ISSN: 2754-4915

Journal of Material Sciences & Manufacturing Research

SCIENTIFIC Research and Community

Research Article

Open Access

Using Transfer Learning to Reduce Preparing Time of Online Diagnosis with Limited Rotating-Component Data

Hao Tieng[1*], Yu-Yong Li[2], Ming-Xuan Liu[2], Wei-Chih Liao[2], Chien-Yuan Lai[2] and Sheng-Xiang Fan[2]

[1]e-Manufacturing Research Center (Associate Research Fellow), Innovation Headquarters, National Cheng Kung University, Tainan, Taiwan

[2]Institute of Manufacturing Information Systems, National Cheng Kung University, Tainan, Taiwan

ABSTRACT

Under the environment of mass-customization, valid samples for a specific target object become fewer and fewer. Thus, Transfer Learning (TL) is widely applied to those data-consuming Deep Learning (DL) and Machine Learning (ML) models to avoid recollecting high-volume training samples from scratch. However, TL-based methods used in fault diagnosis for mechanical rotating components are still in the infancy phase. Poor practicality and usability problems increase the total preparing time. Without online diagnosis, there might be more unknown risks during the preparing time of a TL method. To reduce the preparing time of existing TL techniques, this paper proposes a TL-based Gradual Improvement Scheme (TLGIS) to achieve a systematic online diagnosis of rotating components. TLGIS solves the problem through seamless interactions among five modules of Data Preprocessing (DP), Data Estimating (DE), Data Modeling (DM), Model Improving (MI), and Accuracy Checking, (AC). Via illustrative examples, TLGIS is demonstrated to be reliable and robust using deep Convolutional Neural Networks (CNN) by two public bearing datasets from the Internet and two types of rotating wheel datasets from a practical CNC machine. In total, TLGIS only requires at least 80 samples and at most 250 samples divided into small-size batches to gradually and continuously fine-tune the model with an accuracy of 90% and 99%, respectively, instead of inputting a large-scale dataset all at once, which is unrealistic in a practical production scenario. Compared to conventional approaches, TLGIS efficiently saves at most 90% of preparing time.

***Corresponding author**

Hao Tieng, e-Manufacturing Research Center (Associate Research Fellow), Innovation Headquarters, National Cheng Kung University, Tainan, Taiwan. E-mail: tienghao@imrc.ncku.edu.tw

Received: August 28, 2022; **Accepted:** September 06, 2022; **Published:** September 13, 2022

Keywords: Rotating Components, Transfer Learning (TL), Mass Customization, TL-Based Gradual Improvement Scheme (TLGIS), Deep Learning (DL)

Abbreviation List

AC: Accuracy checking.
AQ: Accumulating number for updating model.
Acc$_T$: Threshold of accuracy.
CT: Collection time for workpiece-sample.
CNN: Convolutional neural networks.
CWRU: Case western reserve university.
DAQ: Data acquisition.
DE: Data estimating.
DIST: Distance function.
DL: Deep learning.
DM: Data modeling.
DP: Data preprocessing.
DTW: Dynamic time warping.
EDM: Electrical-discharge machining.
FEMTO: FEMTO-ST Institute.
FFT: Fast Fourier transform.
fl: Number of frozen layers.
LP: Learning process.
MI: Model improving.
ML: Machine learning.
nl: Number of convolutional layers.
NOP: Non-optimized parameter.
PHM: Prognostics and health management.
PT: Total preparing time for Transfer Learning.
ReLu: Rectified linear unit.
RS: Ratio of samples.
RS$_T$: Threshold of RS.
SN$_{90}$: Sample number reaching accuracy 90%.
TL: Transfer learning.
TLGIS: TL-based gradual improvement scheme
TLI: Transfer level index.
TLI$_{LT}$: Lower threshold of TLI.
TLI$_{HT}$: Higher threshold of TLI.
WT: Wavelet transform.

List of Symbols in Equations

***f*(*t*):** *t*-second time-domain sensing signal.
X: Discrete signal of *f*(*t*).
***x*[*l*]**: *l*th sampling data point of **X**.
D$_s$: Dataset of **X$_s$** and **Y$_s$** in source domain.
X$_s$(*p*): *p*th process data of **X$_s$**.
***y$_b$*(*p*)**: *p*th real labeled data of **Y$_s$**.
M$_s$: Pre-trained CNN model using **D$_s$**.
θ: Trainable parameters of CNN model.
CV(θ): **θ** of convolution layers.
FC(θ): **θ** of fully-connected layers.
CV (θ)$_0$: Initial **θ** of **CV(θ)**.
FC(θ)$_0$: Initial **θ** of **FC(θ)**.
D$_b$: *b*th dataset of **X$_b$** and **Y$_b$** in target domain.

$\mathbf{X}_b(q)$: qth process data of $\mathbf{X}_b$.
$y_b(q)$: qth real labeled data in $\mathbf{Y}_b$.
$\mathbf{M}_b^{\mathrm{I}}$: Online model without updating using $\mathbf{D}_b$.
$\mathbf{M}_b^{\mathrm{II}}$: Updated online model using $\mathbf{D}_b$.
$\mathbf{M}_{b-k}^{\mathrm{II}}$: Latest updated model at (b-k) th batch.
$\tilde{\mathbf{X}}_b(q)$: qth modified $\mathbf{X}_b$.

$\bar{\sigma}_s$: Average of stand deviation values of $\mathbf{X}_s$
$\bar{\sigma}_b$: Average of stand deviation values of $\mathbf{X}_b$
$\bar{\mu}_s$: Average of mean values of $\mathbf{X}_s$.
$\bar{\mu}_b$: Average of mean values of $\mathbf{X}_b$.

TLI_b : TLI value of $\tilde{\mathbf{X}}_b$.
DS_b : Distributional similarity value of $\tilde{\mathbf{X}}_b$.
RS_b : RS value of $\tilde{\mathbf{X}}_b$.
LP^u : uth iteration of LP.

$\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$: **CV**($\boldsymbol{\theta}$) of $\mathbf{M}^{\mathrm{I}}{}_b$.

$\mathbf{FC}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$: : **FC**($\boldsymbol{\theta}$) of $\mathbf{M}^{\mathrm{I}}{}_b$.

$\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{II}}{}_b$: : **CV**($\boldsymbol{\theta}$) of $\mathbf{M}^{\mathrm{II}}{}_b$.

$\mathbf{FC}(\boldsymbol{\theta})^{\mathrm{II}}{}_b$: **FC**($\boldsymbol{\theta}$) of $\mathbf{M}^{\mathrm{II}}{}_b$.

$\hat{\mathbf{Y}}_b^{\mathbf{I}}$: Predictive labels via $\mathbf{M}^{\mathbf{I}}{}_b$.
$\hat{\mathbf{Y}}_b^{\mathbf{II}}$: Predictive labels via $\mathbf{M}^{\mathbf{II}}{}_b$.
$\hat{y}_b^{\mathbf{I}}(q)$: qth predictive label in $\hat{\mathbf{Y}}_b^{\mathbf{I}}$.
$\hat{y}_b^{\mathbf{II}}(q)$: qth predictive label in $\hat{\mathbf{Y}}_b^{\mathbf{II}}$.
$\mathbf{CP}_b(q)$: qth Correct-prediction judgement of $\hat{y}_b(q)$.
$\mathrm{Acc}_{\mathbf{s}}$: Accuracy of $\mathbf{M}_s$.
$\mathrm{Acc}_b^{\mathbf{I}}$: Accuracy of $\mathbf{M}^{\mathbf{I}}{}_b$.
$\mathrm{Acc}_b^{\mathbf{II}}$: Accuracy of $\mathbf{M}^{\mathbf{II}}{}_b$.
α : Learning rate of LPu.
$\mathbf{M}_b^{\mathbf{FS}}$: Model trained from scratch at bth batch.
$\mathrm{Acc}_b^{\mathbf{FS}}$: Accuracy of $\mathbf{M}_b^{\mathrm{FS}}$.

Introduction

Rotating components play an essential role in the modern mechanical systems; they are widely applied in various manufacturing industries and mainly composed of different bearings, gearboxes, motors, or spindles [1]. They are vulnerable due to severe working conditions with heavy mechanical loading and long operation time. Failures of these components usually result in unplanned downtime and huge economic loss. Thus, fast and accurate diagnostics of rotating components are always the top priority of prognostics and health management (PHM), to enable a reliable and safe operation for mechanical systems by correctly and timely identifying fault types, root causes, and locations of the failures as well as the time of detection [2-4,5-8]. DL redefines the diagnostic paradigm in manufacturing industries through auto-featuring instead of traditional ML methods, as the latter need to extract hand-crafted and fault-related features manually and empirically based on the expertise knowledge [9]. DL has solved various real-world application issues, such as image classification, natural language processing and failure monitoring of rotating components is of course no exception. These years, DL-based vibration monitoring methods are also proven to be extremely promising to provide robust and comprehensive information for the fault diagnosis of rotating components in many studies [10-14].

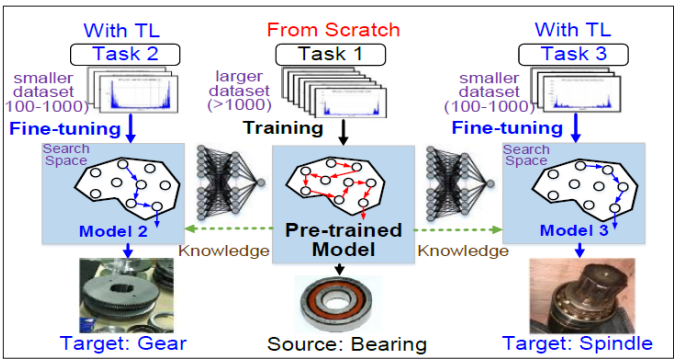

Figure 1: Comparing of ML/DL-based Models Created from Scratch and TL-based Methods

However, data of the rotating systems vary along with complex machinery and changeable working conditions. It is impractical to recollect training data and retrain a new model each time when a new diagnosis task emerges. Compared with other data-rich applications like natural language classification or object recognition in real-world industry applications, collecting a substantial amount of valid samples from each failure type in industries is relatively hard [1]. This deficiency limits the developments of DL models, which are very data-hungry. The model performance can be profoundly improved with a massive amount of training data; and the deeper the model, the more parameters to be trained. Fortunately, the so-called TL paves the way to "stand on the shoulders of giants" for those who tackle with these DL problems.

A. Transfer Learning

TL, cropped in 1976, is not a new concept; however, it is re-emphasized and has been a hot research topic nowadays with a huge success in industrial applications due to the birth of DL [1,2,15,16]. As in Figure 1, there are three rotating-component tasks: the gear, the bearing, and the spindle have to be handled. In the past, each dataset is separately processed to build a data-driven model purely based on their own characteristics. Even if they are three similar rotating-component tasks, the knowledge can only be independently stored inside a specific model without being shared. Rebuilding an entire model from scratch is the only solution, yet it costs a lot of manpower, material resources and time.

With the TL manner, a pre-trained model can be rapidly reused on a specific task in a time-saving way instead of recollecting a new training dataset to learn from scratch. A pre-trained model is usually very deep and well-structured since it learned from a large-scale benchmark labeled dataset beforehand in multi-source domains through various laboratory experiments or related tasks. Then, the knowledge accumulated from previous tasks (such as bearings) can be leveraged and repurposed to adjust a similar target task (such as gears and spindles) using only a small data set via a fine-tuning procedure. This concept is just like that the learning time needed for teaching a violinist to play the viola is much shorter than teaching one beginner who does not understand any string instrument. The violinist has basic musical knowledge and correct posture to hold a violin so that he can spend less time to efficiently acquire how to play the viola.

B. Literature Reviews

Many researchers have made their best efforts to enable TL to achieve fault diagnosis of rotating components in industries by using vibration signals. In our previous works [17-20] we have proposed a model refreshing scheme to deal with some small-scale problems by sharing the ML models to other precision-prediction tasks of similar sizes or operations of machined wheels. In 2017, Zhang et al. were the pioneers who dedicated themselves to develop almost the first work that applied the TL-based concept for fault diagnosis of rolling bearings [21]. Since then, many TL-related works are now springing up like mushrooms. Yang et al. gave a clear and detailed tutorial to guide the industry to integrate the CNN model and TL-based methods to deal with bearing data from different distributions [2]. Wang et al. proposed a modified model based on ResNet-50 to extract and reconstruct low-level features through a multiple scale feature extractor, which gives pseudo labels for intra-class adaption and is validated by using 24 types of TL experiments in total from two different bearing datasets [5]. Cao et al. utilized a pre-trained CNN model to automatically extract the features from gear experimental data without data-preprocessing so as to accurately classify features even with a small set of target data, and a comprehensive comparison of TL structures and learning processes was given as well [22]. Chen et al. proposed a transferable CNN for fault diagnosis of mechanical systems and provided a detailed discussion on the relationship among the accuracy, training samples, and fine-tuned layers [23]. From the aforementioned works, it is a known fact that the rolling bearing, as the connection between the rotating component and support, is considered as the most critical component to identify failure status for the rotating mechanical device at the incipient stage. Thus, the huge amount of rolling bearing labeled data is widely used to serve as the source domain dataset.

C. Problem Statements

Although various contributions had been made by many researchers as introduced in Section I-B, these good results can only be obtained from offline analysis. They are too idealistic to be implemented in a practical online environment. A systematic scenario for the online diagnosis of the bearing is required to enhance the practicality of conventional TL-based methods. To achieve the goal, two poor performances of: 1) updatability and 2) monitorability, should be taken into careful consideration in the environment of mass customization, where rapid and flexible changes of manufacturing processes are indispensable.

1) Updatability

Although using different distributions to fine-tune a pre-trained model is a common solution to reduce the data changeableness and enhance the model generalization, once the distribution distance between source and target domains is too large, the fine-tuning process might not be easily converged. Let SN_{90} be the minimum limit of the sample number that meets the criteria of reaching the diagnosis accuracy 90% after the fine-tuning process, then SN_{90}>1,000 seems to be a basic requirement for existing approaches [21-23]. In a practical real-time scenario, each valid sample has to be gradually acquired and accumulated sample-by-sample during the production. Thus, it definitely spends an extremely long updating time for a fine-tuning process.

2) Monitorability

The fine-tuning process also completely loses its monitoring capability for the target domain. As aforementioned, it needs at least thousands of samples to complete the task of updating a pre-trained model from the source into the target domain. In other words, existing approaches are not feasible under the condition of SN_{90}<1,000. Although fewer samples of existing approaches are required than traditional data-driven methods, obtaining an enough size of fine-tuning samples in a short time is still not realistic. Let CT being the collection time for one specified workpiece-sample, then the total preparing time PT for a TL method can be defined as

$$PT = CT \cdot SN_{90}, \qquad (1)$$

where CT includes the data-collection time of each sample. Data collection has to gather vibration data and corresponding labels, which usually come with a longer metrology delay. Given that CT=15 (mins for each sample) and SN_{90}=1,000 for an illustrative machine, then needed PT will be near to 15,000 mins (almost 11 days).

Thus, this study proposes a TLGIS to provide a systematic online fault-diagnosis procedure by 1) rapidly updating the pre-trained model to adapt the target domain, 2) continuously monitoring the status of rotating components without interruption during the production.

The remainder of this paper is organized as follows. Section II introduces the TLGIS-based system and describes the detailed functionality of each step. Section III illustrates two practical examples. Finally, conclusions are stated in Section IV.

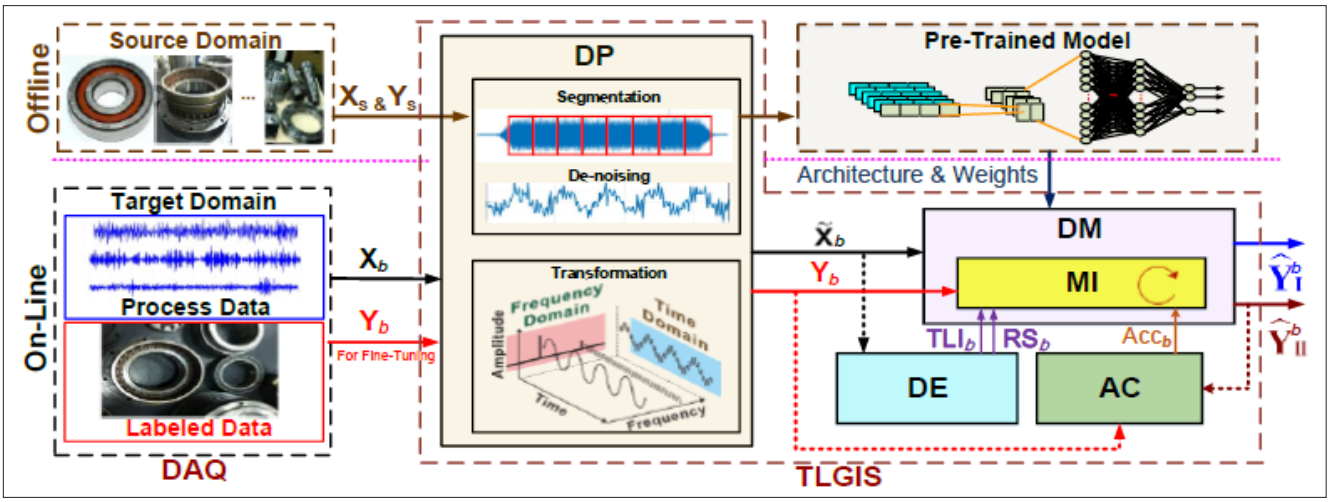

Figure 2: Implementation Structure of TLGIS-based System

TLGIS-Based System

As shown in Figure 2, Data Acquisition (DAQ) and Pre-Trained Model are two basic prerequisites for TLGIS. DAQ has to be performed either in the offline or online environment; while the Pre-Trained Model is built in the offline environment. Details are described in the following.

A. Data Acquisition (DAQ)

DAQ is a vitally important procedure to prepare and gather all the needed samples with good quality for enabling TLGIS. Each sample contains process data and the corresponding labeled data [14]. Process data consist of sensing analog signals and manufacturing parameters that reflect operation stability; while the corresponding labeled data possess the measurement contents of the equipment status. Analog signals of the process data are captured by various types of sensors connected to an analog-to-digital converter; while manufacturing parameters can be directly retrieved from connecting RJ45 (Ethernet) port of the equipment side.

For any time-domain sensing signals with t-second $f(t)$ are acquired by a discrete one-dimensional (1-D) vector X as represented below

$$\mathbf{X} = \mathrm{DAQ}\big(f(t)\big) = \{x[1], x[2], \ldots, x[L]\}, \qquad (2)$$

where L denotes the sampling data point. Note that, for each acquired sample X, 1-D signal is the most commonly used DAQ format in the machinery industry.

B. Pre-trained Model

This step is extremely important since a pre-trained model $\mathbf{M}_s$ has to be ready in advance for activating the procedure of TLGIS. A well-structured network of $\mathbf{M}_s$ helps to speed up both the fine-tuning procedure and the obtaining of a more accurate model in the online environment. Let a large amount of labeled pre-trained dataset with P samples from source domain $\mathbf{D}_s$ in the offline environment be

$$\mathbf{D_S} = \{\mathbf{X_S}, \mathbf{Y_S}\}, \qquad (3)$$

where $\mathbf{X}_s$ is the process dataset and $\mathbf{X_S} = \{\mathbf{X_S}(p) | \mathbf{X_S}(p) \in \mathbf{X}\}$; $\mathbf{Y_S}$ is the real labeled data (failure modes) and $\mathbf{Y_S} = \{y_S(p) | y_S(p) \in N\}$; p= 1,2,.., P. As upper side of Figure 2, in order to build $\mathbf{M}_s$, any $\mathbf{X}_s$ and $\mathbf{Y}_s$ of rotating component from a specific source domain, are also have to be segmented and cleaned by DP. A large number of the pre-trained models are built based on the architecture of a deep CNN due to its ability to automatically extract features from shallow to deep layers. Basic features at shallow layer capture low-level representations such as peak values and skewed shapes of the input signals; while specific features at deeper layers include high-level representations that usually correlating with the failure modes of the input signal [10,25]. A basic CNN architecture mainly comprises of a convolutional layer, a pooling layer and a fully-connected layer. Diagnostic features are extracted from $\mathbf{X}_s$ by a stack of convolutional layers and pooling layers; and then the fault diagnosis finally is made by the fully-connected layer.

Various kernel maps inside convolution layers and fullyconnected layers include the large-scale trainable parameters ($\boldsymbol{\theta}$) that can be continuously updated during any learning process (LP), e.g. training a $\mathbf{M}_s$ or fine-tuning a $\mathbf{M}_s$. Thus, $\boldsymbol{\theta}$ represent the gained knowledge on a specific task, e.g. bearing, which is extremely valuable for providing similar tasks with shared-weight architectures and degrees of translation-invariant characteristics of $\mathbf{M}_s$. Each $\boldsymbol{\theta}$ contains two types of parameters: weights and biases. Weights decide the strength of the connection between two neurons; while biases are additional offsets of the next layer.

A rigorous relationship between LP and $\mathbf{M}_s$ using $\mathbf{D}_s$ can be mathematically written as

$$\mathbf{M_S}\{\mathbf{CV}(\boldsymbol{\theta}), \mathbf{FC}(\boldsymbol{\theta})\} = \mathrm{LP}^u(\mathbf{X_S}, \mathbf{Y_S}, \mathbf{CV}(\boldsymbol{\theta})_0, \mathbf{FC}(\boldsymbol{\theta})_0), \qquad (4)$$

where $\mathbf{M}_s$ is composed of $\mathbf{CV}(\boldsymbol{\theta})$ and $\mathbf{FC}(\boldsymbol{\theta})$, which are respectively updated $\boldsymbol{\theta}$ of convolution layers and fullyconnected layers in $\mathbf{M}_s$ after a U -time iteration LP ; while $\mathbf{CV}(\boldsymbol{\theta})_0$ and $\mathbf{FC}(\boldsymbol{\theta})_0$ are the initial $\boldsymbol{\theta}$ used to train $\mathbf{M}_s$, and u=1,2, …, U. To extract the clear knowledge and explanation from $\mathbf{M}_s$, building a $\mathbf{M}_s$ can totally concentrate on the adequate selection of the model algorithms, the stochastic nature of the process, and the desired model performance. Most important of all, the optimal combination of hyper-parameters for controlling LP, which usually takes a lot of time to collect sufficient $\mathbf{D}_s$, can also be obtained.

Compared to a model executed in the online production environment, $\mathbf{M}_s$ is free from the restriction of the above issues. As emphasized in Section I-B, among various types of rotating components on an equipment, the bearing vibration data are the best source dataset to build $\mathbf{M}_s$ for those tasks lacking data. The datasets include a huge amount of bearing labeled data from normal, wear, to faulty. Most of all, they are publicly available and easily accessible database from the website.

C. TLGIS Modules

TLGIS contains five key modules of DP, DE, DM, MI, and AC. Given the bth small batch dataset with Q_b samples acquired from the target domain ($\mathbf{D}_b$) as

$$\mathbf{D}_b = \{\mathbf{X}_b, \mathbf{Y}_b\}, \qquad (5)$$

where $\mathbf{X}_b$ is the process data and $\mathbf{X}_b = \{\mathbf{X}_b(q) \mid \mathbf{X}_b(q) \in \mathbf{X}\}$; $\mathbf{Y}_b$ is the real measurement data like bearing label data and $\mathbf{Y}_b = \{y_b(q) | y_b(q) \in N\}$; q= 1,2,.., Q_b. The main goal of DE is to correctly estimate $\mathbf{D}_b$ and then properly modify them into a clever format for TLGIS. As stated in Section I-C.(2), Q_b dynamically ranges from 10 to 100 depending on the online production environment, and might be relatively small compared to P. Thus, accumulating the sample number is not realistic and TLGIS must have to be equipped with the ability to overcome this limitation.

Data Preprocessing (DP) Module

DP has to correctly interpret $\mathbf{D}_b$ into a more understandable format to ensure that the relationship between $\mathbf{X}_b$ and $\mathbf{Y}_b$ is reliable enough for later modules.

Process Data ($\mathbf{X}_b$)

To emphasize major components of acquired signals, DP preprocesses each $\mathbf{X}_b$ via three steps of Segmentation, Cleaning, and Transformation [24]. The Segmentation step divides the original signal into several essential parts and deletes irrelevant parts from the original signal. The Cleaning step deals with noised signals by Wavelet Transform (WT) to increases the signal-to-noise ratio. Fast Fourier transform (FFT), which is proven that being very useful in dealing with analog signals, especially the vibration [10]. Other techniques like Principal Component Analysis, or statistical methods also can be applied in this step.

To accelerate fine-tuning procedure and solve the updatability problem addressed in Section I-C.(1), Transformation of DP firstly normalizes $\mathbf{X}_b$, then uses (6) to modify each sample of $\mathbf{X}_b$ into $\tilde{\mathbf{X}}_b$bn to avoid the distribution of $\mathbf{X}_b$ from being too far from $\mathbf{X}_s$. Given qth sample in $\mathbf{X}_b$, then its modification form would be:

$$\tilde{x}[l] = (x[l] \cdot \frac{\bar{\sigma}_b}{\bar{\sigma}_s} + (\bar{\mu}_s - \bar{\mu}_b)), \quad (6)$$

where l=1, 2, …, L;$\bar{\sigma}_b$ and $\bar{\sigma}_s$are the averages of standard deviations (std) of each sample in $\mathbf{X}_b$ ($\bar{\sigma}_b$ = mean($\sigma_b(q)$), e.g $\sigma_b(1)$=std($\mathbf{X}_b(1)$) and $\mathbf{X}_s$ ($\bar{\sigma}_s$ = mean($\sigma_s(p)$), e.g. $\sigma_s(1)$=std($\mathbf{X}_s(1)$)), respectively; $\bar{\mu}_b$ and $\bar{\mu}_s$ are the averages of mean values of each sample in $\mathbf{X}_b$ $\bar{\mu}_b$ = mean($\mu_b(q)$), e.g. $\mu_b(1)$=mean($\mathbf{X}_b(1)$)) and $\mathbf{X}_s$ ($\bar{\mu}_s$ = mean($\mu_s(p)$) e.g. $\mu_s(1)$=mean($\mathbf{X}_s(1)$)), respectively; q=1, 2, …, Q_b; p=1, 2, …, P. The step of Transformation not only ensures the original characteristics and relative properties between $\mathbf{X}_b$ and $\mathbf{X}_s$, but also eliminates unwarranted bias existed in $\mathbf{X}_b$ [18].

Labeled Data ($\mathbf{Y}^b$)

All failure labels must have to be repurposed from original task $\mathbf{M}_s$ to adapt the new task of online environment. Usually, the more label of the new task is, the more numbers of fine-tuning samples needs. Thus, the period of the very beginning is the most severe condition that usually worsens the accuracy of any data-driven model.

To solve the monitorability problem addressed in Section I-C.(2), DE has to obtain corresponded measurement dataset $\mathbf{Y}_b$ for each $\tilde{\mathbf{X}}_b$ so as to rapidly complete the fine-tuning procedure of TLGIS as soon as possible.

As b increases, the amount and diversity of $\mathbf{Y}_b$ are gradually accumulated. DM has a higher chance to be able to learn more concrete knowledge of each label. In brief, DP firstly perform Segmentation and Cleaning; secondly, DP transforms $\mathbf{X}_b$ into $\tilde{\mathbf{X}}_b$.

Data Estimating (DE) Module

DE has to estimates each obtained $\tilde{\mathbf{X}}_b$ by two indexes of the so-called transfer-level-index (TLI) and ratio of samples (RS). The first is to summarize a score of $\tilde{\mathbf{X}}_b$ and $\mathbf{X}_s$ by calculating TLI for $\tilde{\mathbf{X}}_b$ as

$$\mathrm{TLI}_b = 1 - ({}^{\mathrm{DS}_b}/_{\mathrm{DS}_1}), \quad (7)$$

where DS_b stands for the distributional similarity value between two average trends of $\tilde{\mathbf{X}}_b$ and $\mathbf{X}_s$. TLI_b ranges from 0 to 1. Note that, DS_1 is used as a baseline for estimating each computed DS_b, and TLI_1 is accordingly adjusted to 0. Let a distance function DIST being given as

$$\mathrm{DS}_b = \mathrm{DIST}(\bar{\mathbf{X}}_b, \bar{\mathbf{X}}_s), \quad (8)$$

where $\bar{\mathbf{X}}_b$ and $\bar{\mathbf{X}}_s$ are the average trends of $\tilde{\mathbf{X}}_b$ (Q_b samples) and $\mathbf{X}_s$ (P samples), respectively; $\bar{\mathbf{X}}_b = \{\tilde{\mathcal{X}}[1]$),mean $\tilde{\mathcal{X}}[2])$,…,mean $(\tilde{\mathcal{X}}[L])\}$ and $\bar{\mathbf{X}}_s = \mathbf{X}_b = \{\mathcal{X}[1])$,mean $\mathcal{X}[2])$,…,mean $(\mathcal{X}[L])\}$. There are various types of existing distance functions can be directly applied in (10), such as Euclidean Distance. Note that, after each fine-tuning using $\mathbf{D}_b$, $\mathbf{X}_b$ will become a part of $\mathbf{X}_s$. In this manner, as b increasing, more and more $\mathbf{D}_b$ are inputted into the prepared $\mathbf{M}_s$. It means $\mathbf{M}_s$ created by $\mathbf{X}_s$ will gradually have characteristics of $\tilde{\mathbf{X}}_b$and result in a small distance between $\tilde{\mathbf{X}}_b$ and $\mathbf{X}_s$. In this study, the dynamic time warping (DTW) distance function is utilized for

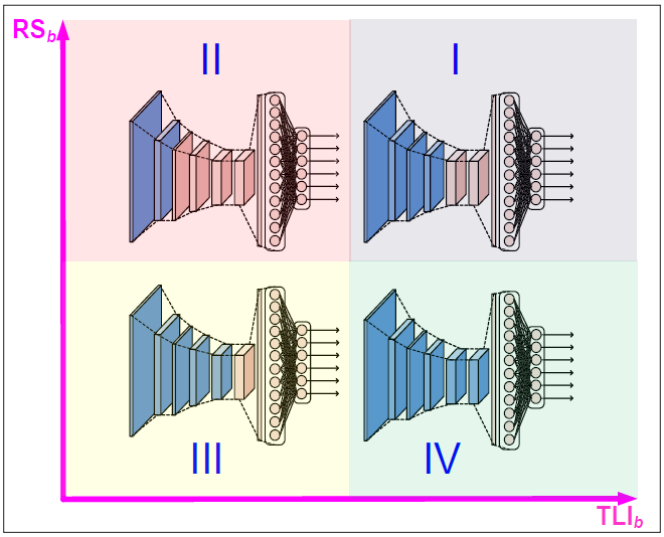

Figure 3: Four Fundamental Fine-tuning Strategies Defined by

measuring the difference between $\tilde{\mathbf{X}}_b$ and $\mathbf{X}_s$. To a certain extent, a large TLI_b value means collected $\mathbf{D}_b$ has a high similarity or a large amount of data size compared with $\mathbf{D}_s$. Thus, the larger the TLI_b value the better [26].

The other index is to compute the RS_b value, which is the ratio of the accumulated sample number and P. In other words, RS_b compares the accumulated amount of $\mathbf{D}_b$ with the amount of $\mathbf{D}_s$. RS_b is defined as

$$\mathrm{RS}_b = \frac{\mathrm{AQ} + Q_b}{P}, \quad (9)$$

where AQ is the accumulating number that have ever used for fine-tuning $\mathbf{M}_s$ from the batch number 1 to b-1. Generally, the sample number of $\mathbf{D}_s$ is tremendously larger than $\mathbf{D}_b$, which are hard to have a significant effect on $\mathbf{M}_s$ during a fine-tuning procedure especially in a beginning period. RS_b provides the information that whether the accumulated samples have an influence on $\mathbf{D}_s$ or not.

In brief, DE estimates the relationship between $\mathbf{X}_s$ and $\tilde{\mathbf{X}}_b$ by TLI_b and RS_b for following modules so that dynamically and gradually achieving the most important purpose in TLGIS, that is, the fine-tuning task under an online production environment with limited samples using $\tilde{\mathbf{X}}_b$ and $\mathbf{Y}_b$.

Data Modeling (DM) Module

Every time $\tilde{\mathbf{X}}_b$ obtained from DE, DM always prepares an online model $\mathbf{M}^{\mathrm{I}}_b$ for the real-time diagnostic purpose. As defined in (10), the initial $\mathbf{M}^{\mathrm{I}}_b$ inherit weights $\mathbf{CV}(\boldsymbol{\theta})\mathrm{I}_b$ and $\mathbf{FC}(\boldsymbol{\theta})\mathrm{I}_b$ from $\mathbf{M}_s$ by directly importing a well-structured architecture of $\mathbf{M}_s$ when b=1; otherwise, the latest updated model $\mathbf{M}^{\mathrm{II}}_{b\text{-}k}$ with $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{II}}_{b\text{-}k}$ and $\mathbf{FC}(\boldsymbol{\theta})^{\mathrm{II}}_{b\text{-}k}$completed by the previous kth (k<b) batch of $D_{b\text{-}k}$, is also regarded as a $\mathbf{M}_s$ and accordingly prepared for $\mathrm{M}^{\mathrm{I}}_b$ when b>1.

$$\begin{cases} \mathbf{M}^{\mathrm{I}}_b\{\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}_b, \mathbf{FC}(\boldsymbol{\theta})^{\mathrm{I}}_b\} = \\ \mathbf{M}_s\{\mathbf{CV}(\boldsymbol{\theta}), \mathbf{FC}(\boldsymbol{\theta})\}, \text{if } b = 1 \\ \mathbf{M}^{\mathrm{II}}_{b-k}\{\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{II}}_{b-k}, \mathbf{FC}(\boldsymbol{\theta})^{\mathrm{II}}_{b-k}\}, \text{otherwise} \end{cases}, \quad (10)$$

where k<b. Before obtaining the corresponding $\mathbf{Y}_b$, DM emphasizes model promptness by inputting $\tilde{\mathbf{X}}_b$ into $\mathrm{M}^{\mathrm{I}}_b$ to compute real-time predictive results $\hat{\mathbf{Y}}^{\mathrm{I}}_b$ as

$$\hat{\mathbf{Y}}^{\mathrm{I}}_b = \mathbf{M}^{\mathrm{I}}_b(\tilde{\mathbf{X}}_b). \quad (11)$$

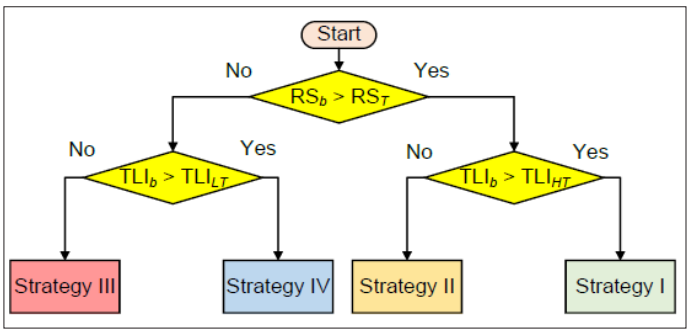

Figure 4: Strategic Decision-making Procedure of MI for Determining Specific Fine-tuning Manner of New Task

Model Improving (MI) Module

Once the fine-tuning is activated, MI has to improve the accuracy of $\mathbf{M}^{\mathrm{I}}{}_b$ using $\tilde{\mathbf{X}}_b$ and $\mathbf{Y}_b$. MI needs to design a specific and proper TL architecture by customizing $\mathbf{M}^{\mathrm{I}}{}_b$ as a new feature extractor to fit the requirement of $\mathbf{D}_b$ based on the RS_b and TLI_b values. MI can be seen as the realization of concretizing the concept of TL into actions. $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ might be fixed (frozen) or partially/entirely retrained (fine-tuned) based on TLI_b results. When b=1, the classifier in $\mathbf{M}^{\mathrm{I}}{}_b$ ($\mathbf{M}^{\mathrm{I}}{}_1$) can be replaced with other types of classifiers to retrain or update entire $\mathbf{FC}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ depending on the target task.

When $\mathbf{Y}_b$ is obtained (b>1), MI fine-tunes the corresponding $\mathbf{M}^{\mathrm{I}}{}_b$ according to the pairs of $\mathbf{D}_b$ and the selected LP algorithm. $\mathbf{M}^{\mathrm{I}}{}_b$ will be updated to become a freshest model $\mathbf{M}^{\mathrm{II}}{}_b$, which can be defined as

$$\mathbf{M}_b^{\mathrm{II}}\{\mathbf{CV}(\boldsymbol{\theta})_b^{\mathrm{II}}, \mathbf{FC}(\boldsymbol{\theta})_b^{\mathrm{II}}\} = \mathrm{LP}^u\left(\tilde{\mathbf{X}}_b, \mathbf{Y}_b, \mathbf{CV}(\boldsymbol{\theta})_b^{\mathrm{I}}, \mathbf{FC}(\boldsymbol{\theta})_b^{\mathrm{I}}\right), \quad (12)$$

where $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ and $\mathbf{FC}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ inherit from latest updated $\boldsymbol{\theta}$ of previous $\mathbf{M}^{\mathrm{II}}{}_{b-k}$; while $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{II}}{}_b$ and $\mathbf{FC}(\boldsymbol{\theta})^{\mathrm{II}}{}_b$ are the updated $\boldsymbol{\theta}$ using the present $\tilde{\mathbf{X}}_b$ and $\mathbf{Y}_b$ of $\mathbf{D}_b$. Note that, during the iteration of LP , (12) processes $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ according to the frozen part and the unfrozen part. Let the total number of convolutional layers of $\mathbf{M}^{\mathrm{I}}{}_b$ be nl and the number of frozen layers be fl, then values of $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ from the first layer to fl remain unchanged during iteration of LP, which means $\mathbf{CV}(\boldsymbol{\theta}|1{:}\,\mathrm{fl})^{\mathrm{I}}{}_b$ will be directly assigned to $\mathbf{CV}(\boldsymbol{\theta}|1{:}\,\mathrm{fl})^{\mathrm{II}}{}_b$; while $\mathbf{CV}(\boldsymbol{\theta}|\mathrm{fl}+1{:}\,\mathrm{nl})^{\mathrm{I}}{}_b$ are continuously updated from layer fl+1 to the last layer nl during iteration of LP.

In a practical environment, since the sample size and similarity of obtained $\mathbf{D}_b$ compared with $\mathbf{D}_s$ may vary from batch-by-batch, MI also changes its LP decisions for each $\mathbf{D}_b$. As Figure 3, four strategies are decided by values of TLI_b (horizontal axis) and RS_b (vertical axis), which are measured in DE. Each strategy of MI represents a specific relationship between TLI_b and RS_b ; then its relative amount of fl and nl for $\mathbf{M}^{\mathrm{I}}{}_b$ can be depicted in blue and red, respectively. Details of four strategies are described in below.

Strategy I (S1)

S1 usually happens in a large $\mathbf{D}_b$ similar to $\mathbf{D}_s$. MI is suggested to freeze $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ of 60% lower layers (fl = 0.6nl) of $\mathbf{M}^{\mathrm{I}}{}_b$, then it retrains $\mathbf{FC}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ and fine-tunes the remaining 20% layers to make top layers well-adapted to $\mathbf{D}_b$. This strategy is the most ideal situation since a large dataset of $\mathbf{D}_s$ has an identical distribution as $\mathbf{D}_s$ does. Whatever tasks encountered, it can achieve the best accuracy by using $\mathbf{D}_b$ to fine-tune $\mathbf{CV}(\boldsymbol{\theta})$ I_b in higher layers, which are responsible to extract the high-precision features that increase model accuracy. Over-fitting and under-fitting don't often occur in S1.

Strategy II (S2)

S2 usually happens in a large $\mathbf{D}_b$ dissimilar to $\mathbf{D}_s$. MI is suggested to freeze $\mathbf{CV}(\boldsymbol{\theta})\mathrm{I}_b$ of 20% lower layers of $\mathbf{M}^{\mathrm{I}}{}_b$, then it retrains $\mathbf{FC}(\boldsymbol{\theta})$ I_b to obtain $\mathbf{M}^{\mathrm{II}}{}_b$. Although $\mathbf{D}_b$ are different from $\mathbf{D}_s$, it seems not a big issue for a large dataset $\mathbf{D}_b$ as in S1, since S2 has the chance to directly rebuild the entire $\mathrm{M}^{\mathrm{II}}{}_b$ (fl=nl) from scratch by using the architecture of $\mathbf{M}^{\mathrm{I}}{}_b$.

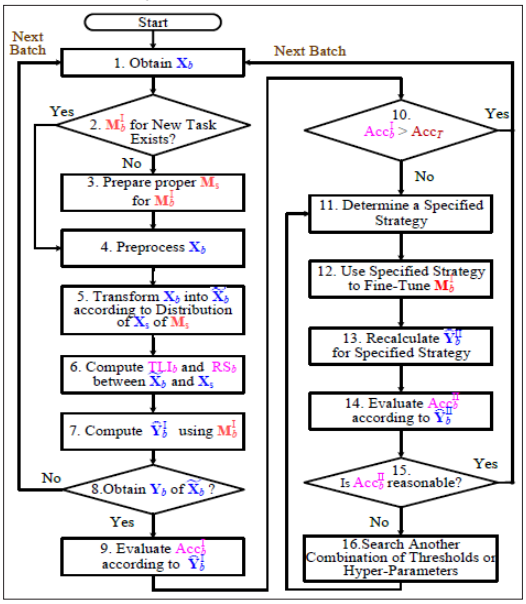

Figure 5: Implementation Flowchart of TLGIS Procedure

Strategy III (S3)

S3 usually happens in a small $\mathbf{D}_b$ dissimilar to $\mathbf{D}_s$. MI is suggested to only fine-tune $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ in the top one layer (the so-called bottleneck layer), freeze nl-1 layer of $\mathbf{M}^{\mathrm{I}}{}_b$ (fl=nl-1), and then retrain $\mathbf{FC}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ by a few $\mathbf{D}_b$. This strategy is extremely rigorous but frequently occurred in current machinery fields. The reason is that $\mathbf{D}_b$ are different from $\mathbf{D}_s$. However, there are too few $\mathbf{D}_b$ to retrain the entire $\mathbf{M}^{\mathrm{I}}{}_b$. A clever way is to temporarily fix $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ in nl-1 layer of $\mathbf{M}^{\mathrm{I}}{}_b$ since the extracted features in lower layers are usually similar and they only repurpose $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ of the top layer into characteristics of $\mathbf{D}_b$ during this transitional period. Nevertheless, for long-term consideration, the only solution to overcome this limitation is to increase the size of $\mathbf{D}_b$ to concretize the fine-tuning procedure.

Strategy IV (S4)

S4 happens in small $\mathbf{D}_b$ similar to $\mathbf{D}_s$. MI is suggested to freeze all (100%) $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ of $\mathbf{M}^{\mathrm{I}}{}_b$ and it only retrains $\mathbf{FC}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ on the classifier. Although the size of $\mathbf{D}_b$ is small, a high similarity exists between $\mathbf{D}_b$ and $\mathbf{D}_s$, which means all $\mathbf{CV}(\boldsymbol{\theta})^{\mathrm{I}}{}_b$ can be fixed since features from low to high levels might be similar to each other.

After the fine-tuning procedure of $\mathbf{M}^{\mathrm{I}}{}_b$, DM accordingly recalculates the outputs as $\hat{\mathbf{Y}}_b^{\mathrm{II}}$ in (13) via $\mathbf{M}^{\mathrm{II}}{}_b$ to represent the self-correction ability, which focuses on the model accuracy.

$$\hat{\mathbf{Y}}_b^{\mathrm{II}} = \mathbf{M}_b^{\mathrm{II}}\left(\tilde{\mathbf{X}}_b\right) \quad (13)$$

In detail, aforementioned four strategies are specifically divided by one threshold of RS (RS_T), one low threshold of TLI (TLI_{LT}), and one high TLI threshold (TLI_{HT}) as categorized in Figure 4. MI firstly checks whether RS_b>RS_T or not; then it depends on the result to compare TLI_b with TLI_{LT} or TLI_{HT}, respectively, to assign a specified strategy for the current pair of $\tilde{\mathbf{X}}_b$ and $\mathbf{Y}_b$. Moreover, the determination of thresholds is gradually and dynamically performed by a grid-search method along with the

TLGIS procedure; details are described in Section II-D.

Accuracy Checking (AC) Module

AC is responsible to evaluate whether the DM performance is acceptable or not. Let the qth result be $\hat{y}_b$ (q), its corresponding actual label $y_b(q)$, and the sample number Q_b; then the fraction of the DM's accuracy is defined as Acc_b in (14). The closer Acc_b is to 1, the better the diagnosis accuracy is achieved.

$$Acc_b = \frac{\sum_{q=1}^{Q_b} \mathbf{CP}_b(q)}{Q_b}, \ \mathbf{CP}_b(q) = \begin{cases} 1, & \text{if } \hat{y}_b(q) = y_b(q) \\ 0, & \text{otherwise} \end{cases}, \quad (14)$$

where $\mathbf{CP}_b$ denotes a vector recording the correct-prediction judgement that indicates whether $\hat{y}_b$ (q) equals to yb (q). Acc^{I}_b and Acc^{II}_b are notations used to represent Acc_b of and $\hat{\mathbf{Y}}^{I}_b$ respectiv $\hat{\mathbf{Y}}^{II}_b$; while $\hat{y}^{I}_b(q)$ and $\hat{y}^{II}_b(q)$ are elements in $\hat{\mathbf{Y}}^{I}_b$ and $\hat{\mathbf{Y}}^{II}_b$ respectively.

Flowchart of the TLGIS Procedure

Given that all signals $f(t)$ TLGIS needs have been processed via DAQ in advance, the complete flowchart of TLGIS is depicted in Figure 5, where it begins when TLGIS receives $\mathbf{X}_b$. TLGIS interactively performs five modules introduced in Section II-C. DP preprocesses input data $\mathbf{X}_b$ then transforms $\mathbf{X}_b$ into $\tilde{\mathbf{X}}_b$; DE estimates the relationship between $\tilde{\mathbf{X}}_b$ and $\mathbf{X}_s$; DM prepares $\mathbf{M}^{I}_b$ and diagnoses failure results; MI designs the updating architecture or executes the LP procedure for $\mathbf{M}^{II}_b$; and AC validates learning results and activates LP of MI.

1. DE obtains $\mathbf{X}_b$ from the target online environment.
2. DM judges if there is an online $\mathbf{M}^{I}_b$; if yes, go to Step 3; otherwise go to Step 4.
3. DM prepares latest $\mathbf{M}^{I}_b$ by copying a proper architecture of $\mathbf{M}_s$ and initializing a newly added FC layer. The Softmax function should also to be repurposed according to $\mathbf{Y}_b$.
4. DP preprocesses Xb.
5. DP transforms $\mathbf{X}_b$ into $\tilde{\mathbf{X}}_b$ according to $\mathbf{X_s}$.
6. DE computes TLI_b and RS_b of $\tilde{\mathbf{X}}_b$ for measuring the relationship between Xb and $\tilde{\mathbf{X}}_b$.
7. DM computes real-time $\hat{\mathbf{Y}}^{II}_b$ by inputting into $\mathbf{M}^{I}_b$.
8. Whether matched label data $\mathbf{Y}_b$ of $\tilde{\mathbf{X}}_b$ can be obtained or not. If yes, b+1 and go back to Step 1; otherwise go to Step 8.
9. AC evaluates $\hat{\mathbf{Y}}^{I}_b$ to calculate the fraction of Acc^{I}_b.
10. AC judges if Acc^{I}_b is higher than a predefined Acc_T (95%); if yes, b+1 and go back to Step 1; otherwise go to Step 11.
11. MI determines a specified strategy for $\{\tilde{\mathbf{X}}_b, \mathbf{Y}_b\}$according to their RS_b and TLI_b.
12. MI fine-tunes $\mathbf{M}^{I}_b$ to obtain $\mathbf{M}^{II}_b$ using the dataset of $\{\tilde{\mathbf{X}}_b, \mathbf{Y}_b\}$ according to the specified strategy.
13. DM re-inputs Xb into fine-tuned $\mathbf{M}^{II}_b$ to obtain Y after updating $\mathbf{M}^{I}_b$.
14. AC evaluates Y to calculate the fraction of Acc^{II}_b.
15. AC judges if Acc^{II}_b is reasonably higher than Acc_T (95%). If yes, b+1 and go back to Step 1; otherwise go to Step 16.
16. MI performs the grid-search procedure to update thresholds and hyper-parameters, then go back to Step 11.

Determination of RS_T, TLI_{LT} and TLI_{HT}

The determination of the three thresholds of RS_T, TLI_{LT}, and TLI_{HT} can significantly affect the performance of TLGIS. For example, let $\mathbf{D}_s$=1,000 and $\mathbf{D}_1$=100, then RS_1 is 0.1; however, RS_1=0.5 when $\mathbf{D}_s$=200. That is, RS_b would be significantly different by the size of $\mathbf{D}_s$ and it means that thresholds cannot be unchanged. The three thresholds interact with each other and can be solved by various multi-objective optimization methods. In this paper, a traditional grid-search method combined with an empirical rule is provided for searching proper thresholds.

When b=1, TLGIS focuses on determining RS_T since TLI is calculated only when b>1 (TLI_1=0). MI initializes default values of RS_T as 0.01 for supposing that any Q_b accounting for at least 1% of the amount of P satisfies the condition of large dataset. Whether RS_1 is higher than RS_T or not, Acc^{II}_b of S2, representing a larger size of $\mathbf{D}_1$, definitely performs better than that of S3. Otherwise, Step 15 of TLGIS modifies RS_T into 0.1×RS_T.

When b>1, TLGIS focuses on searching TLI_{LT} or TLI_{HT}, which are initially set as 0.1 and 0.5, respectively. For $\mathbf{D}_b$ with RS_b<RS_T, any TLI_b being higher than 0.1 is accordingly regarded as S4 and Acc^{II}_b of S4 has to be reasonably better than that of S3. Otherwise, Step 15 of TLGIS modifies TLI_{LT} into TLI_{LT}+0.1. For $\mathbf{D}_b$ with RS_b>RS_T, any TLI_b being higher than 0.5 means that $\mathbf{D}_b$ accounts for 50% of total similarity of $\mathbf{D}_s$ and it qualifies to be an extremely similar dataset with $\mathbf{D}_s$. Under this situation, Acc^{II}_b of S1 is expected to be better than that of S2. Otherwise, Step 15 of TLGIS modifies TLI_{LT} into TLI_{LT}+0.1. Note that, aforementioned initial values and modified values are always relative but not absolute. The grid-search method plays an important role to determine reasonable thresholds among these complex relationships. Related hyper-parameters are also tuned in this manner, e.g. learning rate (α).

Table 1: Descriptions of Femto Dataset

Label	Speed (RPM)	Loading (NT)	Sample No.
0	1,800	4,000	8,850
1	1,650	4,200	8,850
2	1,500	5,000	3,800

Table 2: Descriptions of Cwru Dataset

Label	Speed (RPM)	Loading (HP)	Diameter (Inch)	Sample No.
0	1,797	0	0.007	487
1	1,772	1	0.007	486
2	1,750	2	0.007	486
3	1,730	3	0.007	486
4	1,797	0	0.014	486

III. Illustrative Examples

Two practical examples are used to validate the accuracy and effectiveness of the proposed TLGIS procedure. In this study, $\mathbf{M}_s$ is implemented by a variant of VGGNET: VGG-19 structure, consisting of 16 convolutional layers accompanied with 5 max-pooling layers and ending in 3 FC layers with one Softmax function, to obtain $\boldsymbol{\theta}$ and extract critical features [13]. The rectified linear unit (ReLu) is selected as the activation function for the output values of the convolution layers and fully-connected layers. The reason for choosing to use VGG-19 is that the fine-tuned VGG-19 architecture outperforms other CNNs and hybrid learning methods in image classification tasks [27]. In DP, both $\mathbf{X}_b$ and $\mathbf{X}_s$ are de-noised by WT [24]. Each sample in this Section is firstly adopted for DE. Then, each 1-D spectrum with the length of 1,000 is directly reshaped into a 2-D matrix called "vibration image" with the size of 20×50 [15,28].

Two experiments are performed over a desktop computer system having an Intel Core i9-11900F CPU, 32 GB RAM and one NVIDIA GeForce RTX 3070 Ti GPU. The programs are written using the Keras open source neural network library in Python running on top of the TensorFlow deep learning framework.

Example 1: Transferability of Different Conditions

Example 1 demonstrates the transferability of different bearings under various working conditions by using two open-source bearing vibration datasets provided by Case Western Reserve University (CWRU) and FEMTO-ST Institute (FEMTO). Both datasets have been widely used as the standard reference for detecting and diagnosing rotating bearing faults [13,14].

Pre-trained Model FEMTO Dataset Description

FEMTO is a dataset used for best estimates of the remaining useful life of ball bearings in IEEE PHM Data Challenge Competition 2012. Run-to-failure trends of totally 17 bearings are acquired by two accelerometers (vertical and horizontal directions) under three different working conditions. The useful life of each bearing is going to an end when the amplitude of the vibration signal reaches 20g. As in Table 1, A total of 21,500 samples of vertical-direction vibration data are used for $\mathbf{D}_s$. The LP settings used in this paper are referred to expert experience or experiment results as recommended in [10-14]. $\mathbf{M}_s$ is trained by using ADAM and dropout with hyper-parameters including training batch=128, learning rate $\alpha=10^{-4}$, iteration U=200, filter number=128, stride number= 1×1 pooling size= 3×3, and filter size= 3×3, respectively [28]. During each epoch, the LP procedure randomly selects 80% of the samples in $\mathbf{D}_s$ to train $\mathbf{M}_s$ with the 3 output class labels; while the remaining 20% samples are used to validate the performance during each epoch [29]. The average Accs of 200 epochs is 98.00%, which shows a very good performance in training the three labels of $\mathbf{D}_s$.

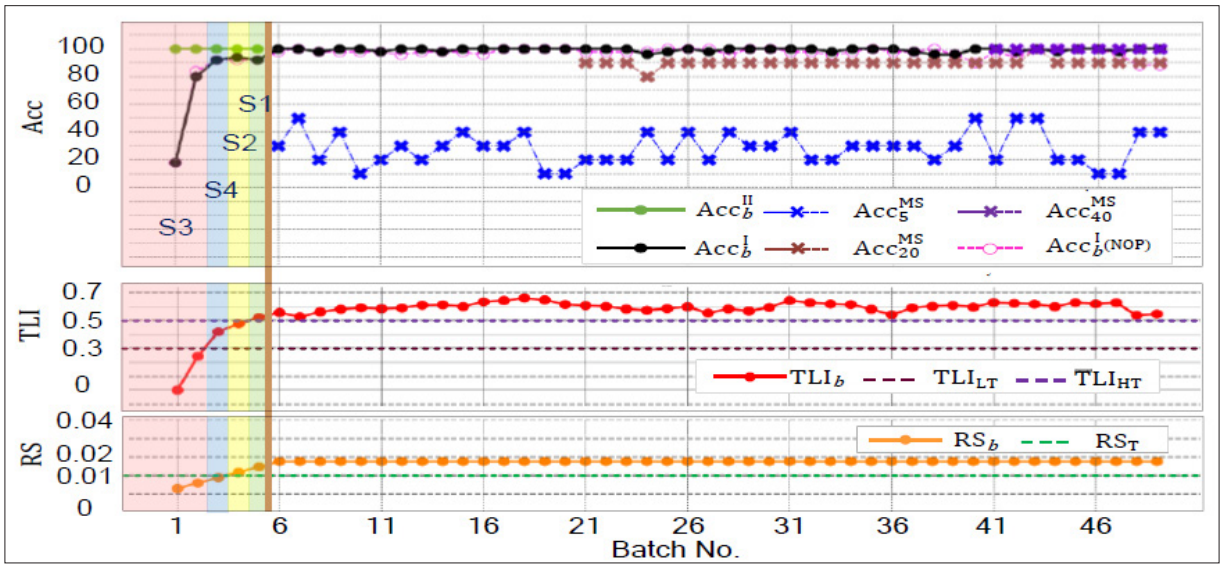

Figure 6: Comparing Results of $\mathbf{M}_b$ Fine-tuned from FEMTO-based $\mathbf{M}_s$ using CWRU via TLGIS, and Results of $\mathbf{M}_b^{FS}$ FS using CWRU

CWRU Dataset Description

The CWRU datasets are very famous for the large amount of rolling bearing vibration with greater variety of fault diameters produced by using electrical-discharge machining (EDM). Many different combinations of datasets can be prepared for machinery applications depending on experiment requirements. In this example, the transferability between different working conditions is the main task to be validated instead of failure modes. Thus, the fault bearingdata of inner race close to the position of the drive-end at the sampling rate of 48k Hz is the only dataset used in Example 1. Totally 2,431 samples are chosen to be $\mathbf{D}_b$ for fine-tuning $\mathbf{M}^I_b$ as described in Table 2.

TLGIS Results of using a FEMTO-based Pre-Trained Model

To simulate a real-world production scenario, samples from the target domain are set to be partially processed in small batches (Q_b=50) rather than completely inputted into $\mathbf{M}^I_b$ since acquiring a large amount of dataset at one time is impractical. $\mathbf{M}_1$ is initialized using the pre-trained weights of $\mathbf{M}_s$ and the output class of Softmax function in the FC layer is repurposed from 3 to 5. Moreover, to validate that the proposed TLGIS procedure is certainly superior to the traditional method, the performance of $\mathbf{M}^I_b$ is compared with the same VGG-19 model directly built from scratch ($\mathbf{M}_b^{FS}$) using $\mathbf{D}_b$ themselves, e.g. $\mathbf{M}^{FS}_{10}$ represents $\mathbf{M}^{FS}_b$ is built using $\mathbf{D}_b$ accumulated from b=1 to 10; while Acc^{FS}_{10} denotes the accuracy of $\mathbf{M}^{FS}_{10}$.

Table 3: Descriptions of WP892 and A1727 Dataset

	Label	Speed (RPM)	Wheel Quality	Sample No.
WP892	0	750	High	350
	1	750	Low	50
A1727	0	2,000	High	350
	1	2,000	Low	50

Taking the CWRU fine-tuning trends with Acc_T=0.95 as an example displayed in the upper side of Figure 6. The first batch (Samples 1-50) $\mathbf{X}_1$ is served as a criterion (TLI_1 = 0) for following TLI_b and according Acc^I_1 (black-horizontaldotted line in Figure 6) is an unacceptable performance of 18% since $\boldsymbol{\theta}$ has not been updated by the pair of $\mathbf{D}_1$. Note that, once $\mathbf{Y}_1$ is obtained, Acc^{II}_1 can be efficiently improved to 100% (green-horizontal-dotted line in Figure 6) by finetuning $\mathbf{D}_1$ using S3. Then, TLI_2 is slightly improved to 0.244; while Acc^I_2 is 80% and Acc^{II}_2 are improved to 100% using S3 as well.

In this manner, TLGIS completes the learning task batchby- batch at b=5, with Acc^I_4 =94% and Acc^I_5 =92%, and both Acc^{II}_4 and Acc^{II}_5 are 100% using S2 and S1, respectively. Note that, Acc^{II}_5 achieves Acc_T, which means that, TLGIS takes 5 batches (totally 250 samples) to complete a five-label finetuning task with TLI_5=0.522 and RS_5=0.015. TLGIS does not need more samples to fine-tune $\mathbf{M}^I_6$ and can correctly monitor the rotating condition of the bearing

with Acc^{I}_{6} =100%. Moreover, $\mathbf{M}^{I}_{6}$ monitors the remaining batches from b=6 to 49 (totally 2,181 samples and $\mathbf{D}_{49}$ only contains 31 samples) with Acc^{I}_{6} =99.3%.

On the other hand, $\mathbf{M}_{b}^{FS}$ is obviously not prepared during the beginning stage since the accumulated number of $\mathbf{D}_{b}$ is not enough to create $\mathbf{M}_{5}^{FS}$. Comparing to that TLGIS needs 250 samples to complete a fine-tuning task, $\mathbf{M}_{5}^{FS}$ can be easily created when $\mathbf{Y}_{1}$ to $\mathbf{Y}_{5}$ are obtained. However, $\mathbf{M}_{5}^{FS}$ is totally not ready with an unacceptable Acc_{5}^{FS} =30% at b=6 for an online monitoring (blue horizontal-dotted line in Figure 6), and Acc_{5}^{FS} =28.4% during b=6-49 because that the amount of five batches is too insufficient to train the huge amount of $\boldsymbol{\theta}$ inside $\mathbf{M}_{b}^{FS}$. This also means that the effect of using TL is significant, and the situation of negative transfer will not occur.

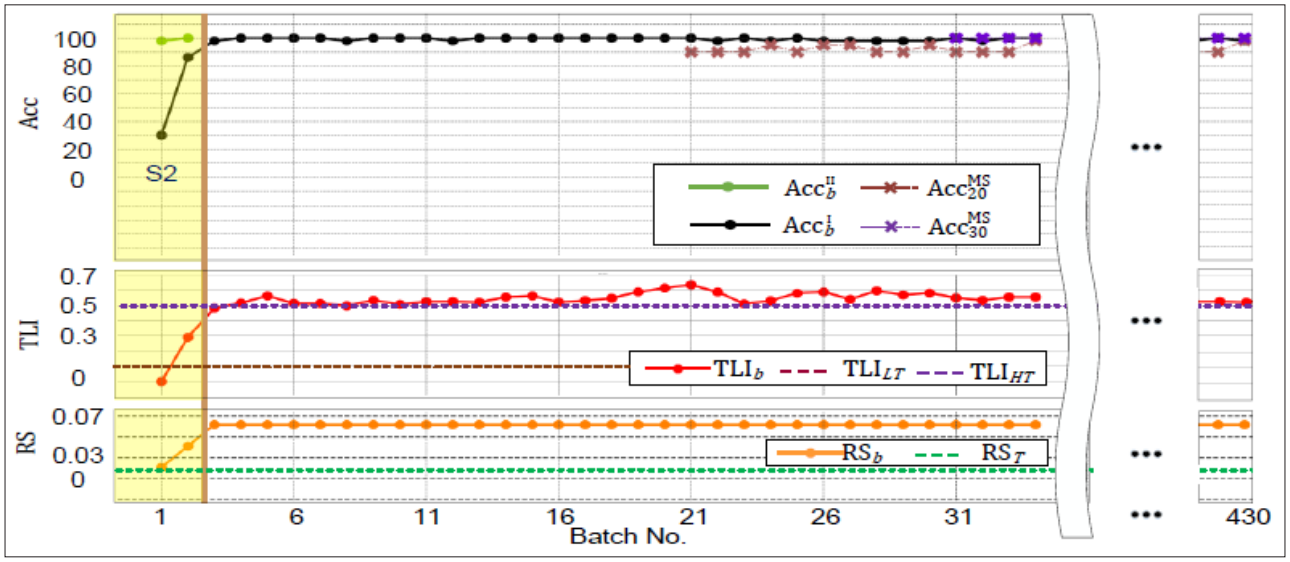

Figure 7: Comparing Results of $\mathbf{M}_{b}$ Fine-tuned from CWRU-based $\mathbf{M}_{s}$ using FEMTO via TLGIS, and Results of $\mathbf{M}_{FS}$ using FEMTO

Another two cases for $\mathbf{M}_{FS}$ provide sufficient samples for the basic training requirement using 20 and 40 batches (1,000 and 2,000 samples), respectively. Observing blood and purple horizontal-dotted line in Figure 6, which represent diagnostic trends of $\mathbf{M}_{20}^{FS}$ and $\mathbf{M}_{40}^{FS}$ with Acc_{20}^{FS} =90.0% (b=21 to 49) and Acc^{FS}_{40}=100% (b=41 to 49), respectively. Although both of them also achieve to a very good performance compare with $\mathbf{M}_{b}^{I}$, one crucial drawback is that they have to firstly accumulate a number of 2,000 samples and required waiting time is extremely long for any industrial applications. In other words, TLGIS results are obviously better than that of $\mathbf{M}_{b}^{FS}$ from any perspective. There is no doubt that $\mathbf{M}_{b}^{FS}$ needs more samples to keep up with the TLGIS accuracy.

TLGIS Results of using a CWRU-based Pre-Trained Model

Different from using the FEMTO-based dataset as a pretrained model, the CWRU dataset is also served as a source domain of $\mathbf{M}_{s}$ and $\mathbf{M}_{b}^{I}$ is gradually fine-tuned by the FEMTO dataset with Q_b=50 as well. Observing Figure7, Acc_{1}^{II} only performs 30% at the beginning stage; however, $\mathbf{M}^{I}_{1}$ is efficiently improved to Acc_{1}^{II} =98% by using S2, which helps $\mathbf{M}^{I}_{2}$ to achieve to Acc^{I}_{2} =86% and Acc^{II}_{2}=100%, respectively. After $\mathbf{M}^{II}_{2}$ qualifying to be used without more fine-tuning samples, TLGIS achieves to Acc^{I}_{3} =98.0% and directly monitor the remaining batches from b =3 to 430 (totally 21,400 samples) with $AccI_{3}$ =99.8%. Obviously, output labels of $\mathbf{M}_{s}$ in the FC layer repurposed from 5 to 3, is relatively an easy task that only needing 2 batches (100 samples) to complete the whole TLGIS procedure. $\mathbf{M}^{FS}_{20}$ and $\mathbf{M}^{FS}_{40}$ are also performed and similar incidents that $\mathbf{M}_{b}^{FS}$ is ready by using at least 1,000 samples with Acc^{FS}_{20} =90.4% (b=21 to 430) and Acc^{FS}_{30} =100% (b=31 to 430), respectively.

Relationships among Acc_{b}^{II}, TLI_{b}, RS_{b} and thresholds

As two bottom parts in Figure 6, relationships among Acc^{II}_{b} and TLI_{b} , RS_{b} using FEMTO $\mathbf{M}_{s}$ and CWRU $\mathbf{D}_{b}$ can be clearly discovered. When b=1, both RS_{1}=0.003 and TLI_{1}=0 being less than their corresponding thresholds are regarded as reasonable for choosing S3. Besides, RS_{T} is determined to be 0.01 according to the available size provided from the first batch. When b=2, S4 is determined to be used according to RS2=0.006<RS_{T} and TLI_{2}=0.244>TLI_{LT}. However, there is still room for improve the determination of TLI_{LT} since Acc2 II of S4<Acc2 II of S3, which is unreasonable. In order to make TLI_{LT} defined reasonably, TLI_{LT} is gradually increased from 0.1 to 0.3 to make TLI2<TLI_{LT} so as to change the strategic decision from S4 to S3. It satisfies that Acc^{II}_{2} of S4 has to be better than that of S3 under the condition of increasing but relatively small TLI_{b} and RS_{b} . During b=1 to 2, S3 uses the largest $\alpha=5\times 10^{-5}$ among four strategies to speed up the weight updates for $\mathbf{M}^{I}_{1}$

.

When b=3, S4 is correctly chosen according to RS_{3}=0.009<RS_{T} and TLI_{3}=0.418>TLI_{LT} since Acc^{II}_{2} of S4 is exactly better than that of S3. Moreover, S4 uses the smallest $\alpha=5\times 10^{-6}$ among the four strategies to update better $\boldsymbol{\theta}$ for $\mathbf{M}^{I}_{3}$ since Acc^{II}_{2} is good enough so that MI retains the characteristics of $\mathbf{M}^{II}_{2}$.

When b=4, S2 is accordingly performed by RS_{4}=0.012>RS_{T} and TLI_{4}=0.478<TLI_{HT} , which represents that it is no need to increase TLI_{HT} . S2 uses $\alpha=2\times 10^{-5}$ to update the $\boldsymbol{\theta}$ of $\mathbf{M}^{I}_{4}$. When b=5, S1 is determined according to RS_{5}=0.015>RS_{T} and TLI5=0.522>TLI_{HT}. There is also no need to remodify since Acc^{II}_{2} of S1 is better than that of S2. S1 uses a smaller $\alpha=3\times 10^{-5}$ than S3 to fine-tune M^{I}_{5}.

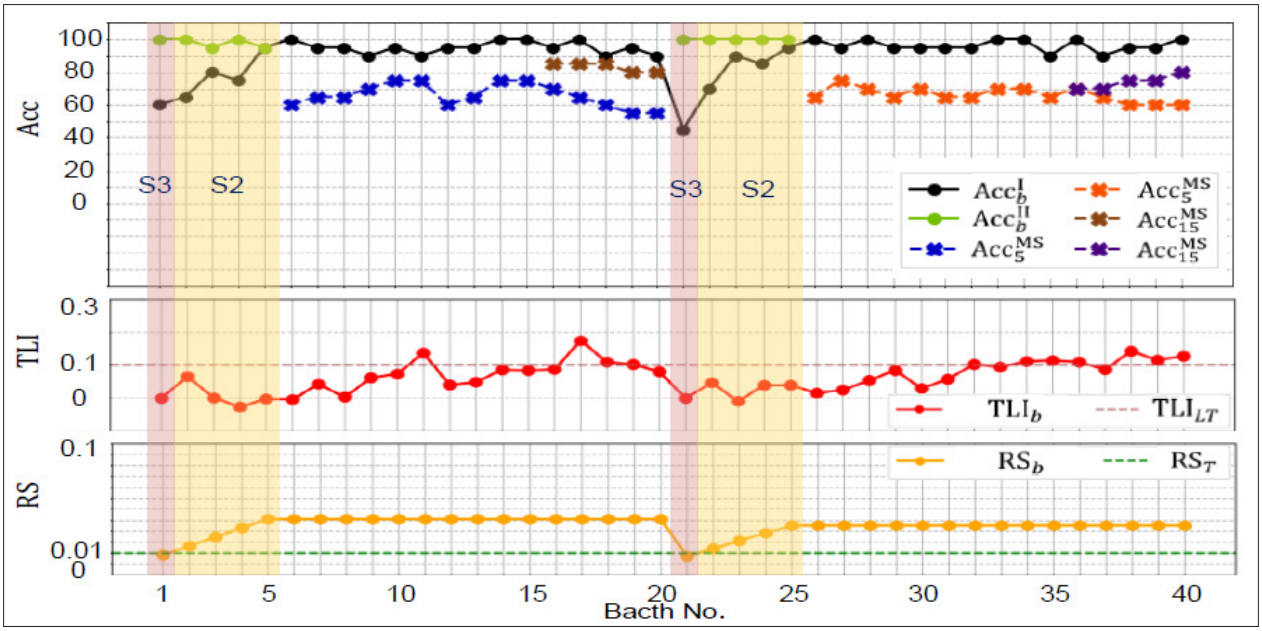

Figure 8: Comparing Results of $\mathbf{M}_b$ Fine-tuned from CWRU-based $\mathbf{M}_s$ using WP892 and A1727 via TLGIS

In this example, four proper thresholds are separately determined by using $\mathbf{D}_1$ to $\mathbf{D}_5$ so that $\mathbf{D}_b$ can be directly assigned to one of four fine-tuning strategies with 5 batches. The amount of total used samples happens in RS_5=0.018, which just exceeds RS_T and means that TLGIS only needs 5 batches (250 samples) to complete LP of $\mathbf{M}_s$. Although RS_b is still small, TLGIS efficiently completes the fine-tuning at this severe condition in 5 batches of $\mathbf{D}_b$.

Another comparing experiment is executed to demonstrate the difference between thresholds searched by a grid-search method, and the non-optimized parameter (NOP) method. Observing the fuchsia-dotted line with hollow circles in Figure 6, Acc^{I}_{b} of TLGIS by using unchanged initial thresholds: RS_T =0.01, TLI_{LT}=0.1, TLI_{HT}=0.5, is a little bit worse than TLGIS using dynamic thresholds. This result proves the fact that proper thresholds for TLGIS is important. As for SN_{90}, SN_{90} of $\mathbf{M}^{I}_{b}$ and $\mathbf{M}^{FS}_{b}$ in Example 1 (two different pre-trained models) are 100 and 1,000, respectively. TLGIS only needs one-tenth the quantity of $\mathbf{M}^{FS}_{20}$ to be able to achieve the goal, which extremely saves costs on PT.

Example 2: Transferability of Different Components

Example 2 demonstrates the TLGIS's transferring ability among three totally different components: one type of rotating bearing and two types of rotating vehicle wheels. The CWRU dataset introduced in Example 1 is used as a training dataset to build the $\mathbf{M}_s$. As for $\mathbf{D}_b$, vibration data are collected during the dry-run process from two different types of rotating vehicle wheels (WP892 and A1727) on an actual lathe, respectively. In other words, there are two fine-tuning tasks will be executed for the two different targets. The first task happens to $\mathbf{M}_s$ (CWRU) that is fine-tuned via $\mathbf{D}_b$ of target WP892; while the second task happens to a mixed Mb (CWRU+WP892) that is continued to be fine-tuned via $\mathbf{D}_b$of target A1727.

Pre-trained Model

As in Table 1, $\mathbf{D}_s$ is composed of a total number of 2,431 samples of vertical-direction vibration data. The architecture of $\mathbf{M}_s$ is the same as Example 1. During each epoch, the LP procedure randomly selects 80% samples in $\mathbf{D}_s$ to train $\mathbf{M}_s$ with the 5 output class labels; while the remaining 20% samples are used to validate the performance during each epoch. The average Accs of 200 epochs is 98.25%, which shows a very good performance in training the three labels of $\mathbf{D}_s$.

Target Data Description

Generally, the vehicle wheel is fixed by a clamper rotated through a belt-driven spindle during a lathe machining [14]. Two accelerometers are installed on the surface of the spindle housing at the place close to the rotating spindle, to collect vibration at the idle state after the lathe wheel finishes the wheel machining. The rotating wheel generates vibration by passing from the wheel side to the accelerometer side through the spindle. The rotating structure of wheels fixed on a clamp can be regarded as a similar characteristic of a rolling-element bearing on the spindle. Once the fixed wheel is unbalanced, the acquired signal may show some relevant characteristics like fault types of inner or outer race for a bearing. Therefore, in order to solve the problem of the lack of wheel rotation data samples, the method of TL is used to help the fault diagnosis of the rotating spindle. In this case, CWRU is a good data source for $\mathbf{D}_s$.

As in Table 3, two levels of machining quality (low/high) corresponding to two labels of the WP892 and A1727 dataset are defined to assign the real measurement status for machined wheels that are examined via an off-machine balancing measurement. Rotating speed are set as 750 Hz and 2,000 Hz for WP892 and A1727, respectively. Note that, the stable signal in the middle section (L=1,000) is selected and segmented from each vibration sample $\mathbf{X}$. Comparing with $\mathbf{D}_s$, each of WP892 and A1727 only has a few valid 400 samples for $\mathbf{D}_b$. This is a frequently happened situation in this highly mass-customization era and TLGIS has to reduce the preparation time on $\mathbf{M}_s$ without training from scratch since CT of a batch usually ranges from 6-8 hours.

Fine-tuning results of TLGIS

Figure 8 shows the experimental results, the left-side portion (Batch No. 1-20) is the first fine-tuning task for WP892. The first batch $\mathbf{X}_1$ is served as a criterion (TLI1=0.066) for following TLI_b and according Acc1I (black-horizontal-dotted line in Figure 8) is an unacceptable performance of 60% since θ has not been

updated by the pair of $\mathbf{D}_1$. Once $\mathbf{Y}_b$ is obtained, the same strategic decision-making procedure as in Example 1 can be used to decide which strategy is used to fine-tune $\mathbf{D}_b$. After 5 batches (Batch No. 1-5 and totally 100 samples) to complete the fine-tuning task (Acc^{I}_{6}=100%), Acc^{I}_{6-20} (totally 300 samples) can be achieved to 94.6% under the condition RS=0.041. The right-side portion (Batch No. 21-40) of the Figure 8 is the second fine-tuning task, which represents that the fine-tuning target is changed from WP892 to A1727. Thus, TLI_{21} is decreased to 0.046 for a low similarity between A1727 and $\mathbf{D}_s$; while RS21 0.007 represents the cumulative sample size of A1727 is relatively few comparing with $\mathbf{D}_s$.

After inputting 5 batches (Batch No. 21-25 and totally 100 samples) into $\mathbf{M}_b$, the second fine-tuning task is completed (Acc^{I}_{26}=100%). Acc^{I}_{26-40} (Batch No. 26-40 and totally 300 samples) can be achieved to 95.7% under the condition RS=0.035. From this result, it can be observed that by the TLGIS method, good performance can be achieved by only a small amount of dataset. On the other hand, Acc^{FS}_{1-5} and Acc^{FS}_{21-25} (blue and orange horizontal-dotted line in Figure 8) respectively are 66% and 67%.

Even the sample number of $\mathbf{M}_b^{FS}$ is increased to 300 (15 batches), which is almost the total sample number of WP892 or A1727, Acc^{FS}_{1-15} and Acc^{FS}_{21-35} (brown and purple horizontal-dotted line in Figure 8) are still the unacceptable 80% and 75%, respectively. The only solution to increase the performance is to collect more labeled samples. Note that, these results prove two facts: 1) training a high-accuracy $\mathbf{M}_b^{FS}$ from scratch needs high-volume $\mathbf{D}_s$ but it is very time-consuming. 2) comparing with the accuracy of $\mathbf{M}^{FS}_b$, TLGIS significantly improves the accuracy almost 30% in two targets, which tells the other truth that there is no negative transfer happening. It is exactly possible for TLGIS to reduce the preparation time for online diagnosis of rotating component data.

Discussions

TLGIS performs better compared to a model trained from scratch $\mathbf{M}^{FS}_b$. In both examples, TLGIS shows its ability to focus on the monitoring of equipment status even when the process of fine-tuning models is unfinished. RS_b and TLI_b also provide rich information that efficiently increasing the transparency of a fine-tuning task. Observing the trend of learning curves, TLGIS has a higher accuracy at the beginning based on a pre-trained model $\mathbf{M}_s$ than that of $\mathbf{M}^{FS}_b$. As *b* increases, a higher slope means that TLGIS has a better learning rate. TLGIS adopts four fine-tuning strategies to dynamically determine the architecture and hyper-parameters of $\mathbf{M}^{I}_b$ using a grid-search manner. In the end of the fine-tuning, a closer asymptote to 100% indicates that TLGIS successfully develops a well-structured fine-tuned model from $\mathbf{M}_s$ to $\mathbf{M}^{II}_b$ after the whole fine-tuning procedure.

Table 4: Tlgis Results on Different Acc_T and Needed Samples

$\mathbf{M}_S$ (Labels)→ $\mathbf{M}_b$ (Labels)	Acc_T (%)	Fine-tuning Sample Number	Testing Sample Number	Acc^{I}_b (%)
FEMTO(3)→ CWRU(5)	95% 92% 90%	250 150 150	2,181 2,281 2,281	99.3 95.4 94.8
CWRU(5)→ FEMTO(3)	95% 92% 90%	100 100 100	21,400 21,400 21,400	99.8 99.8 99.8
CWRU(5)→ WP892(2)	95% 92% 90%	100 100 80	300 300 320	94.6 91.5 90.0
CWRU(5)→ WP892(2) → A1727(2)	95% 92% 90%	160 120 100	240 280 300	95.7 92.5 90.0

Considering an online operation, there is always a trade-off relationship between the accuracy and sample number. Both conditions cannot be satisfied simultaneously. Theoretically, the larger dataset TLGIS has, a higher accuracy can be obtained; however, the major contribution of this paper is to solve the poor performance of two practicality problems. As Table 4, different Acc_T results in different amount of fine-tuning samples. The higher Acc_T means that more fine-tuning samples are needed and it usually has higher Acc^{I}_b.

In the case FEMTO(3)→CWRU(5) of Example 1, using at most 250 samples in 5 batches achieves Acc^{I}_b=99.3% under Acc_T=95%; while a little bit worse Acc^{I}_b=94.8% can be obtained by 150 samples in 3 batches saving time for collecting 100 samples under Acc_T=90%. In the case FEMTO(3)→CWRU(5) of Example 1, only 100 samples are needed among three different Acc_T. Observing results of CWRU(5)→WP892(2)→A1727(2) of Example 2, setting Acc_T as 90% makes $\mathbf{M}^{II}_b$ rapidly usable by saving time for preparing 60 samples to complete the TLGIS procedure compared to using Acc_T=95%; however, Acc^{I}_b=90% can be worse than that of almost 6% in this condition.

From the viewpoint of applied values, preparing a workable and serviceable model is the first priority during the earlier fine-tuning period. An optimal model with best-fit parameters is only available when a large amount of samples is obtained. Instead, TLGIS provides earlier monitoring with acceptable accuracy than $\mathbf{M}^{FS}_b$ since it massively decreases the ready time to adapt to production environments by using fewer samples. In summary, two basic necessary steps make TLGIS possible to complete the ongoing improvement for a specified target task. The first is to select a proper pre-trained model for the target task. The second is to provide TLGIS with needed pairs of samples including process and metrology data during an on-line production scenario. Consequently, the accuracy of TLGIS can be maintained and TLGIS is always ready for an on-line diagnostic task of rotating components.

Conclusions

This paper is motivated by problems that applying TL techniques into an on-line environment is too impractical to execute. Existing TL techniques have achieved significant results but there is a lack of systematic procedure so that TL cannot be widely used in real-world applications due to the two fatal issues: 1) updatability and 2) monitorability of current TL-based methods, which are judiciously addressed in Section I-C. Thus, this paper designs a TLGIS scheme to resolve above problems via the seamless interaction among five modules of Data Preprocessing (DP), Data Estimation (DE), Diagnostic Model (DM), Model Improvement (MI), and Accuracy Check, (AC). The implementation structure and flowchart of a TLGIS-based system are clearly described in Section II. Transfer-Level-Index (TLI) and Ratio-of-Samples (RS) are two rigorously defined indicators used to estimate the dataset relationship between the source and target domains so as to dynamically modify the input dataset and determine a proper fine-tuning strategy. Based on the specific fine-tuning strategy, TLGIS gradually completes the fine-tuning procedure so as to successfully achieve the online diagnosis of rotating components. Two illustrative examples in Section III prove that TLGIS is feasible and promising for meeting the requirements of efficiently and flexibly adapting the complicated and uncertain production environment in real-world applications under mass customization. Section III also discusses the level of applicability by setting different thresholds in an accuracy/sample trade-off problem. In addition, basic prerequisite steps are provided for practitioners to successfully apply TLGIS in their practical problems.

Acknowledgment

This work was financially supported by the "Intelligent Manufacturing Research Center" (iMRC) from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan. This work was also supported by the Ministry of Science and Technology of Taiwan, R.O.C., under Contracts MOST 110-2221-E-006-167. In addition, the authors would like to thank FEMCO, Ltd. in Taiwan, R.O.C. for providing the raw wheel data used in the illustrative examples.

References

1. Shao H, Jiang H, Zhao H, Wang F (2017) a novel deep autoencoder feature learning method for rotating machinery fault diagnosis. Mechanical Systems and Signal Processing 95: 187-204.
2. Yang F, Zhang W, Tao L, Ma J (2020) Transfer learning strategies for deep learning-based PHM algorithms. Applied Sciences 10: 2361.
3. Calabrese F, Regattieri A, Botti L,Mora C, Galizia FG (2020) Unsupervised fault detection and prediction of remaining useful life for online prognostic health management of mechanical systems. Applied Sciences 10: 4120.
4. Liang YC, Wang S, Li WD, Lu X (2019) Data-driven anomaly diagnosis for machining processes. Engineering 5: 646-652.
5. Wang X, Shen C, Xia M, Wang D, Zhu J (2020) Multi-scale deep intra-class transfer learning for bearing fault diagnosis Reliability Engineering & System Safety 202: 107050.
6. Pang G, Shen C, Cao L, Hengel AVD (2021) Deep learning for anomaly detection: A review. ACM Computing Surveys (CSUR) 54: 1-38.
7. Yang HC, Tieng H, Cheng FT (2015) Total Precision Inspection of Machine Tools with Virtual Metrology. Journal of the Chinese Institute of Engineers 39: 221-235.
8. Tieng H, Yang HC, Hung MH, Cheng FT (2013) A Novel Virtual Metrology Scheme for Predicting Machining Precision of Machine Tools. Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA 2013) Karlsruhe, Germany: 264-269.
9. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521: 436-444.
10. Tieng H, Li YY, Tseng KP, Yang HC,Cheng FT (2020) An automated dynamic-balancing-inspection scheme for wheel machining. IEEE Robotics and Automation Letters 5: 2224-2231.
11. He K, Zhang X, Ren S, Sun J (2016) deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA: 770-778.
12. Toh G, Park J (2020) Review of vibration-based structural health monitoring using deep learning. Applied Sciences 5: 1680.
13. Neupane D,Seok J (2020) Bearing fault detection and diagnosis using case western reserve university dataset with deep learning approaches: a review. IEEE Access 8: 93155-93178.
14. Zhang S, Zhang S, Wang B,Habetler TG (2020) Deep learning algorithms for bearing fault diagnostics-a comprehensive review. IEEE Access 8: 29857-29881.
15. Bozinovski S (2020) Reminder of the first paper on transfer learning in neural networks, 1976. Informatica 44: 291-302.
16. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, et al. (2020) a comprehensive survey on transfer learning. Proceedings of the IEEE 109: 43-76.
17. Cheng FT, Tieng H, Yang HC, Hung MH, Lin YC, et al. (2016) Industry 4.1 for wheel machining automation. IEEE Robotics and Automation Letters 1: 332-339.
18. Tieng H, Chen CF, Cheng FT, Yang HC (2017) Automatic virtual metrology and target value adjustment for mass customization. IEEE Robotics and Automation Letters 2: 546-553.
19. Yang HC, Tieng H, Cheng FT (2015) Automatic Virtual Metrology for Wheel Machining Automation. International Journal of Production Research 54: 6367-6377.
20. Tieng H, Tsai TH, Chen CF, Yang HC , Huang JW, et al. (2018) Automatic Virtual Metrology and Deformation Fusion Scheme for Engine-Case Manufacturing. IEEE Robotics and Automation Letters 3: 934-941.
21. Zhang R, Tao H, Wu L, Guan Y (2017) Transfer learning with neural networks for bearing fault diagnosis in changing working conditions. IEEE Access 5: 14347-14357.
22. Cao P, Zhang S, Tang J (2018) Preprocessing-free gear fault diagnosis using small datasets with deep convolutional neural network-based transfer learning. IEEE Access 6: 26241-26253.
23. Chen Z, Gryllias K, Li W (2019) intelligent fault diagnosis for rotary machinery using transferable convolutional neural network. IEEE Transactions on Industrial Informatics 16: 339-349.
24. Yang HC, Tieng H, Cheng FT (2015) Total precision inspection of machine tools with virtual metrology. J Chin Inst Eng 2015: 1446-1447.
25. Bengio Y (2009) Learning deep architectures for AI. Foundations and trends in Machine Learning 2: 1-127.
26. Zhao J, Itti L (2018) ShapeDTW: shape dynamic time warping. Pattern Recognition 74: 171-184.
27. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research 15: 1929-1958.
28. Shaha M, Pawar M (2018) Transfer Learning for Image

Classification. 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA): 656-660.

29. Zhang W, Peng G, Li C (2017) Bearings fault diagnosis based on convolutional neural networks with 2-D representation of vibration signals as input. MATEC Web Conf 95: 13001.