been pioneering works of online quality monitoring taking advantage of online learning concepts in the literature, but it is still far from realization of minimum operator intervention in the quality monitoring because it calls for full supervision in labelling data samples. This paper proposes Parsimonious Network++ (ParsNet++) as an online semisupervised learning approach being able to handle extreme label scarcity in the quality monitoring task. That is, it is capable of coping with varieties of semisupervised learning conditions including random access of ground truth and infinitely delayed access of ground truth. ParsNet++ features the one-pass learning approach to deal with streaming data while characterizing elastic structure to overcome rapidly changing data distributions. That is, it is capable of initiating its learning structure from scratch with the absence of a predefined network structure where its hidden nodes can be added and discarded on the fly in respect to drifting data distributions. Furthermore, it is equipped by a feature extraction layer in terms of 1D convolutional layer extracting natural features of multivariate time-series data samples of sensors and coping well with the many-to-one label relationship, a common problem of practical quality monitoring. Rigorous numerical evaluation has been carried out using the injection molding machine and the industrial transfer molding machine from our own projects. ParsNet++ delivers highly competitive performance even compared to fully supervised competitors.

1. Introduction

1.1. Background. Predictive maintenance has attracted increasing interest from both academia and industry because it offers optimization of machine's life cycle, accurate planning of machine's maintenance, and prevention of unnecessary downtime and product's wastage [1]. In realm of tool condition monitoring, replacing a tool too frequently not only leads to expensive maintenance cost but also interrupts the production cycle. On the other hand, blunt tools incur high energy consumption due to the application of high cutting force or undermines the surface finishing.

Accurate quality monitoring plays a vital role in reducing rejection's rate by customers leading to high customer satisfaction and in meeting particular standards set by relevant authorities. Common practice for quality monitoring is still done via multi-staged visual inspection deemed overly labour-intensive, error-prone, and slow. Another drawback of manual quality monitoring is found in the issue of consistency. That is, human operators are often biased and are affected by uncertain factors such as experiences, fatigues, boredom, etc. This rationale triggers increasing demand of data-driven quality monitoring utilizing artificial intelligence (AI) techniques feeding real-

time information of product's quality [2]. Compared to the traditional first principle approach, the data-driven quality monitoring cuts down the development time significantly. It relies on a dataset collected from sensors or cameras installed at the end of production line to build a predictive model after being preprocessed via the signal processing and feature extraction techniques to produce meaningful features.

1.2. Related Works. In-depth study has been devoted to developing reliable quality monitoring approaches. In [3], the tool condition of the metal-turning process is predicted using neural networks. A fuzzy neural network is utilized to predict the tool wear of the ball-nose end-milling process using vibration data [4]. In [5, 6], a fault detection approach in the rolling mills process is proposed using all-coverage data-driven approach making possible to integrate many sensors. The rise of deep learning with its automatic feature engineering step to extract natural features allows simplification of the data-driven quality monitoring enabling to bypass complex feature extraction step. In [7], convolutional neural networks based on ResNet50 are put forward to perform quality monitoring in the laser-based manufacturing processes. A stacked sparse autoencoder (SSAE) combined with the genetic algorithm to tune its parameters is proposed to determine the laser welding quality [8]. Despite their success in various manufacturing applications, such approaches are offline in nature and fixed once deployed thus being unable to adapt to rapidly changing conditions of process parameters. Their iterative training process is not memory-wise and does not keep pace with the high-speed manufacturing process. A complete retraining process from scratch is solicited in handling the process change.

The online quality classification approach has been advanced in [9] where the GEN-SMART-EFS is combined with the incremental partial least square (iPLS) method for the feature selection method to monitor the quality of microfluidic chip. An extension of this work is presented in [10] where the forgetting strategy is implemented to handle the concept drift and the multiobjective evolutionary computation approach for process optimization. Another approach for prediction of tool wear in the metal-turning process is proposed in [11]. It is based on Parsimonious Ensemble+ (pENsemble+) algorithm making use of the online active learning approach to handle the issue of label's scarcity.

1.3. Our Approach. The area of online quality classification still deserves investigation because existing methods are still far from being truly autonomous approaches. They are mostly developed from the fully supervised learning principle necessitating considerable labelling efforts in streaming environments. It suffers from substantial operator dependencies to fully annotate data samples for model's updates notably in the high-speed production processes. In [12], a semisupervised deep learning approach is proposed for quality monitoring tasks using the stacked autoencoder approach. However, this approach is not designed for streaming environments. Another approach is proposed in [13] for online semisupervised quality monitoring using the notion of weighted principal component regression. This approach is, however, a non-deep learning approach. Another open issue lies in the feature extraction step often being application-specific [14] and calling for intensive offline phases. Notwithstanding the fact that deep learning solution starts to pick up research interest where the concept of deep features is utilized to bypass complex feature engineering step, they are built upon an offline training process thus becoming outdated quickly under nonstationary traits of manufacturing processes. Furthermore, they are developed under a fully supervised working principle incurring considerable labelling cost. Another issue lies in the existence of many-to-one label relationship [15] where a batch of data are associated with a single and constant class label. This problem might lead to the overfitting problem of a particular class or the loss of granularity if a batch of data is combined into a single instance. This problem is frequently found in the condition monitoring problem, in which a quality check is only performed after the whole lot is produced. In summary, there exists a strong demand for an online semisupervised deep learning algorithm for quality monitoring. Such algorithm is capable of learning from streaming data without retraining from scratch while bypassing a complex feature engineering phase. That is, a new concept arising due to changing environments can be quickly handled without compromising complexity while natural features are extracted on the fly.

An online semisupervised deep neural network, namely, Parsimonious Network++ (ParsNet++), is proposed to undertake real-time learning under scarcity of labelled samples for online quality monitoring in the injection molding process [16] and in the industrial transfer molding process. ParsNet++ forms a significant extension of a recently developed algorithm for semisupervised learning of high-pace data streams, Parsimonious Network (ParsNet) [17]. ParsNet++ is capable of starting its learning process from scratch with no predefined structure while its hidden node is automatically grown and pruned from data streams to overcome the concept drift. It handles the partially labelled data streams under two settings: random access of ground truth and infinitely delayed access of ground truth. The key feature exists in the autoregularization method dealing with the accumulation of mistakes due to noisy pseudolabel.

The underlying innovation of ParsNet++ lies in the existence of feature extraction layer coping with raw samples where the 1D convolutional layer is integrated to deal with multivariate time-series data collected from sensors and the many-to-one label relationship. This property enables skipping a complex feature engineering step because of its aptitude in extracting natural features. The feature extraction layer is structured as a stacked convolutional layer generating deep features to be fed to the fully connected layer. Furthermore, the fully connected layer is structured as a self-evolving single-hidden-layer neural network to handle process change.

The structural learning mechanism of ParsNet++ is driven by the network significance (NS) method derived from the bias-variance decomposition method. It differs from the original NS method in [18] with the presence of autonomous clustering mechanism (ACM) estimating the probability density function. ACM addresses the obsolete probability density function if the concept drift occurs while also relaxing a strict normal distribution assumption which does not fit for real-world cases. Unlike conventional clustering technique, ACM features a self-evolving property making possible for automatic generation and pruning mechanism of clusters. ACM distinguishes itself from AGMM of the original ParsNet often being unstable in the high input dimension cases.

The parameter learning phase is carried out under a joint optimization problem minimizing both reconstruction loss and discriminative loss coupled with autoregularization mechanism. That is, the regularization process is derived from the concept of synaptic intelligence (SI) proposed to prevent the issue of catastrophic forgetting problem [19]. It calculates the parameter importance using the accumulated gradient of network synapses. This technique is generalized here where it is used to memorize optimal network parameters induced by the clean labels. The label enrichment method is carried via the label augmentation mechanism where originally labelled samples are perturbed by injecting controlled noise while leaving their labels unchanged. By extension, the self-labelling mechanism is carried out to generate pseudolabel of unlabelled samples. It is inferred by the predictive output of ACM and network itself if both of them are confident with their own predictions.

Autonomous quality monitoring with weak supervision is formalised under two settings: random access of ground truth and infinitely delayed access of ground truth. The former case portrays partially labelled data streams where only a fraction of data samples possess true class label. The latter case goes one step ahead where labelled samples are served only during the warm-up phase leaving the rest unlabelled. Furthermore, the quality monitoring problem consists of two scenarios, current batch prediction and one-step ahead prediction. The current batch prediction is meant to predict the current product quality whereas the second one aims to forecast the product quality for the next data stream, all of which are carried out in the prequential test-then-train protocol, the standard simulation protocol of data streams and simulated using real-world use cases of injection molding machine, and industrial transfer molding machine from our own project. Our rigorous numerical study demonstrates the success of ParsNet++ for the online quality classification under weak supervision where it delivers the most encouraging results even compared to fully supervised competitors.

In summary, this paper delivers four major contributions discussed in the sequel:

(1) This paper presents ParsNet++ to handle online quality classification of injection molding process and industrial transfer molding process under semisupervised environments. That is, the semisupervised environments are induced by both random access of ground truth and infinitely delayed access of ground truth.

(2) This paper offers an extension of ParsNet [17] where 1D convolutional layer is introduced to address the issue of feature extraction and the many-to-one label relationship problem.

(3) Autonomous clustering mechanism (ACM) is developed for a flexible density estimation approach navigating the structural learning phase. ACM replaces the role of AGMM in the original ParsNet [17] suffering from the execution issue of the high-dimensional problem. Furthermore, ACM incurs fewer parameters than those of AGMM.

(4) The codes of ParsNet++, raw numerical results, and injection molding dataset are made publicly available in https://github.com/ContinualAL/ParsNetPlus to enable further study of the proposed research topic.

The remainder of this paper is structured as follows: Section 2 discusses the problem formulation; Section 3 outlines the learning policy of ParsNet++; Section 4 elaborates the injection molding machine; our numerical study is explained in Section 5; and some concluding remarks are drawn in Section 6.

2. Problem Definition

Learning from data streams is defined as a learning problem of never-ending data batches $B_1, B_2, B_3, \ldots, B_K$ where K is the number of data streams and unknown in practice. This property demands the one-scan learning scheme where a data stream is discarded once learned to suppress the computational and memory complexities to a low level. A data stream comprises data samples $B_k = \{X_k\} = \{x_t\}_{t=1}^{T}$ having no label where X_k denotes input data batch while $x_t \in \mathfrak{R}^u$ denotes an input vector. T, u are, respectively, the batch size and the input dimension. In the realm of the fully supervised learning setting, the ground truth access $y_t \in \{l_1, l_2, \ldots, l_m\}$ where m is the output dimension can be instantly elicited. This assumption is unrealistic notably in the context of quality classification. Some delay is expected because the product quality is examined through visual inspection. Semisupervised data stream is formalised here under two settings: sporadic access to ground truth and infinitely delayed access to ground truth.

Random access to ground truth: this case delineates a fact where the operator labels data samples sporadically leading to partially labelled data streams. That is, a true class label y_t arrives in the random fashion. In other words, B_k is only partially labelled with the target label.

Infinitely delayed access to ground truth: the second case is more stringent than the first case where the access of true class label is only given for prerecorded samples being fed in the warm-up period before process runs leaving the rest unlabelled. In other words, only initial labels are provided. Specifically, only the first data batch B_1 is labelled without changing the data order.

As with conventional data streams, semisupervised data streams do not follow static and predictable data distributions where they contain the concept drifts. That is, there is changing data distributions resulting in the change of joint probability distribution $P(X,Y)_t \neq P(X,Y)_{t+1}$. It requires a model which can adapt to the concept drifts with/without the presence of true class labels. That is, a model should be capable of adapting to the concept drift even if the true class label is absent. The concept drift is induced in our experiment with the injection molding machine by varying the holding pressure and the injection speed of the injection molding machine to be $900, 700, 500, 300, 100$ psi and $60, 70, 80, 90, 100$ rpm, respectively. The online quality classification problem is presented as a multiclass classification problem with three classes, namely, good, weaving, and short-forming. The number of data samples in three classes is, respectively, 1008, 1074, and 870, respectively. This problem is guided by 48 input attributes recording different machine parameters.

3. Learning Policy of ParsNet++

Overview of ParsNet++'s learning policy is depicted in Algorithm 1. It starts from the learning process of ACM estimating the complex probability density function $p(x)$ and determining the addition factor of hidden nodes M. Note that ParsNet++ directly injects M hidden nodes if the hidden node growing condition is satisfied. Furthermore, ACM itself is flexible to changing learning environments $p(x)_t \neq p(x)_{t+1}$ since it features an elastic structure making possible for clusters to be added or pruned on the fly. The probability density function $p(x)$ produced by ACM is fed to the structural learning phase of ParsNet++ where the generative learning phase is carried out first to condition the network structure with the absence of true class label. The structural learning phase involves the hidden node growing and pruning processes adapting to the virtual drift problem. That is, the structural evolution is navigated by the reconstruction error. The parameter learning phase is devised to minimize the reconstruction loss and to create an ideal discriminative representation of unlabelled samples. The network parameters are further evolved in the discriminative phase with the access of true class labels once completing the generative phase. In other words, the generative and discriminative training phases occur in a fully coupled fashion. The label enrichment mechanism is carried out afterward by executing the augmentation of labelled samples module and the generation of pseudolabel mechanism. Both pseudolabel and augmented label are learned in the discriminative learning fashion minimizing the predictive loss and carried out along with the dynamic regularization method. Network parameters are shared during the generative and discriminative learning phases having a closed-loop configuration. That is, the network parameters of the generative learning phase are passed to the discriminative learning phase while the network parameters of the discriminative learning phase are fed back to the generative learning phase to cope with upcoming data stream, in other words, the discriminative phase function to refine the generative learning phase using the ground truth information. In addition to the generative phase, the structural learning phase takes place in the discriminative phase to overcome the real concept drift and utilizes the same probability density function $p(x)$ as per the generative training phase. Table 1 provides a list of notations used in the paper.

3.1. Parameter Learning of ParsNet++. The parameter learning method of ParsNet++ is governed by the following loss function:

$$L = \underbrace{L(X,\hat{X})}_{L_1} + \underbrace{L\left(Y_{\text{ori}},\hat{Y}_{\text{ori}}\right)}_{L_2} + \underbrace{L\left(Y_{\text{aug}},\hat{Y}_{\text{aug}}\right)}_{L_3} + \underbrace{L\left(Y_{\text{ps}},\hat{Y}_{\text{ps}}\right)}_{L_4} + \underbrace{\frac{1}{2}\alpha_1\rho\left(\theta-\theta^*\right)^2}_{R}, \tag{1}$$

where L_1 stands for the reconstruction loss solved in the generative phase via convolutional denoising autoencoder (CDAE), L_2 denotes the predictive loss of originally labelled samples having a much lower quantity than that of the batch size, and L_3 and L_4 label the predictive loss of augmented label and pseudolabel, respectively. The last term is the autoregularization term. The pseudolabel is induced by the self-labelling mechanism to unlabelled samples while the augmented label is produced by injecting small perturbation to originally labelled samples without changing its label. Nonetheless, the self-labelling mechanism does not reflect the ground truth and possibly delivers noisy label compromising the model's generalization. The autoregularization here plays a role in avoiding this situation by preventing the important parameters to move away from its optimal parameters as a result of the originally labelled samples. That is, θ and θ^*, respectively, denote the current network parameters and the optimal parameters induced by the ground truth while ρ is the indicator of parameter importance. The original label, augmented label, and pseudolabel are mixed here to enable the autoregularization to be executed seamlessly [17]. Furthermore, the structural learning phase takes place in L_1 and L_2 here because the augmented label does not reflect the true data distribution undermining the drift adaptation mechanism and the pseudolabel risks on noisy label misleading the estimation of bias and variance. Equation (1) is formed as an unconstrained optimization problem allowing alternate optimization strategy via the stochastic gradient descent (SGD) method. Notwithstanding the fact that pseudolabels might be noisy, the pseudolabel generation mechanism still plays an important role to enhance model's generalization because it enriches the label representation; i.e., one might consider extreme label scarcity here. Moreover, the autoregularization is

Input: partially labelled data batches: $B_1, B_2, B_3, \ldots, B_K$
  for data batch $B_k = B_1$: B_K **do**
    Testing and update performance metrics
    if $k < S$ **then** {S: initialization batch number}
      for epochs = 1:E **do**
        Update ACM
        $(X_{\text{aug}}, Y_{\text{aug}}) \leftarrow (X_{\text{ori}}, Y_{\text{ori}})$
        $X_{\text{gen}} \leftarrow [X_{\text{ori}}, X_{\text{aug}}]$ {gen:generative phase}
        for all X_{gen} **do**
          Structural evolution
          $\widehat{X_{\text{gen}}} \leftarrow \text{net}(X_{\text{gen}})$
          Calculate $L(X_{\text{gen}}, \widehat{X_{\text{gen}}})$ {L_1 in (1)}
        end for
        $X_{\text{dis}} \leftarrow [X_{\text{ori}}, X_{\text{aug}}]$
        for all X_{dis} **do** {dis:discriminative phase}
          Structural evolution
          $\widehat{Y_{\text{dis}}} \leftarrow \text{net}(X_{\text{dis}})$
          Calculate $L(Y_{\text{dis}}, \widehat{Y_{\text{dis}}})$ {L_2 and L_3 in (1)}
        end for
      end for
    else
      Update ACM
      if B_k exists unlabelled data **then**
        Generate pseudolabel $(X_{\text{ps}}, Y_{\text{ps}})$ via (2)
      end if
      $(X_{\text{aug}}, Y_{\text{aug}}) \leftarrow (X_{\text{ori}}, Y_{\text{ori}})$
      $X_{\text{gen}} \leftarrow [X_{\text{ori}}, X_{\text{aug}}, (X_{\text{ps}})]$
      Calculate $L(X_{\text{gen}}, \widehat{X_{\text{gen}}})$ {L_1 in (1)}
      for all X_{gen} **do**
        Structural evolution
        $\widehat{X_{\text{gen}}} \leftarrow \text{net}(X_{\text{gen}})$
      end for
      $X_{\text{dis}} \leftarrow [X_{\text{ori}}, X_{\text{aug}}, (X_{\text{ps}})]$
      for all X_{dis} **do**
        Structural evolution
        $\widehat{Y_{\text{dis}}} \leftarrow \text{net}(X_{\text{dis}})$
        Calculate $L(Y_{\text{dis}}, \widehat{Y_{\text{dis}}})$ {L_2, L_3 and (L_4) in (1)}
        Update net with R in (1) {autoregularization}
      end for
    end if
  end for

Algorithm 1: ParsNet++ algorithm.

implemented to address the issue of noisy pseudolabels. The generative and discriminative phases are carried out alternately here. Note that the infinite delay case only relies on the augmented label and the pseudolabel.

3.1.1. Generation of Augmented Label. The issue of label scarcity is addressed by the label enrichment strategy including the generation of augmented label. It results from the injection of small Gaussian noise to the originally labelled samples without changing their labels also known as the consistency regularization technique. That is, small random Gaussian noise with zero mean is utilized to produce the corrupted version of originally labelled samples, i.e., $N(0, 0.001)$ [17]. Since the augmented label is drawn from the true class label, it is not subject to the autoregularization method. Furthermore, only augmented label and pseudolabel are exploited in the infinite delay problem whereas original label is not retained during the process runs. In other words, original label is accessed in the warm-up phase without being carried to the next data streams.

3.1.2. Generation of Pseudolabel. The label enrichment mechanism involves the generation of pseudolabel produced by the self-labelling phase of unlabelled samples. The self-labelling mechanism relies on the network prediction as well as the ACM prediction if they return high confidence as follows:

$$\begin{aligned} P(Y|X)_{\text{net}} &\geq \alpha_2, \\ P(Y|X)_{\text{ACM}} &\geq \alpha_3, \end{aligned} \tag{2}$$

where α_2, α_3 are two predefined thresholds set to be higher than 0.55. The ACM's output is calculated as per the output

Table 1: List of notations.

Notation	Meaning
B_k	$k-$th data batch in data streams
x, y	Single input data vector and single ground truth output vector separately
X, Y	Input data batch and batch label
θ, θ^*	The current network parameters and optimal network parameters
ρ	Network parameter importance, calculated by (3)
α_1	Regularization factor: $\alpha_1 = (e_{\text{recons}} - e^{\min}_{\text{recons}})/(e^{\max}_{\text{recons}} - e^{\min}_{\text{recons}})$
α_2, α_3	Predefined thresholds in (2)
Car_m	The cardinality of the $m-$th cluster
$\text{Car}_{o,m}$	The cardinality of the $o-$th class of the $m-$th cluster
F	Convolution layer
Z	Feature map
C	Cluster center
$\widetilde{Z}_L$	Partially destroyed input vector with the masking noise
$D(X, Y)$	The $L-2$ distance between two data samples
Act_m	The contribution of $m-$th cluster
ω_m	The mixing coefficient for hidden node pruning criterion
$\wedge$	Reconstructed symbol
ACM	Autonomous clustering mechanism
ori	Original data
aug	Augmented data (generation of augmented label of Section 3)
ps	Pseudodata (generation of pseudolabel of Section 3)

posterior probability [20] $P(Y|X)_{\text{ACM}} = (\sum_{m=1}^{M} P(y_o|N_m)P(N_m)P(X|N_m)/\sum_{o=1}^{C}\sum_{m=1}^{M} P(y_o|N_m)P(N_m)P(X|N_m))$ where $P(N_m)$ denotes the prior probability $(\text{Car}_m/\sum_{m=1}^{M}\text{Car}_m)$, $P(y_o|N_m)$ stands for the class posterior probability $P(y_o|N_m) = (\text{Car}_{o,m}/\sum_{o=1}^{C}\text{Car}_{o,m})$, and $P(X|N_m)$ labels the likelihood function $P(X|N_m) = \exp(-(Z_L - C_m)^2)$. Car_m stands for the cardinality of the $m-$th cluster while $\text{Car}_{o,m}$ denotes the cardinality of the $o-$th class of the $m-$th cluster. Furthermore, the network and ACM predictions are normalized as $P(Y|X)_{\text{net/ACM}} = y_1/(y_1 + y_2)$ where y_1, y_2 denote the highest and second highest outputs. This trait underpins the class-invariant trait being similar to the binary classification problem. As a result, $P(Y|X)_{\text{net/ACM}} \approx 0.5$ indicates low confidence level and confused prediction. This condition implies the predicted output falls adjacent to the decision boundary. The pseudolabel is propagated to model's update only if the predictive outputs of ACM and network are agreeable. Despite the pseudolabel generation mechanism risks on the noisy pseudolabel, it is still integrated in the ParsNet++ learning mechanism because of the existence of autoregularization making sure only clean pseudolabels to be learned. On the other hand, α_2, α_3 control the self-labelling mechanism where the higher values lead to the decrease of the pseudolabels whereas the lower values lead to the increase of the pseudolabels.

3.1.3. Autoregularization Method. The autoregularization is developed to cope with noisy pseudolabel leading to accumulation of mistakes. It prevents a model to forget its optimal condition resulting from learning original label. Specifically, it prevents important parameters θ from moving too far from their previous locations θ^* resulting in the performance degradation. This approach is originally proposed in the so-called synaptic intelligence (SI) technique addressing the catastrophic forgetting problem of continual learning [19]. Our main contribution here is to contextualize this approach for the semisupervised learning environment to prevent the catastrophic forgetting problem as a result of noisy pseudolabel.

$(1/2)\alpha_1\rho(\theta - \theta^*)^2$ still accepts the pseudolabel by setting α_1, regularization factor, as $(e_{\text{recons}} - e^{\min}_{\text{recons}})/(e^{\max}_{\text{recons}} - e^{\min}_{\text{recons}})$ where e_{recons} stands for the reconstruction error of the generative phase only if clean pseudolabel is fed. That is, wrong pseudolabel distracts the direction of network's gradient resulting in the increase of reconstruction error. The Z-score is applied to scale the reconstruction error in the range of $[0, 1]$. ρ determines the importance of network parameters derived from the accumulated network gradient as follows:

$$\rho = \frac{\sum_{t=1}^{\text{step}} \Delta\theta_t \left(\partial L/\partial\theta_t\right)}{\left(\theta_T\right)^2 + \varepsilon}, \tag{3}$$

where θ_T stands for the total parameter movement during the training process and $\Delta\theta$ denotes the parameter's movement during two consecutive time steps $\theta_t - \theta_{t-1}$. ε is a predefined constant to avoid division with zero. ρ is updated only when observing the original label and the augmented label because the autoregularization functions to compensate the noisy pseudolabel. Hence, step denotes the number of original label and augmented label. It is worth mentioning that the higher the network gradient is, the more important the network parameter is. The parameter importance indicator (3) is calculated in respect to the accumulation of network loss and network gradients.

3.2. Network Structure of ParsNet++. ParsNet++ is built upon the convolutional denoising autoencoder structure where the feature extraction layer utilizes the stacked

convolutional layers while the fully connected layer is formed as a single-hidden-layer network having a self-organizing property. It receives raw input features collected from sensors $X_t^{\text{sen}} \in \mathfrak{R}^u$ which in turn maps them to the output space $Y_k \in \mathfrak{R}^m$. Specifically, the 1D convolutional layer is deployed to process the sensor data. Raw samples are executed by the convolutional layer $F(\cdot)$ as follows:

$$Z_l^i = F\left(X, W_{\text{conv}^{l,i}}\right), \tag{4}$$

where the convolutional layer $F(\cdot)$ is parameterized by a filter $W_{\text{conv}}^{l,i}$ denoting the $i-$th filter of the $l-$th convolutional layer while Z_l^i stands for the feature map of the $l-$th layer produced by the $i-$th filter. The 1D filter $W_{\text{conv}}^{l,i} \in \mathfrak{R}^g$ is used here.

After stacking L convolutional layers, the output of the last 1D convolutional layer is flattened to produce an input vector $Z_L \in \mathfrak{R}^r$ where r denotes the number of natural features extracted by the feature extraction part of ParsNet++. It is passed to a single hidden-layer neural network functioning to classify data samples into m target classes. ParsNet++ is underpinned by a closed-loop configuration between the generative and discriminative learning phases where the denoising autoencoder (DAE) [21] is implemented to extract robust input features. The DAE makes use of noise injecting mechanism avoiding the identity mapping issue while functioning as the regularization mechanism. The DAE takes the natural features Z_L and maps it into the latent space:

$$\begin{aligned} f_{\text{enc}} &= \text{Relu}\left(\widetilde{Z}_L W_{\text{enc}} + b\right), \\ \widehat{Z}_L &= f_{\text{dec}} = \text{Relu}\left(f_{\text{enc}} W_{\text{dec}} + c\right), \end{aligned} \tag{5}$$

where $W_{\text{enc}} \in \mathfrak{R}^{r\times j}$ and $b \in \mathfrak{R}^j$ are the connective weights and bias of the encoder while $W_{\text{dec}} \in \mathfrak{R}^{j\times r}$ and $c \in \mathfrak{R}^r$ are the connective weights and bias of the decoder. j denotes the number of hidden nodes. Note that W_{dec} is the inverse mapping of W_{enc} and is known as the tied-weight constraint. $\widetilde{Z}_L$ is a partially destroyed input vector where the masking noise is used here. That is, a subset of input vector is set blank. The Relu activation function $\max(0, x)$ is used here instead of the sigmoid activation function. The discriminative phase utilizes a softmax function $\text{softmax}(x) = (\exp x / \sum_{i=1}^m \exp x)$ to produce the output class posterior probability:

$$\widehat{y} = P(Y|X) = \text{softmax}\left(\text{Relu}\left(Z_l W_{\text{in}} + b_{\text{in}}\right) W_{\text{out}} + d\right), \tag{6}$$

where $W_{\text{out}} \in \mathfrak{R}^{j\times m}, d \in \mathfrak{R}^m$ are the connective weights and bias of the softmax layer. ParsNet++ utilizes shared network parameters between the generative and discriminative phases where $W_{\text{enc}} = W_{\text{in}}, b_{\text{in}} = b$. Both phases are carried out in the closed-loop fashion where a model is firstly trained during the generative phase with the absence of ground truth. The discriminative phase further refines it with the presence of class labels.

3.3. Growing and Pruning of Hidden Nodes. ParsNet++'s structural evolution is governed by the network significance (NS) method estimating the network bias and variance in the one-pass learning fashion. M new hidden nodes are added if the network experiences high bias condition whereas the hidden node pruning mechanism is triggered in the case of high variance. M stands for the number of clusters generated using the autonomous clustering mechanism. It is worth mentioning that both mechanisms are carried in the generative and discriminative fashions where the bias and variance are enumerated in respect to the predictive error while the reconstruction error is referred to during the generative phase. We only present the structural learning mechanism in the discriminative phase here for the sake of simplicity but the same step can be followed for the generative phase. The NS method can be expressed as follows:

$$\text{NS} = \left(E\left[\widehat{y}^2\right] - E[\widehat{y}]^2\right) + (E[\widehat{y}] - y)^2 = \text{Var} + \text{Bias}^2. \tag{7}$$

The key for solving (7) lies in the expected output $E[\widehat{y}]$. ACM is applied here to estimate the complex probability function $p(x)$ and results in the following expression:

$$E[\widehat{y}] = W_{\text{out}} \sum_{m=1}^{M} \int_{-\infty}^{+\infty} \text{relu}\left(Z_l W_{\text{in}} + b_{\text{in}}\right) N(X; \omega_m, c_m) \text{d}x, \tag{8}$$

where ω_m, c_m, respectively, denote the $m-$th mixing coefficient and center of clusters, respectively. Equation (8) can be derived independently for each cluster while the overall expected output is enumerated by applying the mixing coefficient ω_m taking into account the contribution of each cluster to the overall estimation. This step leads to the following expression:

$$E[\widehat{y}] = W_{\text{out}} \sum_{m=1}^{M} \omega_m \left(c_m W_{\text{in}} + b_{\text{in}}\right), \tag{9}$$

where $\sum_{m=1}^M \omega_m = 1$ meets the partition of unity property. On the other hand, the term $E[\widehat{y}^2]$ is derived under the i.i.d condition leading to $E[\widehat{y}^2] = E[\widehat{y}]E[\widehat{y}]$.

The hidden unit growing condition is formulated using the statistical process control (SPC) method [22] as follows:

$$\begin{aligned} \mu_{\text{bias}}^t + \sigma_{\text{bias}}^t &\geq \mu_{\text{bias}}^{\min} + \sigma_{\text{bias}}^{\min}, \\ k_1 &= 1.2\ \exp\left(-\text{Bias}^2\right) + 0.8, \end{aligned} \tag{10}$$

where $\mu_{\text{bias}}^t, \sigma_{\text{bias}}^t$ are the empirical mean and standard deviation of the network bias while $\mu_{\text{bias}}^{\min}, \sigma_{\text{bias}}^{\min}$ are the minimum network bias up to the $t-$th time instant. $\mu_{\text{bias}}^{\min}, \sigma_{\text{bias}}^{\min}$ are reset once (10) is satisfied while $\mu_{\text{bias}}^t, \sigma_{\text{bias}}^t$ are calculated across all samples because of the nature of bias estimation being accurate when considering all samples. Formula (10) is meant to detect the high bias condition leading to the hidden unit growing condition. Note that the SPC method in essence functions to detect anomalous points or a drifting concept. The original SPC method is, however, modified here to induce the flexible confidence level with the use of $k_1 \in [1, 2]$ being equivalent to the confidence degree between 68.2% and 95.2%. It implies the hidden unit growing process to be carried out in the case of high bias whereas it is hindered in the case of low bias.

As with the hidden unit growing mechanism, the hidden unit pruning strategy is undertaken using the SPC method as follows:

$$\mu_{\text{var}}^{t} + \sigma_{\text{var}}^{t} \geq \mu_{\text{var}}^{\min} + 2 * k_2 * \sigma_{\text{var}}^{\min}, \quad k_2 = 1.2 \exp\left(-\text{var}^2\right) + 0.8. \tag{11}$$

The key difference lies in the term 2 directed to avoid a direct-pruning-after-adding situation. This leads to the confidence level between 68.2% and 99.9%. That is, the hidden unit pruning condition is carried out frequently in the case of high variance while the hidden unit pruning situation is prevented in the case of low variance. Once (11) is met, the hidden unit pruning condition is executed as follows:

$$E[s] \leq \mu_{E[s]} - 0.5\sigma_{E[s]}, \tag{12}$$

where $E[s] = \sum_{m=1}^{M} \omega_m (c_m W_{\text{in}} + b_{\text{in}})$ denotes the statistical approximation of hidden nodes. Equation (12) enables multiple hidden nodes to be discarded at once and results in rapid complexity reduction.

3.4. Autonomous Clustering Mechanism. ParsNet++ is guided by autonomous clustering mechanism (ACM) to generate a complex probability density function $p(x)$ during the hidden node growing and pruning processes. It differs from the original ParsNet [17] where autonomous Gaussian mixture model (AGMM) is applied. The bottleneck of AGMM exists in the high input dimension often being unstable. ACM features an open structure where clusters are added or discarded on the fly to cope with the concept drifts $p(x)_t \neq p(x)_{t+1}$ and is capable of initiating its learning process from scratch. The component's growing process is governed by the compatibility measure examining the spatial proximity of a data point to existing clusters whether it is within the cluster's coverage. The cluster pruning technique makes use of the cluster's utility checking the cluster's activity during its lifespan.

Suppose that $D(X, Y)$ is the $L-2$ distance between two data samples; the compatibility test is formulated:

$$D\left(Z_L, C_{\text{win}}\right) > \mu_D + k_3\sigma_D, \tag{13}$$

where $k_3 = 2\exp\left(-D(Z_L, C_{\text{win}})^2\right) + 1$. μ_D, σ_D stand for the mean and standard deviation of distance calculation $D(Z_L, C_{\text{win}})$. As with (10) and (11), (13) is formalised by the statistical process control (SPC) method. The use of k_3 controls the cluster's growing process in such a way that the growing process is performed frequently if a sample is remote to the existing cluster $k_3 \approx 2$. This situation portrays a fact where a data sample is uncovered by existing clusters. On the other hand, this condition is difficult to be fulfilled if a data sample is adjacent to existing clusters, i.e., low clustering loss $k_3 \approx 3$. A new cluster is constructed if (13) is satisfied. That is, the cluster center is set as the sample of interest $C_{M+1} = Z_L$ with $\text{Car}_{M+1} = 1$ where M is the number of clusters. If (13) is violated, the winning cluster is fine-tuned:

$$c_{j,m} = c_{j,m} + \frac{\left(X - c_{j,m}\right)}{\text{Car}_m + 1}, \quad \text{Car}_m = \text{Car}_m + 1, \tag{14}$$

where Car_m denotes the cluster's cardinality. Note that the adaptation process is localized only to the winning cluster to avoid the cluster's overlapping case and associates the data sample of interest to the winning cluster. That is, the cluster's cardinality is incremented here. Equation (14) ensures the cluster's convergence as the factor of the cluster's cardinality.

The cluster pruning procedure is implemented to prevent the issue of cluster's explosion due to the problem of outliers. That is, outliers are wrongly inserted as clusters by (13). It checks the cluster's significance whether it plays a major role during its lifespan. A cluster can be pruned without loss of generalization if it plays little during their lifespan. The cluster's contribution is examined from the average of cluster activity as follows:

$$\text{Act}_m = \frac{\sum_{n=1}^{\text{Life}} \Phi_m}{\text{Life}}, \tag{15}$$

where $\Phi_m = \exp\left(-(Z_L - C_m)^2\right)$ measures the spatial proximity of a data sample to the cluster of interest in the latent space while Life denotes the time period of a cluster since it is added. Furthermore, the unity variance is assumed in calculating Φ_m where $\sigma^2 = 1$. The cluster pruning mechanism is executed as follows:

$$\text{Act}_m \leq \left|\mu_{\text{Act}_m} - 0.5 * \sigma_{\text{Act}_m}\right|. \tag{16}$$

The cluster pruning mechanism enables more than one cluster to be discarded at once leading to rapid complexity reduction and follows the half-sigma rule. Furthermore, the number of clusters M is also used as an addition factor in the network growing phase (10) because the clustering mechanism explores the true data distribution. As an implementation note, the monitoring period is applied here. That is, a cluster is not removed during the monitoring period to evolve its shape. On the other hand, the mixing coefficient, ω_m, is formed as the relative cardinality as follows:

$$\omega_m = \frac{\Phi_m^* \text{Car}_m}{\sum_{m=1}^{M} \Phi_m^* \text{Car}_m}, \tag{17}$$

where it features the partition of unity property and takes into account both the distance information and the cluster support. A cluster should possess high influence in the network bias and variance estimation if it is adjacent to the data sample of interest and has high population.

4. Injection Molding Process

eScentz, as shown in Figure 1, is a scent-emitting USB device made by SIMTech. It is used as the testbed product at the model factory@SIMTech. The injection moulding process is used to manufacture the black cartridge, white cartridge holder, and a transparent part which is used to contain the

Figure 1: eScentz, scent-emitting USB device made by SIMTech.

scent in the cartridge. The injection molding machine (Arburg Allrounder 470 A) is shown in Figure 2.

Focus is on the transparent part as it is critical to the functionality of the device; i.e., defects in the part can lead to leaking of the liquid scent. There are a number of different types of possible defects but the most common ones are flow lines which is a mark or line formed when two melt flow fronts meet during the filling of the injection mold and short shot where the mold is partially filled with plastic melt [23]. Examples of a good part and the different types of defects are shown in Figure 3.

5. Numerical Study

This section demonstrates the advantage of ParsNet++ in assessing the quality of transparent mold manufactured by the injection molding machine. ParsNet++ is simulated in two simulation environments: random access of ground truth and infinitely delayed access of ground truth. The former one describes a case where each data batch contains partially labelled data points with unknown class distribution while the latter one portrays a semisupervised problem where ground truth is accessed only in the initial phase leaving the rest unlabelled. 50% of labelled samples are set as the default setting for the random access of ground truth. The infinitely delayed access of ground truth only utilizes the first data batch. Furthermore, both scenarios are simulated in the prediction of current batch as well as future batch. The prediction of current batch monitors the current quality of transparent molds Y_k based on the sensor data X_k. The prediction of future batch relies on the current data batch to forecast the future product quality Y_{k+1}. The contribution of each learning module is studied in the ablation study section while the effect of label proportions is elaborated. Our numerical study follows the prequential test-then-train procedure, the standard evaluation protocol of data stream mining. Moreover, the t-test is put forward to statistically validate the numerical results.

5.1. Baselines. The numerical results of ParsNet++ are benchmarked against recently published algorithms in the literature:

(i) *Online deep learning (ODL)* [24] is an online learning algorithm constructed under the vanilla neural network structure. It makes use of the hedging idea where there exists a direct connection of the hidden layer to the output layer.

Figure 2: Injection molding machine used to collect experimental data at the model factory@SIMTech.

(ii) *Neural networks with dynamically evolved capacity (NADINE)* [18] adopts a flexible network structure under the multilayer perceptron (MLP) architecture. That is, both of hidden layers and nodes are dynamically grown and reduced in respect to variations of data streams.

(iii) *Parsimonious network (ParsNet)* [17] is perceived as a predecessor of ParsNet++. ParsNet++ distinguishes itself of ParsNet with the presence of feature extraction layer crafted under the convolutional framework; 1D CNN is integrated to handle raw input features. In addition, ParsNet++ is underpinned by the ACM rather than AGMM to perform density estimation on the fly.

(iv) *SCARGC* [25] is devised for the infinite delay problem and considered as a state-of-the art algorithm in this domain. It utilizes the pool-based principle.

Since these algorithms are not designed to handle visual data of high dimension, their predictions are only guided by sensory data $X^{\text{sen}} \in \Re^{48}$. The use of image data significantly reduces its performance due to the absence of the feature extraction layer. In addition, comparison is also made against two popular deep learning algorithms, *ResNet18* [26] and *VGG11* [27], only using the image data happening to be an RGB image with a size of $X^{\text{img}} \in \Re^{150\times150\times3}$. They do not exploit the sensor data due to the absence of 1-D CNN. All of the algorithms except ParsNet and SCARGC are a fully supervised algorithm. The simplest structure of ResNet and VGG is adopted here because of the low data size leading to the issue of overfitting. All algorithms are executed under the same computational platform by using their published codes and run under the same simulation protocol as ParsNet++ to ensure fair comparison. The numerical results are taken from the average of five consecutive runs.

5.2. Network Structure and Hyperparameters. ParsNet++ utilizes 1D CNN as a feature extractor to predict the mold quality Y_k where 1D CNN looks after the raw sensory data. Extracted features from the CNN are concatenated into a long vector and fed to the fully connected layer, a single-hidden-layer neural network with the self-evolving property.

(a) (b) (c)

Figure 3: Different types of defects in the transparent part used to seal the liquid scent in the cartridge. (a) Normal. (b) Flow lines. (c) Short shot.

1D CNN is developed from 3 convolutional layers underpinned by the 1D filter. The number of input and output channels across the three layers is, respectively, set as [48, 60], [60, 40], and [40, 20] for injection molding dataset. For transfer molding dataset, two-layer 1D CNN has been applied as feature extractor, in which the input and output channels are [608, 8] and [8, 4], respectively.

The hyperparameters of ParsNet++ are fixed throughout our simulation scenario as $\alpha_2 = 0.6$ and $\alpha_3 = 0.8$ while the learning rates and momentum coefficient are selected as 0.01 and 0.95 of stochastic gradient descent optimizer (SGD). Hyperparameters of other algorithms are chosen as those reported in their original papers. We chose 100 as the batch size for all algorithms. Table 2 reports the hyperparameters of consolidated algorithms. For injection molding dataset, initialization batch S and epochs E, shown in Algorithm 1, are 5 and 10 in sporadic access experiment and 1 and 15 in infinite delay experiment. For transfer molding dataset, S and epochs are 1 for both sporadic access and infinite delay experiment which also signify that ParsNet++ runs in the single pass way.

5.3. Numerical Results. Table 3 reports our numerical results for the current batch prediction under the setting of sporadic access of ground truth. It is evident that ParsNet++ outperforms ParsNet with significant gap. This finding clearly encourages the 1D CNN of ParsNet++ automatically extracting deep natural features and the ACM technique for estimation of probability density function. Moreover, ParsNet++ beats NADINE, ODL happening to be a fully supervised algorithm with significant margin. Note that ParsNet, ODL, and NADINE are guided by sensor data as with ParsNet++. ParsNet++ is compared with ResNet18 and VGG11 making use of image data and being popular deep learning approaches. Although the two approaches are an offline algorithm trained in the offline fashion and are fully supervised, ParsNet++ exhibits superior performances. That is, ParsNet++ exceeds VGG11 and ResNet18 with noticeable difference. This result is confirmed with the statistical test in Table 4 where the performance gap between ParsNet++ against all algorithms is statistically significant.

Table 5 exhibits our consolidated numerical results for the next batch prediction. The same finding as the current batch prediction is found here where ParsNet++ beats ParsNet with significant performance gap. This facet substantiates the advantage of feature extraction module of ParsNet++ generating deep natural features while ACM approximates the true probability distribution better than the AGMM of ParsNet. By extension, ParsNet++ outperforms fully supervised algorithms, NADINE and ODL, working with more favourable condition than ParsNet++. NADINE, ODL, and ParsNet are akin to ParsNet++ where raw sensor data are exploited as input features but suffer from the absence of feature extraction layer. Our numerical results are statistically validated with the statistical test in Table 6 where ParsNet++'s performance is statistically better than its competitors.

In realm of infinitely delayed access of ground truth, ParsNet++ delivers superior performance with almost 40% improvement from ParsNet and SCARGC. ParsNet++'s accuracy is 85.60% for the current batch prediction and 83.25% for the next batch prediction whereas its counterparts deliver the accuracy below 50%. This mechanism confirms the generalization power of ParsNet++ in dealing with various semisupervised learning situations. These numerical results are presented in Tables 6 and 7. Note that the true class labels are only supplied in the initial batch for the infinite delay case being more challenging condition than the sporadic access case. This facet is confirmed by the fact where numerical results of all algorithms worsen. Figure 4 visualizes the predictive quality of ParsNet++ where precision, recall, and F_1 metrics show similar trend. This observation signifies the fact that ParsNeT++ handles all target classes equally well. The detailed numerical results are presented in Table 8. In addition, this figure also depicts the dynamic nature of ParsNet++ in which its hidden nodes are dynamically added and pruned on the fly. It is also observed that ParsNet++ timely responds on performance decrease as a result of concept drifts. That is, new nodes are injected if network's performance is compromised in the case of concept drift.

5.4. Ablation Study. The ablation study is carried out to validate the influence of each learning module of ParsNet++. ParsNet++ is configured into three variations: (A) ParsNet++ is set with only the parameter learning scenario using the stochastic gradient descent method with the absence of other

Table 2: Hyperparameters of the model.

Model	Learning rate	Others
ParsNet++	0.01	Momentum: 0.95, $\alpha_2 = 0.6$, $\alpha_3 = 0.8$, $\varepsilon = 0.001$
ParsNet	0.01	Momentum: 0.95, confidence level of network: 0.6, confidence level of AGMM: 0.55
ODL	0.01	$\beta = 0.99$
SCARGC	—	Number of clusters: 100, pool size: 300
NADINE	Dynamic between [0.001, 0.02]	Momentum: 0.95
ResNet18 and VGG11	0.01	Momentum: 0.95

Table 3: Classification performance on injection molding dataset of sporadic access current batch prediction scenario.

Model	Accuracy
ParsNet++	0.9136 ± 0.0061
ParsNet	0.8214 ± 0.0116
NADINE*	0.7690 ± 0.0228
ODL*	0.8040 ± 0.0001
ResNet18#	0.8650 ± 0.0085
VGG11#	0.8410 ± 0.0099

Note: * indicates that the numerical results of the corresponding baseline used fully supervised sensor data; # indicates that the numerical results of the corresponding baseline exploit fully supervised image data.

Table 4: *t*-test result on injection molding dataset.

Scenario	Model 1	Model 2	*T* value	*P* value
Sporadic access (current batch prediction)	ParsNet++	ParsNet	15.7305	2.66*e*−07
	ParsNet++	NADINE	13.6995	7.77*e*−07
	ParsNet++	ODL	40.1705	1.62*e*−10
	ParsNet++	ResNet18	10.3871	6.39*e*−06
	ParsNet++	VGG11	13.9605	6.72*e*−07
Sporadic access (next batch prediction)	ParsNet++	ParsNet	16.7425	1.64*e*−07
	ParsNet++	NADINE	26.3042	4.69*e*−09
	ParsNet++	ODL	11.328168	3.32*E*−06
Infinity delay (current batch prediction)	ParsNet++	ParsNet	33.8308	6.37*e*−10
	ParsNet++	SCARGC-SVM	44.5193	7.15*e*−11
	ParsNet++	SCARGC-1NN	32.4618	8.84*e*−10
Infinity delay (next batch prediction)	ParsNet++	ParsNet	18.9618	6.19*e*−08
	ParsNet++	SCARGC-SVM	23.5353	1.13*e*−08
	ParsNet++	SCARGC-1NN	28.3756	2.57*e*−09

Table 5: Classification performance on injection molding dataset of sporadic access next batch prediction scenario.

Model	Accuracy
ParsNet++	0.9204 ± 0.0129
ParsNet	0.7968 ± 0.0103
NADINE*	0.7646 ± 0.0003
ODL*	0.7949 ± 0.0002

Note: * indicates that the numerical results of the corresponding baseline are fully supervised.

Table 6: Classification performance on injection molding dataset of infinity delay current batch prediction scenario.

Model	Accuracy
ParsNet++	0.8560 ± 0.0182
ParsNet	0.4740 ± 0.0175
SCARGC-SVM	0.3877 ± 0.0149
SCARGC-1NN	0.3466 ± 0.0300

learning modules. That is, the label augmentation mechanism, the dynamic regularization mechanism, and the structural learning mechanism are deactivated; (B) ParsNet++ is equipped by the label enrichment mechanism and the dynamic regularization mechanism but with the absence of structural learning method; (C) the structural learning mechanism of ParsNet++ is switched on but without the pseudolabel generation step and the dynamic regularization mechanism. Our

Table 7: Classification performance on injection molding dataset of infinity delay next batch prediction scenario.

Model	Accuracy
ParsNet++	0.8325 ± 0.0348
ParsNet	0.3943 ± 0.0382
SCARGC-SVM	0.4014 ± 0.0216
SCARGC-1NN	0.3536 ± 0.0146

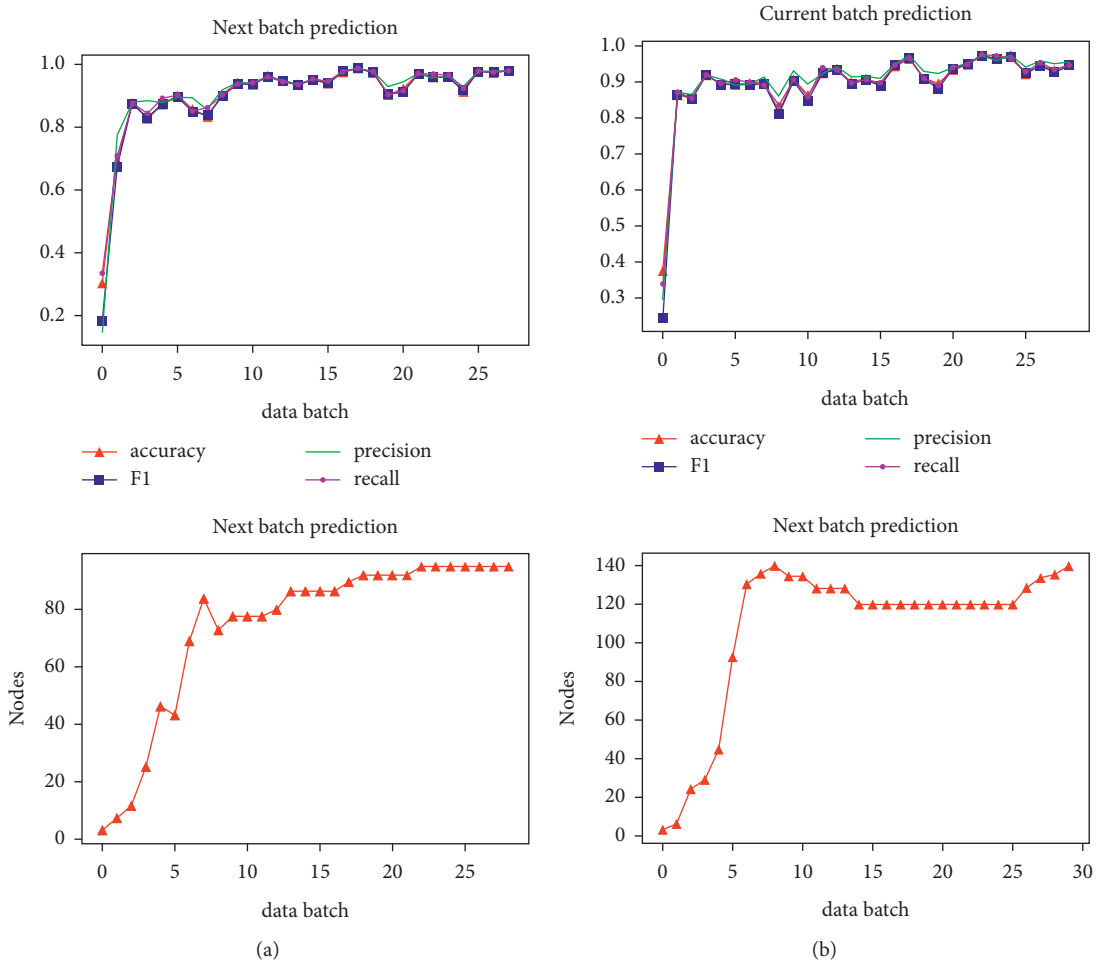


Figure 4: Comparison of classification metrics on injection molding dataset. (a) Sporadic access on current batch prediction and (b) sporadic access on next batch prediction.

Table 8: Classification metrics injection molding dataset of ParsNet++.

Experiment	F_1	Precision	Recall
Sporadic access-current	0.9122 ± 0.0065	0.9254 ± 0.0054	0.9170 ± 0.0062
Sporadic access-next	0.9201 ± 0.0132	0.9323 ± 0.0075	0.9252 ± 0.0126
Infinity delay-current	0.8517 ± 0.0194	0.8749 ± 0.0144	0.8607 ± 0.0185
Infinity delay-next	0.8251 ± 0.0411	0.8559 ± 0.0443	0.8410 ± 0.0322

numerical results are produced under future batch prediction, all of which are executed under the sporadic access of ground truth with 50% labelled samples. Table 9 exhibits our numerical results.

It is observed that the worst-performing result comes from the model (A) where all mechanisms are turned off. The label enrichment mechanism and the dynamic regularization mechanism enhance the performance by almost

Table 9: Classification performance on ablation study of sporadic access on next batch prediction of injection molding dataset.

	Accuracy	F_1 score	Nodes
A	0.7951 ± 0.0906	0.7608 ± 0.1376	—
B	0.8875 ± 0.0112	0.8815 ± 0.0141	—
C	0.9172 ± 0.0182	0.9156 ± 0.0190	128.4 ± 46.7
ParsNet++	0.9204 ± 0.0129	0.9201 ± 0.0132	96.4 ± 26.3

10% as reported by Model (B). This fact clearly demonstrates the advantage of these learning strategies in coping with the issue of label's scarcity. Noticeable performance improvement is attained using the structural learning mechanism clearly confirming the advantage of a dynamic structure from that of a static structure as shown in Model (C). This case portrays the importance of drift handling mechanism when handling the problem of data streams. Note that Model (C) excludes the pseudolabel generation mechanism and the dynamic regularization approach. The numerical result increases further when combining the self-evolving structure, the label enrichment mechanism, and the dynamic regularization mechanism as exemplified by ParsNet++ configuration. This configuration enables the issue of label scarcity and concept drift to be simultaneously overcome.

5.5. Effect of Label Proportions. This section examines the learning performance of ParsNet++ under different label proportions. That is, ParsNet++'s performance is evaluated under seven label proportions: 10%, 20%, 30%, 40%, 50%, 60%, 70%. The simulation protocol follows the sporadic access of ground truth in which two evaluation metrics, accuracy and F_1, are applied. Table 10 reports the average numerical results across five independent runs.

Our numerical results show that ParsNet++'s performance is compromised with only 5% of labelled samples. The increase of label proportions improves its learning performance and this trend does not continue after 50% label proportion. The best-performing result is achieved with 50% labelled samples whereas performance's deterioration is observed with 60% and 70% labelled samples compared to that of 50% labelled samples. This finding demonstrates that the increase of labelled samples does not ensure the performance's improvement. The performance deterioration with 60% and 70% labels results from the issue of sample redundancy due to the consistency regularization step in which small perturbations are injected to original samples without changing their labels. The consistency regularization method might lead to the issue of overfitting if the proportion of labelled samples is high. That is, it produces indistinguishable samples which slightly affect model's generalization. Note that the 50%, 60% cases are better than the 40% case.

5.6. Industrial Transfer Molding Process. The industrial transfer molding process portrays a process from a semiconductor industry occurring in the encapsulation stage where a batch of integrated circuits (ICs) are packaged in a case to avoid corrosion and physical damage [15]. The quality monitoring step in this phase plays a key role because it might result in heavy penalties if defective products are sent to the customer. The encapsulation process makes use of an industrial transfer molding machine, very similar to the injection molding machine where it is used to form the support of electronic components. That is, the transfer molding is a process whereby the casting material is entered into the mold [15].

Table 10: Classification performance of ParsNet++ in the injection molding problem with different label proportions.

Labeled percentage (%)	Accuracy	F_1 score
10	0.8047 ± 0.0291	0.7834 ± 0.0422
20	0.8509 ± 0.0354	0.8444 ± 0.0420
30	0.8805 ± 0.0161	0.8773 ± 0.0194
40	0.8847 ± 0.0189	0.8830 ± 0.0213
50	0.9204 ± 0.0129	0.9201 ± 0.0132
60	0.8969 ± 0.0232	0.8939 ± 0.0246
70	0.9035 ± 0.0133	0.9032 ± 0.0152

Each production is undertaken in lot sizes having 1 to 424 strips where each strip comprises a number of products. The product quality is examined only after the complete lot has been finished. The goal of this problem is to feed real-time prediction of the product quality while being still in production. The use of artificial intelligence (AI) is urgently required because it enables redundancy in checking such that the product's integrity is ensured. We collected production data over the period of six months. This problem is formulated as a binary classification problem and suffers from the class imbalanced problem where only 4% of data contains defects while the remainder is of the normal class. The unique property of this problem lies in the many-to-one label relationship where multiple instances are assigned with a single label. That is, the quality of product is not determined from a single product quality rather the whole lot. If a lot happens to have over 48 defects, the whole lot is thrown away or this case portrays the defect case.

Our numerical study follows the prequential test-then-train protocol as the injection molding problem where one step ahead prediction is simulated. That is, a model is used to predict the quality of next lot based on the current machine parameters and process variables. Both the sporadic access of ground truth and the infinitely delayed access of ground truth are simulated here. Important parameters of the moulding process include cavity pressure, ram velocity, ram position, and mould temperatures. This problem is a high-dimensional problem with 608 input features. Tables 11 and 12 report our numerical results for both the sporadic access and the infinite delay scenarios. ParsNet++ is compared with ParsNet and SCARGC. Since this problem suffers from the many-to-one label relationship where many data samples are associated with a single class label, a simple mean operation is executed for the feature extraction strategy in ParsNet and SCARGC. Note that this case is not applicable for ParsNet++ because the use of 1DCNN enables automatic feature engineering where data points of each lot/strip are scanned using 1D filter.

Table 11: Classification performance on transfer molding dataset for next batch prediction.

Model	Accuracy	F_1	Nodes
ParsNet++	0.9566 ± 0.0004	0.4888 ± 0.0001	33.0 ± 24.36
ParsNet	0.9577 ± 0.0001	0.4892 ± 0.0001	20.94 ± 2.64

Table 12: Classification performance on transfer molding dataset for infinite delay next batch prediction.

Model	Accuracy	F_1	Nodes
ParsNet++	0.9582 ± 0.0000	0.4892 ± 0.0000	7.00 ± 2.19
ParsNet	0.8844 ± 0.0310	0.5052 ± 0.0098	22.03 ± 2.91
SCARGC-SVM	0.9227	0.5023	—
SCARGC-NN	0.8959	0.4974	—

It is obvious from Table 11 that ParsNet++ and ParsNet exhibit comparable performance in the context of sporadic access protocol. This finding is supported by the fact of class imbalance where only 4% of data samples belong to the positive class. On the contrary, ParsNet++ outperforms both SCARGC and ParsNet in the case of infinite delay as shown in Table 12. This observation substantiates ParsNet++ generalization power in coping with different scenarios of semisupervised learning. Note that the infinite delay problem is more challenging than the sporadic access problem because true class labels are supplied only in the warm-up phase. This issue leads to performance degradation of ParsNet and SCARGC where the automatic feature engineering step is absent; i.e., features are extracted by applying the mean operation. Nonetheless, we acknowledge that the class imbalance problem still deserves in-depth future study. It is seen from the low F_1 scores of consolidated algorithms.

5.7. Sensitivity Analysis. This subsection aims to study the effect of hyperparameters to the performance of ParsNet++. Specifically, the effect of α_2, α_3 is analyzed while excluding other parameters. Other hyperparameters such as momentum coefficient and learning rate are set to the same values for all consolidated algorithms. In addition, they are default parameters of SGD method where their effects have been well-understood from the literature. ε is merely a small constant to avoid division with zero. The sensitivity analysis is carried out by varying $\alpha_2 = [0.2, 0.4, 0.6, 0.8]$ and $\alpha_3 = [0.2, 0.4, 0.6, 0.8]$. Table 13 reports the numerical results of all combinations. α_2, α_3 are required to be set higher than 0.55; therefore, 0.6 and 0.8 are selected for α_2, α_3, respectively. Note that we do not apply specific hyper-parameter selection in our main experiments. That is, only simple hand-tuning is applied to set the parameters. Our sensitivity analysis is undertaken in the case of sporadic access of ground truth under the next batch prediction with 50% label proportion.

From Table 13, variation of α_2, α_3 does not lead to significant performance deterioration. That is, the difference between the worst and best results is around 2%. It is worth stressing that α_2, α_3 should be set higher than 0.55 since it reflects the confused prediction. This aspect should narrow down the choice of hyperparameters, i.e., $\alpha_2, \alpha_3 \in [0.2, 0.4]$ to be unreasonable values. Such a case leads to performance variation to be less than 1%. On the other hand, α_2, α_3 govern the pseudolabel generation where predictions of the network and the ACM are used to generate the pseudolabel. The case of $\alpha_2 = \alpha_3 = 0.2$ produces the worst result because it produces too many noisy pseudolabels. The increase of α_3 improves the prediction because it reduces the prediction's uncertainty of ACM. Note that the ACM's prediction relies on the class posterior probability $P(y_o|N_m)$ where it no longer represents the class distribution in the case of extreme label scarcity.

Table 13: Classification performance for sensitivity analysis of ParsNet++ in the next batch prediction.

	$\alpha_2 = 0.2$	$\alpha_2 = 0.4$	$\alpha_2 = 0.6$	$\alpha_2 = 0.8$
$\alpha_3 = 0.2$	0.9019 +/− 0.0218	0.9019 +/− 0.0218	0.9019 +/− 0.0218	0.9204 +/− 0.0116
$\alpha_3 = 0.4$	0.9055 +/− 0.0204	0.9055 +/− 0.0204	0.9055 +/− 0.0204	0.9182 +/− 0.0139
$\alpha_3 = 0.6$	0.9123 +/− 0.0146	0.9123 +/− 0.0146	0.9123 +/− 0.0146	0.9135 +/− 0.0149
$\alpha_3 = 0.8$	0.9204 +/− 0.0129	0.9204 +/− 0.0129	**0.9204 +/− 0.0129**	0.9183 +/− 0.0124

Bold value shows the parameters that we used in the experiment.

6. Conclusion

A semisupervised quality classification in data stream environments including its deep learning solution termed Parsimonious Networks++ (ParsNet++) is presented in this paper. ParsNet++ features an open structure automatically generating and pruning its hidden nodes on the fly thereby addressing concept drifts of partially labelled data streams. The parameter learning strategy is formulated as a joint optimization problem of the reconstruction loss, the predictive loss of the original label, the predictive loss of the augmented label, and the predictive loss of pseudolabel. In addition, the regularization strategy is put forward to combat the noisy pseudolabel problem preventing the important parameters to be perturbed by the noisy pseudolabels. ParsNet++ extends ParsNet with the integration of feature extraction layer enabling automatic feature engineering mechanism. 1D CNN is integrated to perform the automatic feature engineering step and to handle the many-to-one label relationship while incorporating the ACM for flexible density estimation approach. Comprehensive experiments with the injection molding machine and the industrial transfer molding machine have been carried out to experimentally validate the advantage of ParsNet++. ParsNet++ is tested in two semisupervised learning scenarios: infinite delay access of ground truth and random access of ground truth with comparisons against prominent algorithms for both current batch quality monitoring and future batch quality monitoring. ParsNet++ outperforms its

counterparts with noticeable margin and delivers comparable accuracy to those of fully supervised learning algorithms. There are few important issues unexplored in ParsNet++. The issue of class imbalance still deserves an in-depth study where this issue requires a specific strategy in order to reduce false positive rates of ParsNet++'s prediction. This aspect is seen in ParsNet++'s results of the industrial transfer molding problem where the F_1 score is rather low. Another uncharted area lies in the issue of transferability to different machines. Its solution makes possible to utilize a single model to be transferred across different machines of the same types or different types with little capital expenditure.

Data Availability

Codes and data of this paper can be found in https://github.com/ContinualAL/ParsNetPlus.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Weng Weiwei and Mahardhika Pratama contributed equally to this study.

Acknowledgments

This project was financially supported by National Research Foundation, Republic of Singapore, under IAFPP in the AME domain (contract no. A19C1A0018).

References

[1] E. P. Carden and P. Fanning, "Vibration based condition monitoring: a review," *Structural Health Monitoring*, vol. 3, no. 4, pp. 355–377, 2004.

[2] S. Ravikumar, K. I. Ramachandran, and V. Sugumaran, "Machine learning approach for automated visual inspection of machine components," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3260–3266, 2011.

[3] D. E. Dimla and P. M. Lister, "On-line metal cutting tool condition monitoring," *International Journal of Machine Tools and Manufacture*, vol. 40, no. 5, pp. 739–768, 2000.

[4] X. Li, B. Lim, J. Zhou et al., "Fuzzy neural network modelling for tool wear estimation in dry milling operation," in *Proceedings of the Annual Conference of The Prognostics and Health Management Society*, pp. 1–11, San Diego, CA, USA, October 2009.

[5] F. Serdio, E. Lughofer, K. Pichler, T. Buchegger, and H. Efendic, "Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills," *Information Sciences*, vol. 259, 2014.

[6] F. Serdio, E. Lughofer, K. Pichler, M. Pichler, T. Buchegger, and H. Efendic, "Fuzzy fault isolation using gradient information and quality criteria from system identification models," *Information Sciences*, vol. 316, pp. 18–39, 2015.

[7] C. González-Val, A. Pallas, V. Panadeiro, and A. Rodriguez, "A convolutional approach to quality monitoring for laser manufacturing," *Journal of Intelligent Manufacturing*, vol. 31, no. 3, pp. 789–795, 2020.

[8] Y. Zhang, D. You, X. Gao, C. Wang, Y. Li, and P. P. Gao, "Real-time monitoring of high-power disk laser welding statuses based on deep learning framework," *Journal of Intelligent Manufacturing*, vol. 31, no. 4, pp. 799–814, 2020.

[9] E. Lughofer, R. Pollak, A.-C. Zavoianu et al., "Self-adaptive evolving forecast models with incremental PLS space updating for on-line prediction of micro-fluidic chip quality," *Engineering Applications of Artificial Intelligence*, vol. 68, pp. 131–151, 2018.

[10] E. Lughofer, A.-C. Zavoianu, R. Pollak et al., "Autonomous supervision and optimization of product quality in a multi-stage manufacturing process based on self-adaptive prediction models," *Journal of Process Control*, vol. 76, pp. 27–45, 2019.

[11] M. Pratama, E. Dimla, T. Tjahjowidodo, E. Lughofer, and W. Pedrycz, "Online tool condition monitoring based on parsimonious ensemble+," *IEEE Transactions on Cybernetics On-Line And In Press*, vol. 50, no. 2, 2018.

[12] X. Yuan, C. Ou, Y. Wang, C. Yang, and W. Gui, "A novel semi-supervised pre-training strategy for deep networks and its application for quality variable prediction in industrial processes," *Chemical Engineering Science*, vol. 217, no. 115, p. 509, 2020.

[13] X. Yuan, Z. Ge, B. Huang, Z. Song, and Y. Wang, "Semi-supervised JITL framework for nonlinear industrial soft sensing based on locally semisupervised weighted PCR," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 532–541, 2017.

[14] W. Caesarendra, M. Pratama, B. Kosasih, T. Tjahjowidodo, and A. Glowacz, "Parsimonious network based on a fuzzy inference system (PANFIS) for time series feature prediction of low speed slew bearing prognosis," *Applied Sciences*, vol. 8, no. 12, Article ID 2656, 2018.

[15] K. J. Lee, E. K. Yee Yapp, and X. Li, "Unsupervised probability matching for quality estimation with partial information in a multiple-instances, single-output scenario," in *Proceedings of the 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1432–1437, Kristiansand, Norway, November 2020.

[16] A. Ashfahani, M. Pratama, E. Lughofer, and E. Yapp Kien Yee, "Autonomous deep quality monitoringin streaming environments," 2021, http://arxiv.org/abs/14091556.

[17] M. Pratama, A. Ashfahani, and M. A. Hady, "Weakly supervised deep learning approach in streaming environments," in *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data)*, pp. 1195–1202, IEEE, Los Angeles, CA, USA, December 2019.

[18] M. Pratama, C. Za'in, A. Ashfahani, Y. S. Ong, and W. Ding, "Automatic construction of multi-layer perceptron network from streaming examples," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1171–1180, Beijing, China, November 2019.

[19] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proceedings of the 34th International Conference on Machine Learning, PMLR, International Convention Centre*, D. Precup and Y. W. Teh, Eds., vol. 70, pp. 3987–3995, Proceedings of Machine Learning Research, Sydney, Australia, August 2017.

[20] B. Vigdor and B. Lerner, "The bayesian artmap," *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1628–1644, 2007.

[21] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: learning

useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[22] J. Gama, *Knowledge Discovery From Data Streams*, Chapman & Hall/CRC, Boca Raton, FL, USA, 1st edition, 2010.

[23] D. V. Rosato, D. V. Rosato, and M. G. Rosato, *Injection Molding Handbook*, Springer, Boston, MA, USA, 2000.

[24] D. Sahoo, Q. Pham, J. Lu, and S. C. H. Hoi, "Online deep learning: learning deep neural networks on the fly," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 2660–2666, Stockholm, Sweden, July 2018.

[25] V. M. A. Souza, D. F. Silva, J. Gama, and G. E. A. P. A. Batista, "Data stream classification guided by clustering on nonstationary environments and extreme verification latency," in *Proceedings of the 2015 SIAM International Conference on Data Mining*, pp. 873–881, Vancouver, BC, Canada, April 2015.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference On Computer Vision And Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 2015.