Supervised Contrastive Learning for Multisensor Signals Classification in Automobile Engine Manufacturing

Yoon Sang Cho and Seoung Bum Kim

***Abstract*—Classification of multisensor signals is an important problem in maintaining stable process operations, particularly in advancing predictive modeling for early detection of abnormal states. Self-supervised learning methods, one of the representation learnings, have been widely studied. However, they have focused on using unlabeled data. In this study, we aim to address the challenge of effectively utilizing fully labeled data for modeling multisensor signals. We introduce supervised contrastive learning (SCL) for the classification of multisensor signals. Our training framework involves a two-step process: SCL for encoder pretraining with time-series data augmentations, and classifier training with the pretrained encoder. Our method exhibits superior performance, outperforming traditional supervised learning approaches by a substantial margin. Furthermore, we demonstrate the practical applicability of our approach for early prediction problems through experiments conducted with real-process data obtained from automobile engine manufacturing. Our work offers a promising method for multisensor signal analysis and early fault detection in manufacturing industries.**

***Index Terms*—Automobile engine manufacturing, classification, multisensor signals, supervised contrastive learning (SCL), time-series data augmentation.**

I. Introduction

MULTISENSOR signal data are widely encountered in manufacturing systems [1]. Multisensor signal data are composed of time-series values obtained from multiple sensors attached to process equipment. Multiple sensors monitor aspects such as acceleration and vibration of the process states and machinery, producing a flow of signal data that can represent the features of both normal and abnormal states.

Classification of such multisensor signal data is an important and challenging problem in the manufacturing process. In particular, identifying abnormal signals plays a crucial role in maintaining stable process operations and preventing equipment failures. Traditional approaches, including rule-based systems, statistical methods, and signal-processing techniques, have addressed this problem [2]. However, their implementation necessitates domain expertise and prior knowledge for identifying abnormal patterns in diverse sensor types. Moreover, these approaches encounter challenges in effectively capturing complex patterns inherent in extensive quantities of multisensor signal data [3].

To this end, machine learning models trained on multisensor signals have been applied to accomplish various predictive tasks, including fault detection and diagnosis and machinery health monitoring [4], [5], [6], [7]. Nevertheless, they make an essential demand for manual feature extraction or dimensionality reduction steps [3]. It is difficult to perform the joint optimization of the feature extraction method and the machine learning model, resulting in the inability of the extracted features to directly represent the abnormal status of equipment [8]. Consequently, exploring an effective method for representation learning of multisensor signal data becomes imperative.

Deep learning models incorporate representation learning and classification tasks in unified network architecture. Representation learning involves training models to automatically discover and create meaningful representations or features from raw data. Long short-term memory and convolutional neural networks (CNNs) have become the primary methods with their outstanding predictive performance in time-series modeling [1], [9], [10]. Although the CNNs have been successfully applied to image and text data, they have also shown remarkable performance in time-series data [1]. They have applied for various predictive tasks, including fault detection and diagnosis [11], [12], [13], [14], machinery health monitoring [8], [10], and early detection [15], [16] in many industries.

In recent studies, the importance of representation learning as a pretraining task has emerged in performing supervised classification tasks [17], [18]. The main pretraining strategy is called self-supervised learning. Self-supervised learning

Manuscript received 12 July 2023; revised 6 December 2023; accepted 25 January 2024. This work was supported in part by the National Research Foundation of Korea under Grant RS-2022-00144190, in part by the Institute for Information Communication Technology Planning and Evaluation under Grant IITP-2020-0-01749, in part by the National Research Foundation of Korea under Grant BK21 FOUR, and in part by the National Research Foundation of Korea under Grant 2022R1A6A3A03054796. Paper no. TII-23-2607. *(Corresponding author: Seoung Bum Kim.)*

Yoon Sang Cho was with the Korea University, Seoul 02841, South Korea. He is now with the New York University Medical Center, New York, NY 10016 USA (e-mail: yscho187@korea.ac.kr).

Seoung Bum Kim is with the Korea University, Seoul 02841, South Korea (e-mail: sbkim1@korea.ac.kr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2024.3363189.

Digital Object Identifier 10.1109/TII.2024.3363189

does not require any labeled data but it creates self-defined pseudolabels as supervision; the *positive* samples are created from the data augmentation method, and the *negative* samples are other samples in the minibatch. Then, it performs representation learning by minimizing and maximizing the distance of *positive* and *negative* samples from the *anchor,* which is a target observation. The distance metric is often used as the inner product of the Euclidean distance between representation vectors of the examples [17]. It has shown successful performance in learning powerful representations, not only in the computer vision domain [17], [18] but also in time-series data or sensor signal data [19]. However, they have a risk of considering the samples of the same labels as *negative* samples. Self-supervised contrastive learning considers only one *positive* sample, which is augmented data from the *anchor*, and *negative* samples, comprising all other data within a batch. Thus, within the *negative* samples, samples of the same labels with the *anchor* could be observed in a batch. It indicates that representations of the same label data are trained to push each other apart in the learned feature space.

In self-supervised learning of time-series data, augmentation techniques enable deep-learning models to learn various time-series patterns and increase their classification performance [20]. Many time-series data augmentation techniques are borrowed from image augmentations, such as random horizontal and vertical flipping. Two main methods of time-series augmentation are as follows: 1) the methods of magnitude domain include adding random noise under such names as jittering, scaling, and magnitude warping, and 2) the methods of time-domain include slicing and time-warping [20]. However, in the previous literature, time-series augmentation has not been considered for multisensor signals of manufacturing systems. Moreover, self-supervised learning methods for manufacturing systems have mainly focused on overcoming the lack of labeled data.

Modern large-scale manufacturing processes that collect real-time multisensor signals have created a need for advanced predictive models to represent the features of the sensor signals and identify abnormal states. In particular, in the early detection of abnormal signals, effective representation learning with data augmentations is required to guarantee predictive performance because, otherwise, abnormal symptoms may not become noticeable early enough to prevent an incident. It also necessitates data augmentations to enforce models to learn various scales and patterns of abnormal signals.

However, most existing approaches have largely focused on manual feature extraction methods of multisensor signals, revealing limitations for real-world applicability. Specifically, the traditional supervised learning approaches have overlooked data augmentation designed for time-series data in the context of multisensor signals in manufacturing contexts. While recent studies have made strides in leveraging self-supervised learning for representation learning, they have focused on utilizing unlabeled data, neglecting the potential benefits of fully labeled data.

Our study addresses these limitations by introducing supervised contrastive learning (SCL) that leverages data augmentations and contrastive learning to maximize the utility of fully labeled data. Time-series augmentation [18] provides additional data of various scales and patterns, and contrastive learning based on labels increases the representation ability of class-wise features of multisensor signals. The SCL plays a role as the encoder pretraining strategy. It helps improve the model's performance by initializing it with meaningful representations. Then, it is followed by classifier training by transfer learning (with a frozen pretrained encoder) or fine-tuning (with a trainable pretrained encoder) to build a final prediction model.

This study aims to improve the predictive performance in classification problems of multisensor signals using SCL. We address both single-label and multilabel classification problems in benchmark and case study scenarios, demonstrating the comprehensive effectiveness of our method. In particular, in the case study, we apply SCL for early fault detection, which requires an advanced method to facilitate the identification of abnormal features for early prediction in manufacturing processes.

Although SCL was originally proposed for image data classification, we believe this approach can also excel in the classification of multisensor signals. In this study, we propose using SCL to increase the performance of multisensor signals classification. The main contributions of this study can be summarized as follows.

1) We investigate the effectiveness of SCL leveraging the principles of data augmentations and contrastive learning framework in the classification of multisensor signals. To the best of our knowledge, this is the first attempt to apply the SCL strategy to multisensor signal data.
2) We adopt time-series augmentations to reflect the various signal patterns in encoder pretraining. In both supervised learning and SCL methods, this highlights the versatility and robustness of the approach across various signal representations.
3) We evaluate our method in both single-label and multilabel classification problems. Extensive experiments involving different model structures and augmentation types provide a comprehensive understanding and rationale of the proposed method's effectiveness.
4) We demonstrate the applicability of our method for early fault detection in the real-world manufacturing process. We introduce data preprocessing methods for constructing input and output data for early prediction and apply our SCL for maximizing the discernment of features of early alarms, particularly pertinent in automobile engine manufacturing scenarios.

The rest of this article is organized as follows. Section II introduces related work. Section III describes the details of the methodology. Sections IV and V describe the performance of our method under benchmark datasets and a case study, respectively. Finally, Section VI contains our concluding remarks.

II. Related Work

A. Multisensor Signals Classification in Industries

Table I summarizes the comparisons of state-of-the-art training methods and their applications. Deep learning models based on supervised learning are the most well-known approaches

TABLE I
COMPARISONS OF TRAINING METHODS

Methods	Applications	Comparisons
Supervised Crossentropy	Multisensor signals [3], [11], [13], [21], [22], [23], [24], [25], [26], [27]	No augmentations
Self-supervised Contrastive [17]	Multisensor signals [29-31]	Unsupervised pretraining
Supervised Contrastive [18]	Image [18], Text [33]	Fully supervised pretraining

for multisensor signals classification. Qiao et al. [3] proposed a time-distributed spatiotemporal feature-learning method for machine health. Lee et al. [21] proposed a CNN structure in which a receptive field tailored to multisensor signals slides along the time axis to extract fault feature maps and perform fault detection. In addition, Yao et al. [22] attempted fault diagnosis with CNN and a temporal attention mechanism to extract meaningful temporal parts from the sensor signals. Jing et al. [23] used CNNs for multiple fault classification by constructing two-dimensional multisensor data. Chen et al. [24] proposed deep neural networks (DNNs) based time-series modeling for equipment reliability analysis. Zhao et al. [11] developed CNN variants that use soft thresholding to improve feature learning capability from noisy signal patterns and diagnosis accuracy. Liu et al. [13] developed a diagnosis framework based on the characteristics of industrial vibration signals. They added a dislocate layer in the CNN model to extract the relationship between signals with different intervals in periodic mechanical signals. In the healthcare industry, deep learning models have been widely applied to prediction systems of patient outcomes [25]. Khan et al. [26] proposed a modified CNN to predict heart disease using patients' blood pressure and electrocardiogram sensor data obtained from wearable devices. Ali et al. [27] introduced a smart healthcare monitoring system for heart disease prediction using ensemble deep learning based on multiple physiological sensor data. However, it is worth noting that the majority of studies have not attempted representation learning through data augmentation techniques in modern manufacturing processes that necessitate learning from various patterns of sensor data (see Table I).

Gupta et al. [16] applied zero-shot learning with a semantic information-based early classification approach using multivariate time series data for early classification of an unseen fault. Zhou et al. [12] applied a few-shot learning with a Siamese CNN to alleviate the overfitting problem and enhance anomaly detection accuracy in industrial cyber-physical systems. These approaches serve as alternatives that aim to address the challenges associated with limited labeled data.

The two-stage approach is also a popular strategy for multisensor signals classification. It consists of an unsupervised pretraining stage to extract compressed features and a supervised fine-tuning stage for classifier training. Xie et al. [14] used a multisignals-to-RGB-image conversion method based on principal component analysis to fuse multisensor signal data in 3-D images and applied it to CNN. Chen et al. [28] proposed sparse autoencoder-based feature extraction of multiple accelerometer sensors and performed classification with a deep belief network to diagnose faults. Luo et al. [15] also constructed sparse autoencoders and DNNs using vibration signals for the early detection of faults in machine tools.

As another two-stage strategy, self-supervised learning approaches for sensor signals have been attracting attention as a means to overcome the lack of labeled data. Zhang et al. [29] proposed a self-supervised contrastive learning method using vibration signals for fault diagnosis. They applied wavelet transforms to extract features from original signals and multi-instance contrastive. Von et al. [30] proposed self-supervised learning of disentangled-variational-autoencoder with a temporal CNN for anomaly detection. Senanayaka et al. [31] proposed an online fault diagnosis framework in electric powertrains without using labeled sensor signal data. The self-supervised learning of CNN was included in their framework. Zhang et al. [32] proposed a self-supervised joint learning fault diagnosis method based on three-channel vibration images to reduce the dependence on labeled data.

Although these approaches present promising solutions to overcome the need for labeled data, our focus remains on enhancing classification performance with fully labeled data. In addition, the relevant studies mentioned above have not considered data augmentation techniques for representation learning. We, thus, propose SCL with time-series augmentation for representation learning of multisensor signals that can use label information effectively.

B. Supervised Contrastive Learning

SCL was first proposed to improve image data classification performance, but it is the most relevant method for our study. Khosla et al. [18] extended the self-supervised batch contrastive approach to a fully supervised setting to leverage label information effectively. They analyzed the effectiveness of supervised contrastive loss (SupConLoss) that pulls together the features of the same classes (*positive* samples) in embedding space while simultaneously pushing apart features of different classes (*negative* samples). As a result, the *SCL* achieved a top-1 accuracy of 81.4% on the ImageNet dataset using ResNet-200 and consistently outperformed cross entropy on other datasets and two residual network (ResNet) variants.

In the field of natural language processing, the state-of-the-art classification models also follow two-stage modeling: pretraining a large language model on an auxiliary task and using cross-entropy loss to fine-tune the model on a task-specific labeled dataset [33]. However, Gunel et al. [33] noted that cross-entropy loss has several shortcomings that can lead to suboptimal generalization and instability. They combined cross entropy and their SCL loss to improve over a strong RoBERTa-Large baseline [34]. The point is that they used text data to apply an SCL objective instead of using it for pretraining to fine-tune the pretrained language model. They verified the performance of their method with the general language understanding evaluation benchmark datasets and few-shot learning settings without data augmentations. Existing studies have shown the effect of using

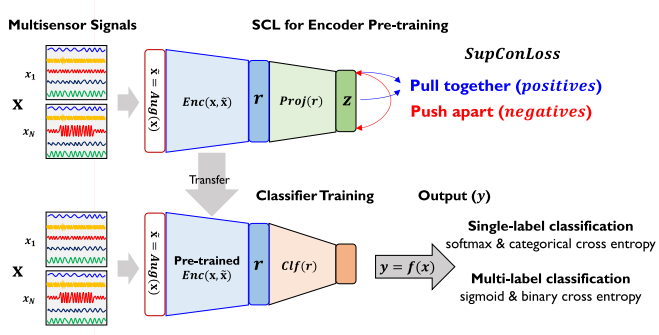


Fig. 1. Overview of our methodology: 1) SCL for encoder pretraining and 2) classifier training with the pretrained encoder.

SCL for image and text data, but its use has not yet been considered in multisensor signal data (see Table I). In this study, we exploit the effectiveness of SCL in multisensor signals classification.

III. Methodology

The predictive modeling framework for multisensor signals classification involves two stages: 1) SCL for encoder pretraining to produce representation vectors of inputs so that representations of multisensor signals in the same class will be more similar compared with representations of multisensor signals in different classes and 2) training a classifier on top of the pretrained encoder. Fig. 1 shows the overview of SCL's use of multisensor signal data.

1) For SCL encoder pretraining, we first apply data augmentation $\mathrm{Aug}(\mathbf{x})$ to obtain an additional batch $\tilde{\mathbf{x}}$. The multiviewed batch $(\mathbf{x}, \tilde{\mathbf{x}})$ is forwarded to the encoder network $\mathrm{Enc}(\mathbf{x}, \tilde{\mathbf{x}})$ to extract features $\boldsymbol{r}$. This resulting $\boldsymbol{r}$ is further put into the projection network $\mathrm{Proj}(\boldsymbol{r})$ to obtain a normalized embedding vector $\boldsymbol{z}$, and it is used for computing supervised contrastive loss (SupConLoss). Within $\boldsymbol{z} = (z_1, \ldots, z_i, \ldots, z_{2N})$, z_i is an *anchor*, and *positive* and *negative* pairs are determined if they have the same or different labels with an *anchor*. During training, the $SupConLoss$ function encourages the model to pull *positive* pairs closer together and to push *negative* pairs apart in the learned feature space.
2) The SCL-based pretrained encoder is transferred to the classifier $Clf(\cdot)$ training stage. For the classifier training, transfer learning, freezing all the pretrained layers, or fine-tuning, which allows updating the pretrained layers, can be applied to specific classification tasks. For the classifier training, the SCL-based pretrained encoder improves the classification performance by initializing the encoder of the classifier to generate meaningful representations.

A. SCL for Encoder Pretraining

The encoder pretraining framework is composed of the following three main components: data augmentation module $\mathrm{Aug}(\cdot)$, encoder network $\mathrm{Enc}(\cdot)$, and projection network $\mathrm{Proj}(\cdot)$.

$\boldsymbol{Aug}(\cdot)$: Given the N number of raw data $\mathbf{x}$, the $\mathrm{Aug}(\cdot)$ generates an additional N number of random augmented data $\tilde{\mathbf{x}} = \mathrm{Aug}(\mathbf{x})$ where $\mathbf{x}$ is the input batch of time-series $\mathbf{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_t, \ldots, \mathrm{x}_T)$ with T number of time steps, and in this study, each element x_t is multivariate. The random time-series augmentation provides different augmented data for each epoch and minibatch during pretraining. For instance, the flipping augmentation randomly selects and alters sensor patterns. It allows the model to learn abnormal patterns, such as fluctuations, regardless of the specific time indices of their occurrence. It contributes to enhancing the robustness of pretrained models by exposing them to diverse patterns within multisensor signals. In addition, it enables us to analyze multiviewed data $(\mathbf{x}, \tilde{\mathbf{x}})$ in SCL.

$\boldsymbol{Enc}(\cdot)$: The multiviewed data $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are input to the $\mathrm{Enc}(\cdot)$ as $\boldsymbol{r} = \mathrm{Enc}(\mathbf{x}, \tilde{\mathbf{x}}) \in \mathcal{R}^{D_E}$. Section IV gives the details of the time-series augmentation methods and the architecture of $\mathrm{Enc}(\cdot)$. The resulting features $\boldsymbol{r}$ are represented to the unit hypersphere $\mathcal{R}^{D_E}$ and $D_E$refers to the hyperparameter indicating the output dimension of the $\mathrm{Enc}(\cdot)$. It is then fed into the $\mathrm{Proj}(\cdot)$, resulting in $\boldsymbol{z} = \mathrm{Proj}(\boldsymbol{r}) \in \mathcal{R}^{D_P}$.

$\boldsymbol{Proj}(\cdot)$: The $\mathrm{Proj}(\cdot)$ consists of a fully-connected network and an l_2 normalization layer. The dense layer projects features to a specific dimension $\mathcal{R}^{D_P}$ and D_P indicates the output dimension of $\mathrm{Proj}(\cdot)$. The l_2 normalization layer normalizes the projected values, enabling us to calculate the distance matrix in SupConLoss. Thus, if we let $i \in I \equiv \{1, \ldots, 2N\}$ be the index of the raw and augmented input batch, the representations $\boldsymbol{z} = (z_1, \ldots, z_i, \ldots, z_{2N})$ are trained with the following SupConLoss:

$$\mathrm{SupConLoss} = \sum_{i \in I} -\log \left\{ \frac{1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)} \right\} \quad (1)$$

where i is the index of *anchor*, $a \in A(i)$ is the index of all *positives* and *negatives* in the batch distinct from i, and $p \in P(i)$ is the index of *positives*. $|P(i)|$ is the number of *positive* samples. The symbol $\cdot$ denotes the inner product and the temperature $\tau \in \mathcal{R}^+$ is a scalar temperature hyperparameter. Smaller τ benefits training more than higher ones but extremely low temperatures are difficult to train because of numerical instability [18]. The similarity of the numerator $(z_i \cdot z_p)$ must be maximized to optimize SupConLoss. In other words, the SupConLoss computes loss from labelwise paired data and encourages the encoder to give closely aligned representations to all entries from the same class.

B. Classifier Training

$\boldsymbol{Clf}(\cdot)$: The classifier training stage aims to train a classifier $Clf(\cdot)$ by taking an encoder learned by SCL. We transfer the pretrained $\mathrm{Enc}(\cdot)$ and stack a fully connected (FC) layer on top of the encoder. A larger number of FC layers can derive greater performance, but in this article, we stacked only one layer

TABLE II
SUMMARY OF BENCHMARK DATASETS

	Datasets	Number of observations	Number of dimensions	Number of time steps	Number of classes
1	Heartbeat	409	61	405	2
2	NATOPS	360	24	51	6
3	PEMS-SF	440	963	144	7
4	Racket Sports	303	6	30	4
5	SelfRegulationSCP1	561	6	896	2
6	SelfRegulationSCP2	380	7	1152	2
7	UWaveGestureLibrary	440	3	315	8

to exploit the representation performance of an SCL-trained encoder.

In multiclass classification, the output data are composed of the one-hot encoded matrix or multihot encoded matrix for single-label classification and multiple-label classification, respectively. The FC layer has a softmax activation and a categorical cross-entropy loss function for single-label classification and a sigmoid activation function and binary-cross entropy loss function for multiple-label classification.

Compared to supervised learning, our approach involves a two-step learning framework, including encoder pretraining by SCL and classifier training. In particular, time-series augmentations were performed in the pretraining step. Compared to self-supervised learning, SCL considers precise *positive* and *negative* samples based on label information. This contributes to initializing the model with discriminative representations across class labels (see Table I).

IV. BENCHMARK EXPERIMENTS

A. Datasets

We examined the effectiveness of the SCL in terms of its classification performance. We obtained seven benchmark datasets composed of multivariate time-series data. All came from the repository of the University of East Anglia and from the repository of the University of California, Riverside [35]. Table II lists a summary of the datasets with their various observations, dimensions, time steps, and classes. We used a standard scaler to perform dimensionwise normalization to make time-series values to a mean of zero and a standard deviation of one for each dimension.

B. Time-Series Augmentations

We experimented with seven different implementations of Aug$(\cdot)$, including flipping (horizontal), jittering, scaling, and their combinations. We also experimented with no augmentation—termed None—to compare a supervised baseline and SCL without data augmentation.

The flipping (horizontal) randomly selects target sensors and horizontally flips the time-series patterns as follows:

$$\tilde{\mathbf{x}} = \mathrm{x}_T \ \ldots, \ \mathrm{x}_t, \ \ldots \mathrm{x}_1. \tag{2}$$

Jittering and scaling are magnitude domain transformations. The jittering adds some noise named ϵ,and we sampled ϵ from the Gaussian distribution of mean $\mu = 0$ and standard deviation $\sigma = 0.03$ as follows:

$$\tilde{\mathbf{x}} = \mathrm{x}_1 \ + \epsilon_1, \ \ldots, \ \mathrm{x}_t + \epsilon_t, \ \ldots, \ \mathrm{x}_T + \epsilon_T. \tag{3}$$

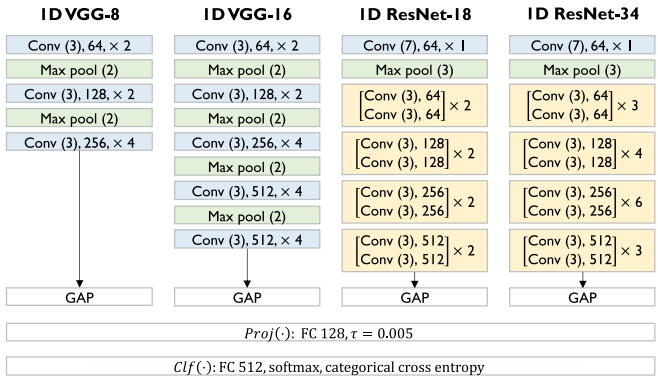


Fig. 2. Parameters of deep learning models with 1D-CNN encoders.

The scaling was conducted by multiplying random scalars as follows:

$$\tilde{\mathbf{x}} = \mathrm{x}_1 \ \alpha_1, \ \ldots, \ \mathrm{x}_t\alpha_t, \ldots, \ \mathrm{x}_T\alpha_T \tag{4}$$

where each α_t was sampled from the Gaussian distribution $\mathcal{N}(1, \ \sigma^2)$ and we set the standard deviation σ as 0.2. In addition, we experimented with all combinations of each augmentation method.

C. Encoder Architectures

To evaluate the SCL with time-series augmentations, we used CNN structures as encoders, including a visual geometry group (VGG) and [36] and ResNet [37] as state-of-the-art in time-series recognition [20]. We used VGG-8, VGG-16, ResNet-18, and ResNet-34 to evaluate the performance of various models with different architectures. In particular, we used 1-D convolutional layers that see the channels of CNN inputs as dimensions of multivariate time-series data. Fig. 2 shows the parameters of encoders. In the VGG network of $Enc(\cdot)$, we replaced FC layers with the global average pooling layer. For Proj$(\cdot)$, we set the number of nodes of an FC and τ as 128 and 0.005, respectively. After pretraining the encoder, $Clf(\cdot)$ also has an FC of 512 nodes with a softmax activation function and cross-entropy loss function to perform single-label multiclass classification.

D. Experiments

We performed fivefold cross validation using 80% and 20% ratios to split the training and validation data, respectively. We trained the encoder with a batch size of 20, an epoch size of 100, and an Adam optimizer for all datasets. We also implemented an early stopping criterion in which if the validation loss did not improve after 10 epochs, the training would terminate. All models were programmed in Python using TensorFlow 2.6 backend. The following GitHub repository provides all codes and Python environments to reproduce our results: https://github.com/YoonSangCho/MultisensorSCL. We used the following microaveraged F1-score $(Micro\ F1)$ to evaluate the performance in the multiclass classification task:

$$\text{Micro F1} = 2\frac{\text{miP} \times \text{miR}}{\text{miP} + \text{miR}} \tag{5}$$

where the microrecall (miR) is $\sum_c^C \mathrm{TP}_c / \sum_c^C (\mathrm{TP}_c + \mathrm{FN}_c)$ and the microprecision (miP) is $\sum_c^C \mathrm{TP}_c / \sum_c^C (\mathrm{TP}_c + \mathrm{FP}_c)$. The C is the number of classes, TP, FP, and FN, respectively, indicate true positives, false positives, and false negatives. In multiclass classification, a positive is a specific c class, and the negative is all the rest. The $\mathrm{Micro\ F1}$ gives equal weight to each observation regardless of its class by considering the total number of TP, FN, and FN. It measures precision and recall across all classes and provides a global measure of the overall performance of the model.

Table III lists the results for each encoder architecture. We compared supervised learning, SCL with a frozen encoder, and SCL with a trainable encoder. The time-series augmentations were conducted in all experiments. Each result presents the averages and standard deviations of fivefold cross-validation results. We highlighted in bold the best performances for each encoder architecture and underlined the performances of the best training strategy for each dataset. A summary of the experimental results from Table III is presented as follows.

1) The best training strategy: Our SCL consistently achieved the best performances except for Dataset 6. From Dataset 1 to Dataset 5 and Dataset 7, the proposed SCL with either transfer learning or fine-tuning with time-series augmentations yielded the highest performances. For Dataset 6, supervised learning exhibited the best performance but time-series augmentations (scaling and flipping) were required (see Table III).
2) Supervised learning with time-series augmentations: Likewise, time-series augmentation demonstrated a significant performance improvement in supervised learning. For example, for the 1D-VGG-8 of Dataset 4, the supervised learning without augmentations achieved 0.6152 (0.0439), whereas the supervised learning with scaling augmentation resulted in 0.8536 (0.0333) (see Table III).
3) Supervised learning versus SCL (without augmentations): Our SCL, with either transfer learning or fine-tuning, consistently outperformed supervised learning even without augmentations in most cases. For example, with the 1D-ResNet-34 encoder in Dataset 3, supervised learning (none) attained 0.7553 (0.0508), whereas our SCL with transfer learning (none) achieved 0.9036 (0.0446) (see Table III).
4) Supervised learning versus SCL (with augmentations): Comparing all augmentation combinations for each encoder and dataset, our SCL exhibited outstanding performance over supervised learning except in seven cases: Datasets 2, 4, and 6 of the 1D-VGG-16, Datasets 1, 2, and 6 of the 1D-ResNet-18, and Dataset 1 of the 1D-ResNet-34 (see Table III). However, we also observed that the time-series augmentations improved supervised learning performance. We can conclude that the pretrained encoders provided effective representation features with various augmentation methods.

V. Case Study

Machine health monitoring plays a pivotal role in improving operational efficiency within manufacturing industries through predictive maintenance. This involves the development of models capable of predicting operational failures of process equipment in advance, with the prediction outcomes serving as scientific guidance for planning maintenance activities. However, in the context of automobile engine manufacturing, a significant challenge is the early identification of abnormal states. Because of the postmaintenance system in automobile engine manufacturing, the occurrence of alarms of equipment failure or abnormal process problems causes an interruption in manufacturing operation. Once an alarm occurs, the process is interrupted until the problem is resolved, leading to a significant shutdown and subsequent operation restart. Thus, classifying multisensor signals that monitor machinery health by predicting alarms early in automobile engine manufacturing is necessary. In addition, data representing equipment states and advanced predictive modeling are required to extract abnormal features from complex patterns. To address this, this section delves into an investigation of the effectiveness and applicability of our SCL for early alarm detection. The evaluation is conducted using real-world multisensor signal data acquired from automobile engine manufacturing.

A. Multisensor Signal Data in Automobile Engine Manufacturing

The goal of early alarm detection is the use of previously observed multisensor signals to predict various types of future alarms. Fig. 3 illustrates multisensor signals and the preprocessing steps of the input and output data for early alarm detection in manufacturing automobile engines. The production process consists of various processes and equipment. We collected multisensor signal data from two manufacturing plants with three process lines (see Fig. 3). Each process line consists of various pieces of equipment. The characteristics of multisensor signal patterns are determined by process lines since they perform different tasks for each other. During production, each piece of equipment generates alarms when the process is not normal. We obtained signal data with 42 sensors, including the load rate and revolutions per minute of machinery tools, and they are attached to the different axes for each piece of process equipment. We constructed the final six analysis datasets by integrating multisensor signal data obtained from all equipment in six different process lines. Table IV lists a summary of the datasets. The number of classes indicates the count per class, and each alarm can co-occur.

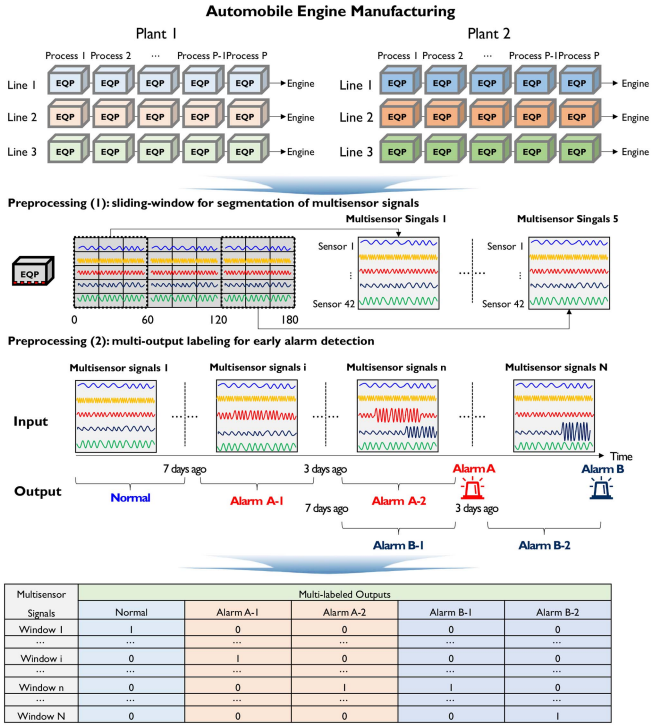
We then conducted the following preprocessing steps to perform early alarm detection.

1) We first used the sliding-window method to segment the multisensor signals into 2-D windows of fixed time steps (t) as input data. By allowing overlapping time steps (k), we aimed to capture signal patterns to the maximum extent (see Fig. 3). For this study, we set t and k at 60 and 30 s, respectively.
2) We then labeled input data with previous times of alarms as output data (see Fig. 3). For this study, we defined the early prediction time as 3 days and 7 days as per the requirements specified by domain experts. The output data thus consisted of multiple labels, also known as

TABLE III
RESULTS OF BENCHMARK EXPERIMENTS USING 1D-CNN ARCHITECTURES FOR EACH TRAINING STRATEGY

Methods			Benchmark Datasets						
Encoder	Training	Augmentation	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6	Dataset 7
1D-VGG-8	Supervised Learning	None	0.7384 (0.0000)	0.9555 (0.0225)	0.8631 (0.0228)	0.6152 (0.0439)	0.7487 (0.4149)	0.5368 (0.0385)	0.0000 (0.0504)
		Flipping	0.7262 (0.0000)	0.8712 (0.0106)	0.8971 (0.0457)	0.7320 (0.0424)	0.7683 (0.2462)	0.5316 (0.0368)	0.0000 (0.0736)
		Jittering	0.7310 (0.0000)	0.9444 (0.0212)	0.8049 (0.0644)	0.6879 (0.0337)	0.7130 (0.1473)	0.4895 (0.1316)	0.0000 (0.0410)
		Jittering and Flipping	0.7335 (0.0000)	0.8967 (0.0202)	0.8697 (0.0618)	0.5232 (0.0389)	0.7505 (0.3909)	0.5342 (0.0486)	0.0000 (0.0412)
		Jittering and Scaling	0.7408 (0.0000)	0.9569 (0.0239)	0.8377 (0.0222)	0.7305 (0.1000)	0.6968 (0.4105)	0.4974 (0.1257)	0.0000 (0.0430)
		Scaling	0.7481 (0.0000)	0.9471 (0.0345)	0.8676 (0.0272)	0.8536 (0.0333)	0.7380 (0.1531)	0.5000 (0.0373)	0.0000 (0.0492)
		Scaling and Flipping	0.7237 (0.0000)	0.8488 (0.0059)	0.8538 (0.0750)	0.6546 (0.0683)	0.7166 (0.2076)	0.5053 (0.0771)	0.0000 (0.0555)
		Scaling and Jittering and Flipping	0.7286 (0.0000)	0.8383 (0.0195)	0.8570 (0.0379)	0.8146 (0.0283)	0.7024 (0.1719)	0.5158 (0.1035)	0.0000 (0.0469)
	SCL + Transfer learning (proposed)	None	0.7555 (0.5957)	0.9639 (0.0248)	0.8792 (0.0319)	0.8493 (0.0297)	0.8022 (0.0418)	0.4842 (0.0198)	**0.5957 (0.0216)**
		Flipping	0.7407 (0.5146)	0.8806 (0.0451)	0.8812 (0.0376)	0.8463 (0.0425)	**0.8289 (0.0359)**	0.5079 (0.0365)	0.5146 (0.0412)
		Jittering	0.7482 (0.5636)	**0.9722 (0.0317)**	0.8964 (0.0220)	0.8508 (0.0354)	0.8182 (0.0729)	**0.5368 (0.0256)**	0.5636 (0.0365)
		Jittering and Flipping	0.7798 (0.5159)	0.8472 (0.0504)	**0.9165 (0.0450)**	0.8383 (0.0295)	0.8218 (0.0544)	0.5184 (0.0338)	0.5159 (0.0390)
		Jittering and Scaling	0.7578 (0.4978)	0.9442 (0.0392)	0.9030 (0.0328)	0.8407 (0.0278)	0.8022 (0.0440)	0.5237 (0.0219)	0.4978 (0.0459)
		Scaling	0.7237 (0.4916)	0.9403 (0.0745)	0.8982 (0.0292)	0.8685 (0.0372)	0.8039 (0.0391)	0.5368 (0.0302)	0.4916 (0.0504)
		Scaling and Flipping	0.7677 (0.5551)	0.8695 (0.0436)	0.8994 (0.0507)	0.8077 (0.0366)	0.8129 (0.0517)	0.5079 (0.0278)	0.5551 (0.0150)
		Scaling and Jittering and Flipping	**0.7799 (0.5189)**	0.8806 (0.0216)	0.8842 (0.0158)	0.8670 (0.0187)	0.8075 (0.0246)	0.5105 (0.0352)	0.5189 (0.0216)
	SCL + Fine-tuning (proposed)	None	0.7408 (0.3329)	0.9639 (0.0401)	0.6231 (0.0210)	0.9055 (0.2570)	0.7647 (0.0469)	0.5237 (0.0518)	0.3329 (0.0459)
		Flipping	0.7480 (0.2039)	0.9000 (0.0456)	0.6343 (0.0600)	0.8631 (0.2243)	0.7594 (0.0580)	0.5289 (0.0664)	0.2039 (0.0235)
		Jittering	0.7237 (0.2043)	0.9528 (0.0438)	0.7532 (0.0211)	0.9043 (0.1114)	0.7736 (0.0469)	0.4921 (0.0668)	0.2043 (0.0317)
		Jittering and Flipping	0.7604 (0.2051)	0.8507 (0.0105)	0.7452 (0.0880)	0.8576 (0.0624)	0.7736 (0.0504)	0.5184 (0.0683)	0.2051 (0.0390)
		Jittering and Scaling	0.7262 (0.2007)	0.9389 (0.0361)	0.7966 (0.0288)	**0.9058 (0.1081)**	0.7611 (0.0242)	0.5237 (0.0673)	0.2007 (0.0729)
		Scaling	0.7383 (0.0696)	0.9526 (0.0399)	0.6358 (0.0158)	0.8926 (0.1952)	0.7665 (0.0434)	0.5131 (0.0584)	0.0696 (0.0827)
		Scaling and Flipping	0.7384 (0.3683)	0.9167 (0.0241)	0.6617 (0.0439)	0.8340 (0.1767)	0.7540 (0.0608)	0.4895 (0.0656)	0.3683 (0.0487)
		Scaling and Jittering and Flipping	0.7506 (0.2431)	0.9056 (0.0364)	0.6442 (0.0505)	0.8542 (0.2192)	0.7754 (0.0200)	0.5158 (0.0962)	0.2431 (0.0598)
1D-VGG-16	Supervised Learning	None	0.7187 (0.0526)	0.9139 (0.0244)	0.8288 (0.0576)	0.6858 (0.0575)	0.7647 (0.3862)	0.5105 (0.0673)	0.0526 (0.0450)
		Flipping	0.7384 (0.0000)	0.8734 (0.0576)	0.8239 (0.0692)	0.8508 (0.0989)	0.6881 (0.0326)	**0.5421 (0.1275)**	0.0000 (0.0365)
		Jittering	0.7359 (0.6863)	**0.9194 (0.0207)**	0.7222 (0.0769)	0.6724 (0.2543)	0.6794 (0.3784)	0.5105 (0.1506)	0.6863 (0.0253)
		Jittering and Flipping	0.6993 (0.0000)	0.9167 (0.0261)	0.6635 (0.0241)	0.5242 (0.3012)	0.7168 (0.4788)	0.5132 (0.0904)	0.0000 (0.0631)
		Jittering and Scaling	0.7065 (0.6312)	0.8873 (0.0165)	0.7545 (0.0486)	0.6976 (0.1446)	0.6933 (0.3911)	0.5105 (0.0761)	0.6312 (0.0365)
		Scaling	0.7188 (0.1909)	0.9094 (0.0412)	0.7473 (0.0610)	0.8057 (0.1979)	0.7380 (0.1488)	0.4974 (0.0341)	0.1909 (0.0253)
		Scaling and Flipping	0.7457 (0.0000)	0.8722 (0.0478)	0.7651 (0.0410)	0.6772 (0.2136)	0.7415 (0.3844)	0.4526 (0.0595)	0.0000 (0.0655)
		Scaling and Jittering and Flipping	0.7186 (0.0000)	0.8969 (0.0623)	0.6233 (0.0567)	**0.8693 (0.3036)**	0.7665 (0.0405)	0.5210 (0.0667)	0.0000 (0.0433)
	SCL + Transfer learning (proposed)	None	0.7286 (0.6738)	0.8412 (0.0158)	**0.8939 (0.0502)**	0.8069 (0.0511)	0.7772 (0.1002)	0.4947 (0.0349)	0.6738 (0.0547)
		Flipping	0.7237 (0.5308)	0.8167 (0.0182)	0.8260 (0.0669)	0.7486 (0.0650)	0.7754 (0.1162)	0.4816 (0.0464)	0.5308 (0.0498)
		Jittering	0.6334 (0.6372)	0.8789 (0.1996)	0.8906 (0.0425)	0.7952 (0.0178)	0.7576 (0.0459)	0.5158 (0.0408)	0.6372 (0.0521)
		Jittering and Flipping	0.7212 (0.4542)	0.8194 (0.0283)	0.8418 (0.0916)	0.8032 (0.0278)	0.7897 (0.0301)	0.5184 (0.0616)	0.4542 (0.0442)
		Jittering and Scaling	0.7189 (0.8377)	0.8805 (0.0520)	0.8896 (0.0467)	0.8513 (0.0205)	0.7647 (0.0528)	0.5289 (0.0279)	0.8377 (0.0469)
		Scaling	0.7237 (0.8232)	0.8889 (0.0350)	0.8836 (0.0405)	0.7788 (0.0256)	0.7772 (0.0631)	0.5000 (0.0316)	0.8232 (0.0360)
		Scaling and Flipping	0.7285 (0.6510)	0.8067 (0.0414)	0.8370 (0.0589)	0.5558 (0.0424)	0.7879 (0.4012)	0.5026 (0.0282)	0.6510 (0.0365)
		Scaling and Jittering and Flipping	0.6529 (0.6146)	0.7694 (0.1684)	0.8786 (0.0320)	0.7921 (0.0348)	**0.7915 (0.0485)**	0.4974 (0.0340)	0.6146 (0.0235)
	SCL + Fine-tuning (proposed)	None	0.7457 (0.8675)	0.8861 (0.0374)	0.4301 (0.0907)	0.8561 (0.3127)	0.7487 (0.0395)	0.5053 (0.0296)	0.8675 (0.0343)
		Flipping	0.7237 (0.4815)	0.7697 (0.0104)	0.8136 (0.1245)	0.8148 (0.2010)	0.6629 (0.0794)	0.5131 (0.1017)	0.4815 (0.0767)
		Jittering	0.7261 (0.7118)	0.7261 (0.0479)	0.6091 (0.2388)	0.7917 (0.3661)	0.7290 (0.0922)	0.4842 (0.1006)	0.7118 (0.0667)
		Jittering and Flipping	**0.7603 (0.4390)**	0.8762 (0.0299)	0.5727 (0.0360)	0.8447 (0.2375)	0.6968 (0.0487)	0.4974 (0.1077)	0.4390 (0.0469)
		Jittering and Scaling	0.6993 (0.8727)	0.8417 (0.0243)	0.6529 (0.0507)	0.8676 (0.2706)	0.6741 (0.0585)	0.5105 (0.1419)	**0.8727 (0.0538)**
		Scaling	0.7164 (0.7711)	0.8667 (0.0694)	0.5828 (0.0610)	0.8348 (0.3230)	0.6862 (0.0345)	0.5026 (0.1099)	0.7711 (0.0855)
		Scaling and Flipping	0.7310 (0.5778)	0.8833 (0.0464)	0.6182 (0.0825)	0.8088 (0.2941)	0.7648 (0.0900)	0.4579 (0.0178)	0.5778 (0.0253)
		Scaling and Jittering and Flipping	0.7262 (0.4782)	0.8500 (0.0305)	0.6795 (0.0949)	0.8249 (0.3032)	0.7399 (0.0352)	0.4947 (0.0633)	0.4782 (0.0662)
1D-ResNet-18	Supervised Learning	None	0.7358 (0.7098)	0.9195 (0.0373)	0.8619 (0.0267)	0.7503 (0.0504)	0.7665 (0.0354)	0.5132 (0.0480)	0.7098 (0.0542)
		Flipping	0.7189 (0.6893)	0.8775 (0.0444)	0.7850 (0.0388)	0.7393 (0.1427)	0.7629 (0.0687)	0.5316 (0.0244)	0.6893 (0.0461)
		Jittering	0.7334 (0.9418)	**0.9235 (0.0304)**	0.8661 (0.0542)	0.7744 (0.0273)	0.7419 (0.1220)	0.5421 (0.1140)	0.9418 (0.0577)
		Jittering and Flipping	0.7504 (0.3768)	0.8425 (0.0452)	0.8936 (0.0893)	0.7845 (0.0260)	0.7611 (0.0382)	0.5237 (0.0334)	0.3768 (0.0216)
		Jittering and Scaling	0.7310 (0.9133)	0.8874 (0.0199)	0.8374 (0.0773)	0.7679 (0.1053)	0.7736 (0.0629)	0.4973 (0.0416)	0.9133 (0.0172)
		Scaling	0.7213 (0.9135)	0.8432 (0.0232)	0.8454 (0.0786)	0.7885 (0.0431)	0.7915 (0.0462)	0.4974 (0.0291)	0.9135 (0.0469)
		Scaling and Flipping	**0.7579 (0.5575)**	0.8491 (0.0207)	0.8556 (0.0240)	0.7377 (0.0582)	0.7843 (0.0660)	**0.5553 (0.0210)**	0.5575 (0.0410)
		Scaling and Jittering and Flipping	0.7407 (0.5841)	0.8371 (0.0391)	0.7751 (0.0674)	0.7956 (0.1333)	0.7701 (0.0476)	0.5316 (0.0329)	0.5841 (0.0273)
	SCL + Transfer learning (proposed)	None	0.7285 (0.9203)	0.8469 (0.0311)	0.8641 (0.0452)	0.7929 (0.0247)	**0.7985 (0.0375)**	0.4868 (0.0202)	0.9203 (0.0690)
		Flipping	0.7041 (0.7308)	0.8111 (0.0303)	0.8802 (0.0552)	0.7745 (0.0310)	0.7791 (0.0557)	0.5026 (0.0344)	0.7308 (0.0399)
		Jittering	0.7163 (0.9475)	0.8802 (0.0352)	0.8754 (0.0218)	0.7889 (0.0376)	0.7701 (0.0498)	0.5237 (0.0471)	0.9475 (0.0144)
		Jittering and Flipping	0.7115 (0.7080)	0.8829 (0.0530)	0.8905 (0.0350)	0.7866 (0.0728)	0.7950 (0.0527)	0.5368 (0.0280)	0.7080 (0.0314)
		Jittering and Scaling	0.6821 (0.9432)	0.8329 (0.0735)	0.8856 (0.0214)	0.8031 (0.0593)	0.7629 (0.0558)	0.5053 (0.0423)	0.9432 (0.0239)
		Scaling	0.7236 (0.9360)	0.8250 (0.0308)	**0.9080 (0.0320)**	0.7276 (0.0574)	0.7736 (0.0509)	0.4816 (0.0258)	0.9360 (0.0461)
		Scaling and Flipping	0.6749 (0.7592)	0.8244 (0.0297)	0.8906 (0.0565)	0.7773 (0.0339)	0.7772 (0.0445)	0.5158 (0.0420)	0.7592 (0.0430)
		Scaling and Jittering and Flipping	0.7138 (0.7163)	0.8562 (0.0339)	0.8782 (0.0276)	0.7621 (0.0704)	0.7754 (0.0275)	0.5026 (0.0250)	0.7163 (0.0620)
	SCL + Fine-tuning (proposed)	None	0.7287 (0.9273)	0.8917 (0.0430)	0.8919 (0.0704)	0.8331 (0.0389)	0.7504 (0.0221)	0.5473 (0.0616)	0.9273 (0.0379)
		Flipping	0.7091 (0.6574)	0.9056 (0.0917)	0.7896 (0.0332)	0.7881 (0.1908)	0.7593 (0.0488)	0.5026 (0.0344)	0.6574 (0.0459)
		Jittering	0.7409 (0.9477)	0.8829 (0.0153)	0.7909 (0.0675)	0.7871 (0.1155)	0.7044 (0.0877)	0.5342 (0.1268)	**0.9477 (0.0718)**
		Jittering and Flipping	0.7285 (0.7241)	0.8917 (0.0539)	0.6803 (0.0360)	0.8152 (0.3165)	0.7577 (0.0144)	0.4921 (0.0612)	0.7241 (0.0356)
		Jittering and Scaling	0.7042 (0.9215)	0.9056 (0.0321)	0.9055 (0.0475)	**0.8646 (0.0362)**	0.7595 (0.0518)	0.5105 (0.0430)	0.9215 (0.0353)
		Scaling	0.7383 (0.9143)	0.9012 (0.0360)	0.6704 (0.0601)	0.7867 (0.2648)	0.7557 (0.0297)	0.5263 (0.0782)	0.9143 (0.0501)
		Scaling and Flipping	0.7141 (0.7312)	0.8472 (0.0847)	0.8782 (0.0673)	0.8396 (0.0604)	0.7291 (0.0221)	0.5000 (0.0262)	0.7312 (0.0465)
		Scaling and Jittering and Flipping	0.7188 (0.7185)	0.9012 (0.0557)	0.7331 (0.0234)	0.7932 (0.2427)	0.7808 (0.0351)	0.5263 (0.0296)	0.7185 (0.0474)
1D-ResNet-34	Supervised Learning	None	0.7359 (0.9167)	0.8658 (0.0331)	0.7553 (0.0508)	0.6748 (0.0917)	0.7362 (0.0717)	0.4789 (0.0277)	0.9167 (0.0303)
		Flipping	0.7407 (0.5620)	0.7962 (0.0294)	0.7370 (0.1109)	0.7579 (0.1597)	0.7665 (0.0509)	0.4763 (0.0386)	0.5620 (0.0699)
		Jittering	0.7139 (0.8241)	0.8234 (0.0270)	0.8486 (0.0584)	0.7709 (0.0642)	0.7203 (0.0438)	0.4842 (0.0722)	0.8241 (0.0410)
		Jittering and Flipping	**0.7602 (0.7269)**	0.8117 (0.0467)	0.7978 (0.0491)	0.7732 (0.1323)	0.7594 (0.0413)	0.5053 (0.0420)	0.7269 (0.0200)
		Jittering and Scaling	0.7212 (0.8849)	0.8306 (0.0168)	0.7901 (0.0361)	0.7537 (0.1419)	0.7028 (0.0786)	0.4790 (0.1603)	0.8849 (0.0343)
		Scaling	0.7310 (0.9064)	0.8602 (0.0229)	0.7678 (0.0293)	0.6799 (0.0988)	0.7647 (0.0545)	0.5184 (0.0509)	0.9064 (0.0593)
		Scaling and Flipping	0.7310 (0.6948)	0.7732 (0.0495)	0.6464 (0.0802)	0.7587 (0.2605)	0.7826 (0.0539)	0.5053 (0.0492)	0.6948 (0.0200)
		Scaling and Jittering and Flipping	0.7213 (0.7741)	0.7938 (0.0433)	0.7836 (0.0611)	0.7816 (0.0907)	0.7522 (0.0618)	0.5000 (0.0198)	0.7741 (0.0406)
	SCL + Transfer learning (proposed)	None	0.7212 (0.9318)	0.8417 (0.0175)	0.9036 (0.0446)	0.7285 (0.0500)	0.7719 (0.0308)	0.4658 (0.0412)	0.9318 (0.0412)
		Flipping	0.7115 (0.7026)	0.8134 (0.0265)	0.8764 (0.0241)	0.7652 (0.0474)	0.7293 (0.0721)	0.4895 (0.1005)	0.7026 (0.0460)
		Jittering	0.7115 (0.9292)	0.8329 (0.0075)	**0.9066 (0.0426)**	0.7431 (0.0389)	0.7789 (0.0536)	0.5105 (0.0300)	0.9292 (0.0388)
		Jittering and Flipping	0.7237 (0.7393)	0.7928 (0.0143)	0.8568 (0.0355)	0.7765 (0.0481)	0.7504 (0.0273)	0.4869 (0.0830)	0.7393 (0.0581)
		Jittering and Scaling	0.7165 (0.9430)	0.8300 (0.0307)	0.9055 (0.0192)	0.7269 (0.0535)	**0.7861 (0.0658)**	0.4790 (0.0284)	**0.9430 (0.0506)**
		Scaling	0.7018 (0.9396)	0.8102 (0.0345)	0.8861 (0.0805)	0.7366 (0.0323)	0.7736 (0.0407)	0.5105 (0.0130)	0.9396 (0.0216)
		Scaling and Flipping	0.7042 (0.7098)	0.7576 (0.0403)	0.8879 (0.0826)	0.7953 (0.0492)	0.7683 (0.0347)	0.5053 (0.0415)	0.7098 (0.0506)
		Scaling and Jittering and Flipping	0.7017 (0.7144)	0.7994 (0.0421)	0.8802 (0.0670)	0.7951 (0.0278)	0.7736 (0.0479)	0.5105 (0.0484)	0.7144 (0.0634)
	SCL + Fine-tuning (proposed)	None	0.6919 (0.9296)	0.8788 (0.1316)	0.8553 (0.0847)	0.8084 (0.0539)	0.7220 (0.0469)	0.5105 (0.0267)	0.9296 (0.0591)
		Flipping	0.7237 (0.6514)	0.8602 (0.0494)	0.8076 (0.0649)	**0.8247 (0.0821)**	0.7486 (0.0688)	0.4710 (0.0701)	0.6514 (0.0641)
		Jittering	0.7164 (0.9114)	**0.9111 (0.0479)**	0.8744 (0.0602)	0.7953 (0.0817)	0.7183 (0.0508)	0.4921 (0.0327)	0.9114 (0.0390)
		Jittering and Flipping	0.7310 (0.7027)	0.8455 (0.0253)	0.8432 (0.1141)	0.7952 (0.0907)	0.7148 (0.0628)	0.5316 (0.1192)	0.7027 (0.0578)
		Jittering and Scaling	0.7212 (0.9385)	0.8833 (0.0292)	0.8619 (0.0663)	0.8053 (0.0670)	0.7025 (0.0376)	0.5132 (0.0568)	0.9385 (0.0677)
		Scaling	0.6992 (0.9193)	0.9051 (0.0441)	0.8884 (0.0558)	0.7943 (0.0440)	0.7307 (0.0652)	**0.5316 (0.0445)**	0.9193 (0.0199)
		Scaling and Flipping	0.7530 (0.7331)	0.8611 (0.0480)	0.7932 (0.0741)	0.7791 (0.2043)	0.7346 (0.0493)	0.5237 (0.0830)	0.7331 (0.0365)
		Scaling and Jittering and Flipping	0.7383 (0.7297)	0.8833 (0.0402)	0.6850 (0.0412)	0.7854 (0.2670)	0.7096 (0.0457)	0.4605 (0.0795)	0.7297 (0.0588)

Micro F1 Scores in fivefold cross validations. Standard deviations are presented in parentheses.



Multisensor Signals	Multi-labeled Outputs				
	Normal	Alarm A-1	Alarm A-2	Alarm B-1	Alarm B-2
Window 1	1	0	0	0	0
...	...	...	...	...	...
Window i	0	1	0	0	0
...	...	...	...	...	...
Window n	0	0	1	1	0
...	...	...	...	...	...
Window N	0	0	0	0	1

Fig. 3. Illustration of multisensor signal data in automobile engine manufacturing. To design early-alarm prediction experiments, we conducted a two-step process: 1) application of a sliding window to construct input data, and 2) labeling the windows indicating early prediction times of specific alarms. Alarm A-1 and Alarm A-2 labels were assigned to windows retrieved from 7 days to 3 days and from 3 days to the occurrence of Alarm A, respectively. Alarm B-1 and Alarm B-2 labels were assigned to windows using the same method. If windows had not been assigned any alarm labels, we labeled them as normal. The *n*th window exhibits the multilabeled outputs. The prediction of the *n*th window as Alarms A-2 and B-1 indicates that Alarm A and Alarm B will occur within 3 days and 7 days. We use windows of multisensor signals and early prediction times of alarms as input and output of the prediction model.

TABLE IV
SUMMARY OF MULTISENSOR SIGNAL DATA IN AUTOMOBILE ENGINE MANUFACTURING

Dataset	Number of observations	Number of sensors	Number of time steps	Number of classes
1	84728	42	60 (s)	17
2	59286	42	60 (s)	9
3	91194	42	60 (s)	19
4	93164	42	60 (s)	11
5	71782	42	60 (s)	7
6	8594	42	60 (s)	5

multihot encoded data, and each input is classified into multiple alarms and time indices.

We applied our SCL to leverage the label information for extracting alarm-related early features of multisensor signals. We hypothesized that the input sensor data could describe symptoms of alarms, and the proposed SCL also works well under multilabeled class classification.

B. Experimental Details

In this case study, we added VGG-8, VGG-16, ResNet-18, and ResNet-34 using 2-D convolutional layers as an encoder that

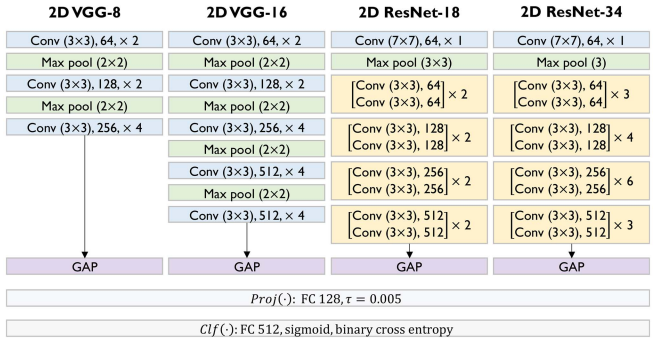


Fig. 4. Parameters of deep learning models with 2D-CNN encoders.

can model time series data as image data because the number of sensors is sufficient to build input data as an image. Fig. 4 shows the parameters of encoders. Thus, the input shape of height, width, and channel is the number of sensors, time steps, and one, respectively. For the Clf$(\cdot)$, we applied the sigmoid activation function and binary cross-entropy loss function to enable multilabel classification. Compared to single-label classification, multiple labels can be assigned to each observation. We, thus, evaluated the performance with the following Jaccard score:

$$\text{Jaccard score} = \frac{1}{N}\sum_{i=1}^{N}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i + \text{FP}_i} \tag{6}$$

where N is the number of observations, and TP_i, FN_i, and FP_i, respectively, are the true positive, false negative, and false positive for each observation. The $\text{TP}_i/\text{TP}_i + \text{FN}_i + \text{FP}_i$ is the Jaccard index and it measures the size of the intersection area divided by the size of the union of the predicted and actual labels for the positive class. We averaged the Jaccard indices of all testing observations. In this case study, we used the batch size of 256 in the training process in fivefold cross validations.

Tables V and VI list the results of case study experiments using 1D-CNN and 2D-CNN architectures, respectively. We highlighted in bold the best performances for each encoder type and underlined the performances of the best training strategy for each dataset in Tables V and VI, respectively. In addition, Table VII was specifically constructed to show the training strategies that yielded the best performance for each dataset. The summary of the experiment results is provided as follows.

1) The best training strategy: Our SCL consistently achieved the best performances except for Dataset 6 (see Table VII). From Dataset 1 to Dataset 5, the proposed SCL + transfer learning and various augmentations achieved the highest performances. For Dataset 6, supervised learning demonstrated the highest performance but time-series augmentation (scaling and jittering and flipping) was required (see Table VII).
2) Supervised learning with time-series augmentations: We observed that time-series augmentation provided a performance improvement of supervised learning in many cases. For example, for the 1D-VGG-16 of Dataset 6, supervised learning (none) achieved 0.5538 (0.2364), whereas supervised learning (scaling) achieved 0.7128

TABLE V
RESULTS OF CASE STUDY EXPERIMENTS USING 1D-CNN ARCHITECTURES FOR EACH TRAINING STRATEGY

Methods			Automobile engine manufacturing datasets					
Encoder	Training	Augmentation	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
1D-VGG-8	Supervised Learning	None (baseline)	0.8422 (0.0229)	0.7237 (0.0351)	0.7530 (0.0097)	0.7929 (0.0211)	0.7490 (0.0201)	0.8079 (0.0385)
		Flipping	0.8465 (0.0184)	0.7390 (0.0422)	0.7589 (0.0101)	0.8208 (0.0279)	0.7712 (0.0350)	0.8131 (0.0258)
		Jittering	0.8423 (0.0196)	0.6703 (0.0469)	0.7499 (0.0202)	0.7912 (0.0319)	0.7584 (0.0311)	0.8042 (0.0292)
		Jittering and Flipping	0.8486 (0.0214)	0.7493 (0.0189)	0.7615 (0.0105)	0.8165 (0.0366)	0.7832 (0.0337)	0.8217 (0.0164)
		Jittering and Scaling	0.8349 (0.0176)	0.6789 (0.0332)	0.7492 (0.0076)	0.7998 (0.0081)	0.7320 (0.0276)	0.8108 (0.0251)
		Scaling	0.8394 (0.0139)	0.6676 (0.0364)	0.7465 (0.0105)	0.7893 (0.0203)	0.7129 (0.0260)	0.7786 (0.0693)
		Scaling and Flipping	0.8598 (0.0089)	0.7095 (0.0288)	0.7606 (0.0084)	0.8100 (0.0219)	0.7310 (0.0244)	0.8062 (0.0049)
		Scaling and Jittering and Flipping	0.8484 (0.0164)	0.7275 (0.0412)	0.7703 (0.0094)	0.8099 (0.0226)	0.7548 (0.0230)	**0.8292 (0.0198)**
	SCL + Transfer learning (proposed)	None	0.8606 (0.0178)	0.7436 (0.0439)	0.7770 (0.0044)	0.8359 (0.0260)	0.7966 (0.0300)	0.8092 (0.0030)
		Flipping	**0.8804 (0.0160)**	**0.7830 (0.0128)**	0.7889 (0.0064)	0.8341 (0.0424)	0.8039 (0.0301)	0.7991 (0.0111)
		Jittering	0.8766 (0.0220)	0.7482 (0.0171)	0.7678 (0.0103)	0.8290 (0.0218)	0.7712 (0.0349)	0.8001 (0.0190)
		Jittering and Flipping	0.8700 (0.0254)	0.7724 (0.0430)	0.7894 (0.0059)	0.7897 (0.0433)	**0.8110 (0.0247)**	0.7931 (0.0126)
		Jittering and Scaling	0.8608 (0.0124)	0.7437 (0.0152)	0.7679 (0.0107)	0.8178 (0.0212)	0.7297 (0.0302)	0.8011 (0.0075)
		Scaling	0.8647 (0.0054)	0.7465 (0.0155)	0.7686 (0.0078)	0.7900 (0.0616)	0.7524 (0.0286)	0.7918 (0.0286)
		Scaling and Flipping	0.8704 (0.0113)	0.7522 (0.0283)	0.7842 (0.0115)	0.8212 (0.0299)	0.7668 (0.0306)	0.8007 (0.0166)
		Scaling and Jittering and Flipping	0.8648 (0.0173)	0.7652 (0.0082)	**0.7907 (0.0089)**	**0.8366 (0.0130)**	0.7638 (0.0228)	0.8036 (0.0143)
	SCL + Fine-tuning (proposed)	None	0.8376 (0.0284)	0.7437 (0.0126)	0.7534 (0.0093)	0.8178 (0.0268)	0.7564 (0.0525)	0.8110 (0.0171)
		Flipping	0.8578 (0.0181)	0.7755 (0.0069)	0.7688 (0.0067)	0.8209 (0.0166)	0.7976 (0.0234)	0.8133 (0.0246)
		Jittering	0.8435 (0.0274)	0.7404 (0.0135)	0.7554 (0.0040)	0.8198 (0.0090)	0.7593 (0.0362)	0.7948 (0.0320)
		Jittering and Flipping	0.8617 (0.0234)	0.7662 (0.0138)	0.7742 (0.0070)	0.8082 (0.0229)	0.7861 (0.0260)	0.8182 (0.0259)
		Jittering and Scaling	0.8518 (0.0047)	0.7369 (0.0227)	0.7468 (0.0126)	0.8140 (0.0147)	0.7130 (0.0441)	0.7845 (0.0183)
		Scaling	0.8537 (0.0064)	0.7382 (0.0198)	0.7433 (0.0123)	0.8026 (0.0143)	0.7242 (0.0423)	0.8065 (0.0161)
		Scaling and Flipping	0.8694 (0.0024)	0.7372 (0.0334)	0.7685 (0.0103)	0.8197 (0.0047)	0.7444 (0.0254)	0.8079 (0.0404)
		Scaling and Jittering and Flipping	0.8611 (0.0133)	0.7542 (0.0114)	0.7673 (0.0117)	0.8326 (0.0066)	0.7466 (0.0287)	0.8158 (0.0296)
1D VGG-16	Supervised Learning	None	0.8004 (0.0199)	0.6757 (0.0246)	0.7258 (0.0159)	0.7780 (0.0315)	0.7149 (0.0489)	0.5538 (0.2364)
		Flipping	0.8232 (0.0239)	0.6851 (0.0242)	0.7571 (0.0151)	0.7384 (0.0858)	0.7374 (0.0399)	0.6450 (0.1404)
		Jittering	0.8256 (0.0182)	0.6986 (0.0623)	0.7280 (0.0211)	0.7409 (0.0259)	0.7335 (0.0180)	0.6531 (0.1253)
		Jittering and Flipping	0.8195 (0.0249)	0.6820 (0.0569)	0.7467 (0.0228)	0.7838 (0.0642)	0.7389 (0.0200)	0.5919 (0.1536)
		Jittering and Scaling	0.8047 (0.0169)	0.6285 (0.0483)	0.7159 (0.0246)	0.7738 (0.0217)	0.7361 (0.0268)	0.6751 (0.1292)
		Scaling	0.8199 (0.0146)	0.6430 (0.0348)	0.7196 (0.0215)	0.7729 (0.0223)	0.7314 (0.0274)	0.7128 (0.0756)
		Scaling and Flipping	0.8191 (0.0151)	0.6908 (0.0298)	0.7361 (0.0123)	0.7828 (0.0470)	0.7332 (0.0264)	0.6552 (0.0992)
		Scaling and Jittering and Flipping	0.8394 (0.0155)	0.7075 (0.0175)	0.7443 (0.0102)	0.7821 (0.0282)	0.7461 (0.0265)	0.7001 (0.0658)
	SCL + Transfer learning (proposed)	None	0.7633 (0.1698)	0.6775 (0.0991)	0.7529 (0.0087)	0.6411 (0.1170)	0.7612 (0.0516)	0.7248 (0.0514)
		Flipping	0.7857 (0.0971)	0.7208 (0.0600)	0.7703 (0.0193)	0.7430 (0.1335)	**0.7795 (0.0308)**	0.7245 (0.0855)
		Jittering	0.8462 (0.0222)	0.6955 (0.0498)	0.7506 (0.0138)	0.7952 (0.0280)	0.7604 (0.0480)	0.5953 (0.1765)
		Jittering and Flipping	0.8146 (0.0919)	0.6790 (0.0852)	0.7674 (0.0061)	0.7729 (0.0537)	0.7586 (0.0282)	0.5580 (0.1140)
		Jittering and Scaling	0.8384 (0.0194)	0.6894 (0.0510)	0.7452 (0.0062)	0.7453 (0.0448)	0.7293 (0.0247)	0.7490 (0.0203)
		Scaling	0.8257 (0.0204)	0.6772 (0.0623)	0.7461 (0.0070)	0.7280 (0.0853)	0.7508 (0.0145)	0.7103 (0.0932)
		Scaling and Flipping	0.8498 (0.0204)	0.6907 (0.0583)	0.7645 (0.0178)	0.7325 (0.0819)	0.7330 (0.0376)	0.6708 (0.0930)
		Scaling and Jittering and Flipping	**0.8627 (0.0116)**	0.7419 (0.0461)	**0.7717 (0.0073)**	0.7313 (0.0792)	0.6961 (0.0876)	0.7396 (0.0611)
	SCL + Fine-tuning (proposed)	None	0.8215 (0.0212)	0.7372 (0.0277)	0.7502 (0.0093)	0.7346 (0.0681)	0.7570 (0.0163)	0.7163 (0.0930)
		Flipping	0.8133 (0.0708)	0.7375 (0.0074)	0.7653 (0.0155)	0.7989 (0.0410)	0.7686 (0.0265)	0.7443 (0.0596)
		Jittering	0.8132 (0.0723)	0.7051 (0.0215)	0.7383 (0.0143)	0.7834 (0.0323)	0.7585 (0.0067)	0.6474 (0.1420)
		Jittering and Flipping	0.8250 (0.0821)	0.7181 (0.0382)	0.7700 (0.0054)	**0.8120 (0.0227)**	0.7395 (0.0301)	**0.7906 (0.0407)**
		Jittering and Scaling	0.8345 (0.0220)	0.7010 (0.0372)	0.7310 (0.0134)	0.7864 (0.0218)	0.7393 (0.0169)	0.7208 (0.0498)
		Scaling	0.8185 (0.0277)	0.6942 (0.0472)	0.7362 (0.0091)	0.7787 (0.0216)	0.7272 (0.0217)	0.7451 (0.0917)
		Scaling and Flipping	0.8482 (0.0202)	0.7257 (0.0433)	0.7642 (0.0113)	0.7772 (0.0709)	0.7459 (0.0300)	0.7760 (0.0283)
		Scaling and Jittering and Flipping	0.8553 (0.0145)	**0.7493 (0.0131)**	0.7676 (0.0150)	0.7938 (0.0423)	0.7369 (0.0268)	0.7701 (0.0324)
1D-ResNet-18	Supervised Learning	None	0.7632 (0.0252)	0.5590 (0.0487)	0.6958 (0.0126)	0.7201 (0.0196)	0.6900 (0.0148)	0.7097 (0.0114)
		Flipping	0.7821 (0.0118)	0.6091 (0.0116)	0.7072 (0.0081)	0.6832 (0.0761)	0.7005 (0.0164)	0.7383 (0.0132)
		Jittering	0.7570 (0.0315)	0.5446 (0.0220)	0.6942 (0.0131)	0.6995 (0.0180)	0.6894 (0.0122)	0.7177 (0.0064)
		Jittering and Flipping	0.7675 (0.0219)	0.5824 (0.0225)	0.7173 (0.0138)	0.7262 (0.0340)	0.6993 (0.0045)	0.7354 (0.0127)
		Jittering and Scaling	0.7493 (0.0174)	0.5227 (0.0200)	0.6951 (0.0124)	0.6570 (0.0207)	0.6716 (0.0120)	0.7071 (0.0202)
		Scaling	0.7477 (0.0142)	0.5605 (0.0311)	0.6892 (0.0112)	0.6684 (0.0324)	0.6609 (0.0223)	0.6731 (0.0222)
		Scaling and Flipping	0.7872 (0.0141)	0.5675 (0.0154)	0.7134 (0.0037)	0.7235 (0.0177)	0.6864 (0.0046)	0.7325 (0.0169)
		Scaling and Jittering and Flipping	0.7764 (0.0093)	0.5604 (0.0539)	0.7146 (0.0079)	0.7172 (0.0341)	0.6777 (0.0240)	0.7319 (0.0252)
	SCL + Transfer learning (proposed)	None	0.8034 (0.0294)	0.6132 (0.0130)	0.7312 (0.0048)	0.7265 (0.0165)	0.6953 (0.0303)	0.7364 (0.0132)
		Flipping	0.8151 (0.0062)	**0.6357 (0.0099)**	**0.7463 (0.0023)**	0.7318 (0.0279)	**0.7270 (0.0081)**	0.7485 (0.0034)
		Jittering	0.7993 (0.0122)	0.6035 (0.0291)	0.7246 (0.0085)	0.7401 (0.0134)	0.7031 (0.0192)	**0.7577 (0.0117)**
		Jittering and Flipping	**0.8233 (0.0091)**	0.6266 (0.0207)	0.7456 (0.0108)	**0.7482 (0.0074)**	0.7241 (0.0088)	0.7446 (0.0088)
		Jittering and Scaling	0.7766 (0.0202)	0.5990 (0.0249)	0.7185 (0.0103)	0.7163 (0.0157)	0.6705 (0.0140)	0.7455 (0.0134)
		Scaling	0.7676 (0.0137)	0.5933 (0.0181)	0.7261 (0.0106)	0.7077 (0.0140)	0.6669 (0.0175)	0.7555 (0.0134)
		Scaling and Flipping	0.7955 (0.0162)	0.6150 (0.0148)	0.7330 (0.0029)	0.7147 (0.0247)	0.7060 (0.0063)	0.7538 (0.0066)
		Scaling and Jittering and Flipping	0.7915 (0.0258)	0.6175 (0.0079)	0.7407 (0.0087)	0.7200 (0.0314)	0.6977 (0.0091)	0.7404 (0.0257)
	SCL + Fine-tuning (proposed)	None	0.7889 (0.0250)	0.5995 (0.0116)	0.7103 (0.0085)	0.7268 (0.0114)	0.6791 (0.0253)	0.7477 (0.0079)
		Flipping	0.7953 (0.0113)	0.6250 (0.0144)	0.7287 (0.0051)	0.7242 (0.0164)	0.7064 (0.0137)	0.7513 (0.0110)
		Jittering	0.7836 (0.0120)	0.5976 (0.0266)	0.7135 (0.0076)	0.7086 (0.0189)	0.6804 (0.0186)	0.7524 (0.0152)
		Jittering and Flipping	0.7958 (0.0172)	0.6178 (0.0302)	0.7311 (0.0060)	0.7372 (0.0211)	0.6991 (0.0094)	0.7499 (0.0181)
		Jittering and Scaling	0.7708 (0.0184)	0.5844 (0.0256)	0.7004 (0.0145)	0.7121 (0.0120)	0.6608 (0.0150)	0.7371 (0.0083)
		Scaling	0.7658 (0.0118)	0.5836 (0.0167)	0.7059 (0.0134)	0.6966 (0.0139)	0.6498 (0.0193)	0.7441 (0.0212)
		Scaling and Flipping	0.7936 (0.0052)	0.6069 (0.0147)	0.7196 (0.0089)	0.7193 (0.0297)	0.6896 (0.0134)	0.7554 (0.0221)
		Scaling and Jittering and Flipping	0.7980 (0.0121)	0.6160 (0.0146)	0.7222 (0.0112)	0.7326 (0.0089)	0.6793 (0.0069)	0.7534 (0.0173)
1D-ResNet-34	Supervised Learning	None	0.7632 (0.0135)	0.5536 (0.0333)	0.6964 (0.0146)	0.6704 (0.0677)	0.6690 (0.0390)	0.7297 (0.0168)
		Flipping	0.7754 (0.0369)	0.6023 (0.0104)	0.7083 (0.0178)	0.6875 (0.0742)	0.7053 (0.0145)	0.7216 (0.0069)
		Jittering	0.7201 (0.0540)	0.5288 (0.0276)	0.6931 (0.0084)	0.6782 (0.0195)	0.6805 (0.0103)	0.7089 (0.0317)
		Jittering and Flipping	0.7735 (0.0092)	0.5910 (0.0192)	0.7111 (0.0131)	0.7022 (0.0769)	0.6945 (0.0192)	0.7240 (0.0278)
		Jittering and Scaling	0.7380 (0.0140)	0.5450 (0.0374)	0.6902 (0.0221)	0.6720 (0.0159)	0.6564 (0.0210)	0.6919 (0.0333)
		Scaling	0.7531 (0.0186)	0.5467 (0.0250)	0.6886 (0.0081)	0.6794 (0.0244)	0.6462 (0.0040)	0.7080 (0.0295)
		Scaling and Flipping	0.7781 (0.0103)	0.5576 (0.0483)	0.7089 (0.0054)	0.7021 (0.0321)	0.6716 (0.0163)	0.7186 (0.0137)
		Scaling and Jittering and Flipping	0.7746 (0.0115)	0.5619 (0.0197)	0.7042 (0.0084)	0.6927 (0.0198)	0.6855 (0.0120)	0.7238 (0.0329)
	SCL + Transfer learning (proposed)	None	0.7771 (0.0353)	0.5636 (0.0387)	0.7266 (0.0067)	0.7148 (0.0176)	0.6969 (0.0282)	0.7029 (0.0103)
		Flipping	0.8125 (0.0171)	**0.6315 (0.0151)**	0.7304 (0.0128)	0.7316 (0.0228)	0.7131 (0.0097)	0.7227 (0.0316)
		Jittering	0.7893 (0.0088)	0.5763 (0.0314)	0.7284 (0.0077)	0.6917 (0.0235)	0.6996 (0.0102)	0.7078 (0.0278)
		Jittering and Flipping	**0.8186 (0.0170)**	0.6287 (0.0128)	**0.7395 (0.0097)**	0.7047 (0.0109)	**0.7146 (0.0162)**	0.7302 (0.0339)
		Jittering and Scaling	0.7623 (0.0170)	0.5839 (0.0284)	0.7205 (0.0043)	0.6980 (0.0135)	0.6650 (0.0117)	0.7052 (0.0291)
		Scaling	0.7621 (0.0266)	0.5824 (0.0203)	0.7190 (0.0101)	0.6965 (0.0268)	0.6332 (0.0462)	0.7078 (0.0322)
		Scaling and Flipping	0.7970 (0.0138)	0.6048 (0.0234)	0.7325 (0.0074)	0.7074 (0.0261)	0.6824 (0.0130)	0.7205 (0.0296)
		Scaling and Jittering and Flipping	0.7945 (0.0126)	0.5993 (0.0061)	0.7354 (0.0088)	0.7022 (0.0295)	0.6791 (0.0184)	0.6891 (0.0513)
	SCL + Fine-tuning (proposed)	None	0.7822 (0.0137)	0.5765 (0.0272)	0.7169 (0.0105)	0.7187 (0.0131)	0.6842 (0.0173)	0.7358 (0.0098)
		Flipping	0.8067 (0.0306)	0.6280 (0.0280)	0.7224 (0.0083)	**0.7362 (0.0200)**	0.7098 (0.0122)	0.7502 (0.0152)
		Jittering	0.7755 (0.0189)	0.5843 (0.0128)	0.7113 (0.0117)	0.7040 (0.0273)	0.6862 (0.0118)	0.7463 (0.0129)
		Jittering and Flipping	0.8109 (0.0181)	0.6291 (0.0143)	0.7332 (0.0098)	0.7271 (0.0168)	0.7018 (0.0153)	**0.7611 (0.0254)**
		Jittering and Scaling	0.7702 (0.0149)	0.5896 (0.0217)	0.7102 (0.0070)	0.6862 (0.0210)	0.6615 (0.0079)	0.7409 (0.0084)
		Scaling	0.7745 (0.0116)	0.5910 (0.0099)	0.7108 (0.0147)	0.6932 (0.0144)	0.6628 (0.0177)	0.7415 (0.0197)
		Scaling and Flipping	0.7902 (0.0135)	0.6127 (0.0131)	0.7294 (0.0092)	0.7288 (0.0085)	0.6883 (0.0107)	0.7544 (0.0262)
		Scaling and Jittering and Flipping	0.7933 (0.0081)	0.6122 (0.0073)	0.7252 (0.0092)	0.7134 (0.0149)	0.6823 (0.0151)	0.7466 (0.0294)

Jaccard Scores in fivefold cross validations. Standard deviations are presented in parentheses.

TABLE VI
RESULTS OF CASE STUDY EXPERIMENTS USING 2D-CNN ARCHITECTURES FOR EACH TRAINING STRATEGY

Methods			Automobile engine manufacturing datasets					
Encoder	Training	Augmentation	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5	Dataset 6
2D-VGG-8	Supervised Learning	None	0.8579 (0.0102)	0.7222 (0.0198)	0.7404 (0.0103)	0.8374 (0.0175)	0.7580 (0.0243)	0.4488 (0.1152)
		Flipping	0.8637 (0.0128)	0.7185 (0.0327)	0.7698 (0.0147)	0.8268 (0.0272)	0.7668 (0.0397)	0.4936 (0.0098)
		Jittering	0.8594 (0.0108)	0.6927 (0.0508)	0.7302 (0.0151)	0.8197 (0.0206)	0.7308 (0.0199)	0.4500 (0.0815)
		Jittering and Flipping	0.8669 (0.0350)	0.7186 (0.0239)	0.7640 (0.0103)	0.8256 (0.0352)	0.7674 (0.0271)	0.4353 (0.1125)
		Jittering and Scaling	0.8600 (0.0094)	0.6721 (0.0373)	0.7437 (0.0175)	0.7949 (0.0289)	0.7198 (0.0469)	0.4497 (0.1038)
		Scaling	0.8578 (0.0047)	0.6305 (0.0396)	0.7356 (0.0154)	0.8139 (0.0174)	0.7115 (0.0407)	0.3739 (0.1877)
		Scaling and Flipping	0.8749 (0.0032)	0.7481 (0.0293)	0.7635 (0.0107)	0.8225 (0.0185)	0.7487 (0.0135)	0.4111 (0.1982)
		Scaling and Jittering and Flipping	0.8586 (0.0259)	0.6937 (0.0545)	0.7661 (0.0117)	0.8261 (0.0101)	0.7520 (0.0413)	0.4002 (0.2234)
	SCL + Transfer learning (proposed)	None	0.8741 (0.0192)	0.7385 (0.0341)	0.7758 (0.0101)	0.8474 (0.0369)	0.7656 (0.0359)	0.3999 (0.3119)
		Flipping	0.8828 (0.0167)	**0.7896 (0.0198)**	**0.7996 (0.0076)**	0.8224 (0.0964)	**0.7937 (0.0376)**	0.3937 (0.2733)
		Jittering	0.8756 (0.0095)	0.7333 (0.0365)	0.7746 (0.0093)	**0.8582 (0.0202)**	0.7793 (0.0221)	0.3592 (0.2070)
		Jittering and Flipping	**0.8855 (0.0074)**	0.7760 (0.0272)	0.7945 (0.0076)	0.8480 (0.0294)	0.7904 (0.0166)	0.3425 (0.1473)
		Jittering and Scaling	0.8638 (0.0057)	0.6978 (0.0177)	0.7794 (0.0130)	0.8252 (0.0266)	0.7245 (0.0194)	0.3589 (0.2771)
		Scaling	0.8564 (0.0113)	0.7272 (0.0373)	0.7814 (0.0091)	0.8367 (0.0122)	0.7392 (0.0262)	0.5331 (0.2573)
		Scaling and Flipping	0.8760 (0.0087)	0.7237 (0.0544)	0.7858 (0.0088)	0.8239 (0.0313)	0.7448 (0.0159)	0.5218 (0.2789)
		Scaling and Jittering and Flipping	0.8748 (0.0051)	0.7486 (0.0439)	0.7932 (0.0099)	0.8393 (0.0199)	0.7444 (0.0289)	0.5101 (0.1796)
	SCL + Fine-tuning (proposed)	None	0.8458 (0.0139)	0.7501 (0.0324)	0.7537 (0.0131)	0.8177 (0.0113)	0.7358 (0.0284)	0.6760 (0.1666)
		Flipping	0.8577 (0.0160)	0.7581 (0.0298)	0.7716 (0.0187)	0.8272 (0.0397)	0.7814 (0.0289)	0.6833 (0.1687)
		Jittering	0.8481 (0.0163)	0.7046 (0.0520)	0.7482 (0.0105)	0.8427 (0.0141)	0.7356 (0.0455)	0.6246 (0.1713)
		Jittering and Flipping	0.8769 (0.0074)	0.7804 (0.0383)	0.7760 (0.0124)	0.8294 (0.0164)	0.7481 (0.0171)	0.5721 (0.1606)
		Jittering and Scaling	0.8270 (0.0202)	0.7081 (0.0245)	0.7529 (0.0090)	0.8114 (0.0212)	0.6897 (0.0253)	0.5552 (0.1236)
		Scaling	0.8292 (0.0174)	0.7110 (0.0288)	0.7467 (0.0114)	0.8144 (0.0119)	0.7135 (0.0325)	0.6845 (0.1734)
		Scaling and Flipping	0.8627 (0.0102)	0.7338 (0.0258)	0.7713 (0.0081)	0.8249 (0.0223)	0.7184 (0.0352)	0.6113 (0.1523)
		Scaling and Jittering and Flipping	0.8567 (0.0136)	0.7461 (0.0223)	0.7700 (0.0056)	0.8413 (0.0101)	0.7554 (0.0140)	**0.6950 (0.1786)**
2D-VGG-16	Supervised Learning	None	0.8457 (0.0226)	0.7001 (0.0538)	0.7426 (0.0170)	0.7544 (0.1266)	0.7657 (0.0189)	0.5058 (0.3503)
		Flipping	0.8259 (0.0778)	0.7310 (0.0392)	0.7621 (0.0117)	0.7684 (0.1062)	0.7636 (0.0318)	0.3932 (0.2814)
		Jittering	0.8631 (0.0156)	0.6846 (0.0400)	0.7349 (0.0145)	0.7636 (0.0907)	0.6748 (0.0936)	0.3931 (0.2597)
		Jittering and Flipping	0.8301 (0.0402)	0.7606 (0.0264)	0.7531 (0.0179)	0.7768 (0.0772)	0.7376 (0.0286)	0.6006 (0.2592)
		Jittering and Scaling	0.8587 (0.0046)	0.6780 (0.0247)	0.7374 (0.0091)	0.8019 (0.0194)	0.7246 (0.0317)	0.6889 (0.1705)
		Scaling	0.8461 (0.0086)	0.6796 (0.0516)	0.7407 (0.0146)	0.7827 (0.0251)	0.7525 (0.0160)	0.4215 (0.2337)
		Scaling and Flipping	0.8540 (0.0445)	0.7390 (0.0502)	0.7586 (0.0073)	0.8113 (0.0212)	0.7533 (0.0308)	0.2760 (0.3041)
		Scaling and Jittering and Flipping	0.8745 (0.0098)	0.7417 (0.0322)	0.7516 (0.0180)	0.8149 (0.0337)	0.7578 (0.0433)	0.3343 (0.2092)
	SCL + Transfer learning (proposed)	None	0.8597 (0.0153)	0.7105 (0.0809)	0.7813 (0.0033)	0.7631 (0.0329)	0.7794 (0.0349)	0.2999 (0.2738)
		Flipping	0.8619 (0.0289)	0.7600 (0.0228)	**0.7949 (0.0093)**	0.7861 (0.1082)	**0.7881 (0.0380)**	0.2997 (0.2736)
		Jittering	0.8626 (0.0337)	0.6721 (0.0872)	0.7773 (0.0061)	0.7825 (0.0695)	0.7386 (0.1083)	0.3001 (0.2739)
		Jittering and Flipping	0.8693 (0.0397)	0.7428 (0.0430)	0.7894 (0.0108)	0.7541 (0.1089)	0.7363 (0.0462)	0.1273 (0.1743)
		Jittering and Scaling	0.8593 (0.0329)	0.6632 (0.0494)	0.7693 (0.0103)	0.7985 (0.0282)	0.6955 (0.0815)	0.3001 (0.2739)
		Scaling	0.8643 (0.0203)	0.7302 (0.0275)	0.7620 (0.0069)	0.7658 (0.0344)	0.7103 (0.0308)	0.5846 (0.1168)
		Scaling and Flipping	0.8756 (0.0157)	0.7270 (0.0455)	0.7925 (0.0035)	0.7632 (0.0346)	0.7480 (0.0330)	0.4696 (0.0677)
		Scaling and Jittering and Flipping	**0.8770 (0.0187)**	0.6934 (0.0459)	0.7903 (0.0043)	0.7963 (0.0305)	0.7209 (0.0793)	0.3003 (0.2739)
	SCL + Fine-tuning (proposed)	None	0.8344 (0.0332)	0.7195 (0.0346)	0.7577 (0.0085)	0.8158 (0.0218)	0.7867 (0.0242)	0.6976 (0.1264)
		Flipping	0.8429 (0.0426)	0.7582 (0.0174)	0.7816 (0.0071)	0.8205 (0.0202)	0.7734 (0.0297)	0.6377 (0.2035)
		Jittering	0.8449 (0.0222)	0.7067 (0.0648)	0.7622 (0.0090)	0.8192 (0.0349)	0.7663 (0.0189)	0.7385 (0.0103)
		Jittering and Flipping	0.8238 (0.1015)	0.7594 (0.0261)	0.7819 (0.0038)	**0.8242 (0.0216)**	0.7742 (0.0080)	0.6451 (0.1076)
		Jittering and Scaling	0.8528 (0.0261)	0.6658 (0.0475)	0.7586 (0.0113)	0.8071 (0.0151)	0.7304 (0.0121)	0.7414 (0.0298)
		Scaling	0.8581 (0.0170)	0.7341 (0.0250)	0.7494 (0.0069)	0.7951 (0.0253)	0.7175 (0.0411)	0.5534 (0.2091)
		Scaling and Flipping	0.8698 (0.0074)	**0.7671 (0.0060)**	0.7757 (0.0151)	0.8121 (0.0214)	0.7447 (0.0147)	0.5831 (0.2606)
		Scaling and Jittering and Flipping	0.8715 (0.0078)	0.7341 (0.0349)	0.7783 (0.0043)	0.8211 (0.0239)	0.7528 (0.0115)	**0.7505 (0.0520)**
2D-ResNet-18	Supervised Learning	None	0.8125 (0.0246)	0.5342 (0.0484)	0.6937 (0.0122)	0.7386 (0.0286)	0.6877 (0.0166)	0.5461 (0.1084)
		Flipping	0.8300 (0.0188)	0.6052 (0.0357)	0.7189 (0.0092)	0.7643 (0.0432)	0.7179 (0.0225)	0.5339 (0.1391)
		Jittering	0.7935 (0.0301)	0.5465 (0.0219)	0.6874 (0.0190)	0.7142 (0.0197)	0.6664 (0.0627)	0.5900 (0.1305)
		Jittering and Flipping	0.8179 (0.0252)	0.6084 (0.0397)	0.7165 (0.0122)	0.7406 (0.0835)	0.7091 (0.0375)	0.5308 (0.1173)
		Jittering and Scaling	0.7885 (0.0162)	0.5515 (0.0334)	0.6867 (0.0121)	0.6986 (0.0146)	0.6655 (0.0104)	0.6610 (0.1166)
		Scaling	0.7645 (0.0228)	0.5554 (0.0253)	0.6789 (0.0068)	0.6948 (0.0224)	0.6740 (0.0114)	0.6858 (0.1023)
		Scaling and Flipping	0.8013 (0.0168)	0.5742 (0.0395)	0.7072 (0.0146)	0.7329 (0.0295)	0.6934 (0.0137)	0.6413 (0.1401)
		Scaling and Jittering and Flipping	0.8095 (0.0100)	0.5794 (0.0318)	0.7116 (0.0084)	0.7306 (0.0216)	0.6967 (0.0094)	0.5158 (0.1375)
	SCL + Transfer learning (proposed)	None	0.8391 (0.0120)	0.6000 (0.0318)	0.7357 (0.0087)	0.7256 (0.0285)	0.7259 (0.0236)	0.5779 (0.2277)
		Flipping	**0.8451 (0.0228)**	0.6735 (0.0118)	0.7464 (0.0091)	0.7673 (0.0396)	**0.7584 (0.0214)**	0.7575 (0.0223)
		Jittering	0.8255 (0.0199)	0.6227 (0.0254)	0.7398 (0.0121)	0.7372 (0.0184)	0.6995 (0.0330)	0.6400 (0.2562)
		Jittering and Flipping	0.8319 (0.0365)	0.6490 (0.0259)	0.7423 (0.0103)	**0.7776 (0.0183)**	0.6899 (0.0848)	0.7559 (0.0066)
		Jittering and Scaling	0.8174 (0.0058)	0.5622 (0.0426)	0.7252 (0.0114)	0.7113 (0.0235)	0.6748 (0.0268)	0.7367 (0.0269)
		Scaling	0.8209 (0.0069)	0.5687 (0.0216)	0.7293 (0.0112)	0.7183 (0.0324)	0.6775 (0.0279)	0.6522 (0.2010)
		Scaling and Flipping	0.8337 (0.0097)	0.6273 (0.0286)	**0.7466 (0.0083)**	0.7462 (0.0303)	0.7123 (0.0193)	0.7469 (0.0148)
		Scaling and Jittering and Flipping	0.8330 (0.0090)	0.6290 (0.0272)	0.7407 (0.0091)	0.7424 (0.0376)	0.7202 (0.0157)	0.7535 (0.0211)
	SCL + Fine-tuning (proposed)	None	0.8219 (0.0187)	0.5933 (0.0237)	0.7102 (0.0151)	0.7461 (0.0229)	0.6953 (0.0196)	0.7609 (0.0161)
		Flipping	0.8440 (0.0104)	**0.6645 (0.0154)**	0.7329 (0.0046)	0.7700 (0.0232)	0.7342 (0.0082)	0.7557 (0.0136)
		Jittering	0.8280 (0.0116)	0.6296 (0.0325)	0.7167 (0.0085)	0.7461 (0.0206)	0.6873 (0.0231)	0.7598 (0.0215)
		Jittering and Flipping	0.8300 (0.0164)	0.6387 (0.0257)	0.7313 (0.0071)	0.7686 (0.0171)	0.6956 (0.0555)	**0.7729 (0.0101)**
		Jittering and Scaling	0.8001 (0.0095)	0.5913 (0.0199)	0.7010 (0.0118)	0.7010 (0.0218)	0.6648 (0.0199)	0.7646 (0.0128)
		Scaling	0.8068 (0.0066)	0.5937 (0.0056)	0.7118 (0.0087)	0.7223 (0.0247)	0.6676 (0.0090)	0.7444 (0.0293)
		Scaling and Flipping	0.8274 (0.0106)	0.6268 (0.0164)	0.7289 (0.0034)	0.7458 (0.0216)	0.6844 (0.0253)	0.7689 (0.0233)
		Scaling and Jittering and Flipping	0.8289 (0.0043)	0.6272 (0.0199)	0.7275 (0.0127)	0.7408 (0.0293)	0.7005 (0.0075)	0.7664 (0.0202)
2D ResNet-34	Supervised Learning	None	0.8037 (0.0329)	0.5610 (0.0445)	0.6910 (0.0125)	0.7285 (0.0476)	0.6897 (0.0201)	0.6861 (0.1253)
		Flipping	0.8060 (0.0486)	0.6039 (0.0364)	0.7091 (0.0156)	0.7444 (0.0390)	0.6911 (0.0516)	0.7379 (0.0197)
		Jittering	0.7869 (0.0186)	0.5326 (0.0394)	0.6948 (0.0132)	0.7322 (0.0454)	0.6965 (0.0194)	0.7053 (0.1173)
		Jittering and Flipping	0.8010 (0.0288)	0.6063 (0.0384)	0.7170 (0.0076)	**0.7659 (0.0295)**	0.7185 (0.0423)	0.6437 (0.1372)
		Jittering and Scaling	0.7836 (0.0086)	0.5126 (0.0305)	0.6737 (0.0093)	0.7031 (0.0173)	0.6571 (0.0305)	0.7463 (0.0265)
		Scaling	0.7891 (0.0107)	0.5220 (0.0216)	0.6910 (0.0091)	0.6869 (0.0307)	0.6448 (0.0329)	0.7159 (0.0197)
		Scaling and Flipping	0.8126 (0.0123)	0.5579 (0.0404)	0.7135 (0.0095)	0.7098 (0.0148)	0.6787 (0.0119)	0.7006 (0.1207)
		Scaling and Jittering and Flipping	0.8152 (0.0133)	0.5822 (0.0509)	0.7132 (0.0108)	0.7228 (0.0334)	0.6949 (0.0208)	0.5531 (0.2602)
	SCL + Transfer learning (proposed)	None	0.8163 (0.0142)	0.5520 (0.0550)	0.7307 (0.0041)	0.6979 (0.0412)	0.6994 (0.0412)	0.7100 (0.0142)
		Flipping	0.8392 (0.0122)	0.5983 (0.0446)	0.7332 (0.0353)	0.7142 (0.0512)	**0.7380 (0.0212)**	0.7140 (0.0179)
		Jittering	0.8164 (0.0307)	0.5182 (0.0374)	0.7270 (0.0119)	0.7089 (0.0191)	0.6934 (0.0261)	0.7027 (0.0072)
		Jittering and Flipping	0.8385 (0.0102)	0.5992 (0.0313)	**0.7460 (0.0061)**	0.7299 (0.0292)	0.7292 (0.0179)	0.4326 (0.2899)
		Jittering and Scaling	0.7887 (0.0040)	0.5364 (0.0478)	0.7142 (0.0076)	0.6709 (0.0281)	0.6565 (0.0090)	0.7078 (0.0210)
		Scaling	0.7859 (0.0214)	0.4765 (0.0592)	0.7254 (0.0053)	0.6898 (0.0069)	0.6365 (0.0161)	0.7023 (0.0123)
		Scaling and Flipping	0.8177 (0.0130)	0.5741 (0.0508)	0.7334 (0.0080)	0.7279 (0.0198)	0.6835 (0.0220)	0.7120 (0.0274)
		Scaling and Jitterin and Flipping	0.8255 (0.0161)	0.5727 (0.0288)	0.7425 (0.0058)	0.7288 (0.0328)	0.6902 (0.0310)	0.5799 (0.3252)
	SCL + Fine-tuning (proposed)	None	0.8088 (0.0100)	0.6041 (0.0222)	0.7123 (0.0083)	0.7388 (0.0290)	0.6923 (0.0118)	0.7575 (0.0110)
		Flipping	**0.8406 (0.0113)**	**0.6438 (0.0304)**	0.7322 (0.0123)	0.7406 (0.0495)	0.7161 (0.0262)	0.7610 (0.0225)
		Jittering	0.8239 (0.0103)	0.5653 (0.0492)	0.7114 (0.0113)	0.7277 (0.0179)	0.6814 (0.0222)	**0.7650 (0.0188)**
		Jittering and Flipping	0.8324 (0.0185)	0.6355 (0.0217)	0.7295 (0.0087)	0.7428 (0.0214)	0.7260 (0.0191)	0.7601 (0.0065)
		Jittering and Scaling	0.7875 (0.0118)	0.5892 (0.0173)	0.7048 (0.0120)	0.6877 (0.0258)	0.6628 (0.0163)	0.7586 (0.0166)
		Scaling	0.7889 (0.0121)	0.5753 (0.0198)	0.7104 (0.0095)	0.6760 (0.0293)	0.6583 (0.0054)	0.7568 (0.0144)
		Scaling and Flipping	0.8156 (0.0098)	0.6254 (0.0120)	0.7344 (0.0035)	0.7390 (0.0141)	0.6913 (0.0158)	0.7549 (0.0305)
		Scaling and Jittering and Flipping	0.8172 (0.0208)	0.6262 (0.0194)	0.7370 (0.0053)	0.7268 (0.0080)	0.7011 (0.0141)	0.7633 (0.0157)

Jaccard Scores in fivefold cross validations. Standard deviations are presented in parentheses.

TABLE VII
BEST TRAINING STRATEGIES FOR EACH DATASET

Dataset	Encoder	Training	Augmentation	Performance
1	2D-VGG-8	SCL + transfer learning	Jittering and Flipping	0.8855 (0.0074)
2	2D-VGG-8	SCL + transfer learning	Flipping	0.7896 (0.0198)
3	2D-VGG-8	SCL + transfer learning	Flipping	0.7996 (0.0076)
4	2D-VGG-8	SCL + transfer learning	Jittering	0.8582 (0.0202)
5	1D-VGG-8	SCL + transfer learning	Jittering and Flipping	0.8110 (0.0247)
6	1D-VGG-8	Supervised Learning	Scaling and Jittering and Flipping	0.8292 (0.0198)

Performances are Jaccard scores in fivefold cross validations. Standard deviations are presented in parentheses.

(0.0756) (see Table V). Similarly, for the 2D-ResNet-34 of Dataset 4, the supervised learning (none) achieved 0.7285 (0.0476), whereas the supervised learning (jittering and flipping) achieved 0.7659 (0.0295) (see Table VI).

3) Supervised learning versus SCL (without augmentations): In the case of 1D-CNN encoders, our SCL, either with transfer learning or fine-tuning, consistently outperformed supervised learning without augmentations in most cases (see Table V). For the 1D-VGG-16 of Dataset 6, our SCL with transfer learning (none) achieved 0.7248 (0.0514) but the supervised learning (none) achieved 0.5538 (0.2364). There was one case in which our method underperformed: For the 1D-VGG-16 of Dataset 4, the supervised learning (none) achieved 0.7780 (0.0315), but our SCL with transfer learning (none) and SCL with fine-tuning (none) achieved 0.6411 (0.1170) and 0.7346 (0.0681), respectively. However, it is worth noting that our proposed SCL with transfer learning and augmentations (jittering and flipping) achieved 0.8120 (0.0227) (see Table V). In the case of the 2D-CNN encoders, our SCL, either with transfer learning or fine-tuning, outperformed supervised learning in all cases (see Table VI).
4) Supervised learning versus SCL (with augmentations): Comparing all augmentation combinations for each encoder, our SCL registered outstanding performance compared with the supervised learning except in two cases: Dataset 6 of the 1D-VGG-8 with scaling, jittering, and flipping augmentations (see Table V) and Dataset 4 of the 2D-ResNet-34 with jittering and flipping augmentations (see Table VI).

We performed the computational complexity analysis and convergence analysis to use our method. To ensure fair comparisons across various training strategies and augmentation types, we focused on Dataset 5 and used the 1D-VGG-8 encoder architecture, demonstrating superior performance. For the computational complexity analysis, we measured the execution time (in seconds) for fine-tuning under each training strategy and augmentation type in Table VIII.

Our findings indicate that training strategies involving augmentations required a higher time complexity than cases without augmentation. Furthermore, time complexity increased proportionally with the number of augmentation types employed. Notably, our SCL with fine-tuning learning exhibited early convergence in all cases. In the transfer learning stage, since the trainable network was only the one-layer neural network, it showed the highest time complexity but yielded the best predictive performances (see Table VIII).

TABLE VIII
COMPUTATIONAL COMPLEXITY ANALYSIS FOR THE 1D-VGG-8 ENCODER OF DATASET 5

Augmentation	Supervised	SCL + transfer learning	SCL + fine-tuning
None	151.87 (29.78)	262.73 (210.56)	72.90 (5.55)
Flipping	190.09 (21.07)	264.83 (291.08)	105.26 (7.46)
Jittering	447.13 (68.33)	650.85 (455.90)	238.91 (62.70)
Scaling	222.22 (50.21)	416.71 (219.53)	120.56 (20.96)
Jittering and Flipping	443.35 (70.24)	771.77 (626.63)	263.93 (53.54)
Jittering and Scaling	415.73 (70.70)	644.65 (320.38)	208.72 (10.42)
Scaling and Flipping	268.36 (60.61)	282.03 (139.45)	138.21 (25.10)
Scaling and Jittering and Flipping	407.02 (32.97)	789.79 (411.49)	246.71 (41.70)

Average training time (seconds) in fivefold cross validations. Standard deviations are presented in parentheses.

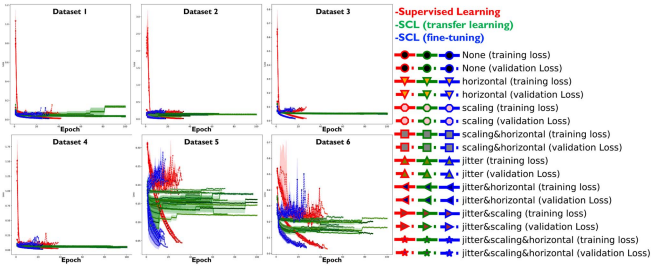


Fig. 5. Illustration of learning curves for all datasets with best encoder structures. Training (solid) and validation (dashed) losses of each augmentation type on the supervised learning (red), SCL with transfer learning using a frozen encoder (green), and SCL with fine-tuning using a trainable encoder (blue) are plotted. The solid lines and shaded regions indicate the average values and standard deviations of each epoch obtained in the fivefold cross validation. Note that we used an early stopping method and selected the model of the best epoch that showed the minimum validation loss. Consequently, the presence of short lines indicates the early stopped experiments.

For the convergence analysis, we illustrated the learning curves for each training strategy and augmentation type in Fig. 5. SCL exhibited lower initial losses than supervised learning. Particularly, the SCL with transfer learning involving a frozen pretrained encoder showed the smallest gap between training and validation losses. These results strongly suggest that SCL-based pretrained encoders effectively capture the class-distinguishable features and facilitate efficient convergence during the training process.

In this case study, by using multisensor signals combined with the machine learning method, we gained insights into the process and equipment conditions, and we validated their effectiveness in detecting early alarms of automobile engine manufacturing. We applied our SCL to leverage the label information for extracting alarm-related early features of multisensor signals and observed the effectiveness of the SCL and time-series augmentations. Although related research in this area has not yet been extensively studied, we highlight that our study is the first attempt to utilize multisensor signals for machine health monitoring in automobile engine manufacturing. In addition, note that this is the first attempt to investigate the power of supervised learning with time-series augmentations using multisensor signals of manufacturing systems.

We can conclude that our method has robust performance even in multilabel classification problems, and we have demonstrated its applicability for early alarm detection. We believe that early alarms detected from our method will serve as a valuable guide for predictive maintenance.

VI. Conclusion

We introduced a useful approach to the classification of multisensor signals through the lens of SCL, leveraging the principles of time-series augmentations and contrastive learning. Our method demonstrated improvements in prediction performances and showed effectiveness in identifying early features of alarms in automobile engine manufacturing. It is evident that the present study not only contributes to the methodological advancements in the classification of time-series data but also holds practical relevance for real-world applications.

Our method can be used to improve the performance of many problems in manufacturing industries, such as fault detection and diagnosis, machine health monitoring, and process quality prediction. In particular, the successful application of our method in early prediction problems will play a crucial role in predictive maintenance across various manufacturing sectors. Industries can proactively schedule maintenance, prevent breakdown, and optimize production processes by early predicting potential faults or anomalies. This will contribute to cost reduction and efficiency improvement.

There is room for improvement in our approach. In this study, for the multilabeled classification, our SCL assigned *positive* samples if they had exact multiple labels that matched the *anchor*'s labels within a batch. However, if samples have at least one or more of the matching labels, there is a possibility that they have similar signal patterns to the *anchor*. In addition, we trained and evaluated our method on training and testing data obtained in the same period. However, the status and signal patterns of manufacturing equipment could vary because of equipment failures and maintenance in real-world industries.

These points raise several future research suggestions for successful real-industrial applications. In SupConLoss, considering class neighbors of the *anchor*, which are similar multilabel outputs, are required if the sensor signal patterns are determined by the types of alarms. One possible direction would be assigning weights proportional to the number of the same labels to the similarities between the *anchor* and *positive* or *negative* samples. Furthermore, the prediction performances should also be evaluated with the training and testing data acquired in past and future periods. For example, experiments using data obtained in January, February, and March as training, validation, and testing data, respectively, would provide a more comprehensive assessment and warrant the successful application of our method.

In conclusion, we believe that our method will stand as a crucial step forward in effectively utilizing labeled data for improved predictive modeling in the domain of multisensor signal analysis in various manufacturing industries.

References

[1] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time series classification with multivariate convolutional neural network," *IEEE Trans. Ind. Electron.*, vol. 66, no. 6, pp. 4788–4797, Jun. 2019, doi: 10.1109/TIE.2018.2864702.

[2] J. J. M. Jimenez, S. Schwartz, R. Vingerhoeds, B. Grabot, and M. Salaün, "Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics," *J. Manuf. Syst.*, vol. 56, pp. 539–557, Jul. 2020, doi: 10.1016/j.jmsy.2020.07.008.

[3] H. Qiao, T. Wang, P. Wang, S. Qiao, and L. Zhang, "A time-distributed spatiotemporal feature learning method for machine health monitoring with multi-sensor time series," *Sensors*, vol. 18, no. 9, Sep. 2018, Art. no. 2932, doi: 10.3390/s18092932.

[4] A. Widodo and B. S. Yang, "Application of nonlinear feature extraction and support vector machines for fault diagnosis of induction motors," *Expert Syst. Appl.*, vol. 33, no. 1, pp. 241–250, Jul. 2007, doi: 10.1016/j.eswa.2006.04.020.

[5] M. S. Safizadeh and S. K. Latifi, "Using multi-sensor data fusion for vibration fault diagnosis of rolling element bearings by accelerometer and load cell," *Inf. Fusion*, vol. 18, pp. 1–8, 2014.

[6] M. Van and H.-J. Kang, "Wavelet kernel local fisher discriminant analysis with particle swarm optimization algorithm for bearing defect classification," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 12, pp. 3588–3600, Dec. 2015, doi: 10.1109/TIM.2015.2450352.

[7] M. Misra, H. H. Yue, S. J. Qin, and C. Ling, "Multivariate process monitoring and fault diagnosis by multi-scale PCA," 2002. [Online]. Available: www.elsevier.com/locate/compchemeng

[8] R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang, "Machine health monitoring using local feature-based gated recurrent unit networks," *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1539–1548, Feb. 2018, doi: 10.1109/TIE.2017.2733438.

[9] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mech. Syst. Signal Process.*, vol. 115, pp. 213–237, 2019, doi: 10.1016/j.ymssp.2018.05.050.

[10] A. Essien and C. Giannetti, "A deep learning model for smart manufacturing using convolutional LSTM neural network autoencoders," *IEEE Trans. Ind. Inform.*, vol. 16, no. 9, pp. 6069–6078, Sep. 2020.

[11] M. Zhao, S. Zhong, X. Fu, B. Tang, and M. Pecht, "Deep residual shrinkage networks for fault diagnosis," *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4681–4690, Jul. 2020.

[12] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Trans. Ind. Inform.*, vol. 17, no. 8, pp. 5790–5798, Aug. 2021.

[13] R. Liu, G. Meng, B. Yang, C. Sun, and X. Chen, "Dislocated time series convolutional neural architecture: An intelligent fault diagnosis approach for electric machine," *IEEE Trans. Ind. Inform.*, vol. 13, no. 3, pp. 1310–1320, Jun. 2017, doi: 10.1109/TII.2016.2645238.

[14] T. Xie, X. Huang, and S.-K. Choi, "Intelligent mechanical fault diagnosis using multisensor fusion and convolution neural network," *IEEE Trans. Ind. Inform.*, vol. 18, no. 5, pp. 3213–3223, May 2022, doi: 10.1109/TII.2021.3102017.

[15] B. Luo, H. Wang, H. Liu, B. Li, and F. Peng, "Early fault detection of machine tools based on deep learning and dynamic identification," *IEEE Trans. Ind. Electron.*, vol. 66, no. 1, pp. 509–518, Jan. 2019, doi: 10.1109/TIE.2018.2807414.

[16] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta, "An unseen fault classification approach for smart appliances using ongoing multivariate time series," *IEEE Trans. Ind. Inform.*, vol. 17, no. 6, pp. 3731–3738, Jun. 2021, doi: 10.1109/TII.2020.3016590.

[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1575–1585.

[18] P. Khosla et al., "Supervised contrastive learning," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 18661–18673, 2020.

[19] A. Zhao, J. Dong, and H. Zhou, "Self-supervised learning from multi-sensor data for sleep recognition," *IEEE Access*, vol. 8, pp. 93907–93921, 2020, doi: 10.1109/ACCESS.2020.2994593.

[20] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLoS One*, vol. 16, Jul. 2021, Art. no. e0254841, doi: 10.1371/journal.pone.0254841.

[21] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," *IEEE Trans. Semicond. Manuf.*, vol. 30, no. 2, pp. 135–142, May 2017.

[22] Y. Yao, S. Zhang, S. Yang, and G. Gui, "Learning attention representation with a multi-scale CNN for gear fault diagnosis under different working conditions," *Sensors*, vol. 20, no. 4, 2020, Art. no. 1233.
[23] L. Jing, T. Wang, M. Zhao, and P. Wang, "An adaptive multi-sensor data fusion method based on deep convolutional neural networks for fault diagnosis of planetary gearbox," *Sensors*, vol. 17, no. 2, 2017, Art. no. 414.
[24] B. Chen, Y. Liu, C. Zhang, and Z. Wang, "Time series data for equipment reliability analysis with deep learning," *IEEE Access*, vol. 8, pp. 105484–105493, 2020, doi: 10.1109/ACCESS.2020.3000006.
[25] H. G. Enad and M. A. Mohammed, "A review on artificial intelligence and quantum machine learning for heart disease diagnosis: Current techniques, challenges and issues, recent developments, and future directions," *Fusion, Pract. Appl.*, vol. 11, no. 1, pp. 8–25, 2023, doi: 10.54216/FPA.110101.
[26] M. A. Khan, "An IoT framework for heart disease prediction based on MDCNN classifier," *IEEE Access*, vol. 8, pp. 34717–34727, 2020, doi: 10.1109/ACCESS.2020.2974687.
[27] F. Ali et al., "A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion," *Inf. Fusion*, vol. 63, pp. 208–222, Nov. 2020, doi: 10.1016/j.inffus.2020.06.008.
[28] Z. Chen and W. Li, "Multisensor feature fusion for bearing fault diagnosis using sparse autoencoder and deep belief network," *IEEE Trans. Instrum. Meas.*, vol. 66, no. 7, pp. 1693–1702, Jul. 2017.
[29] B. Zhang, Y. Mao, X. Chen, Y. Chai, and Z. Yang, "Self-supervised learning advance fault diagnosis of rotating machinery," in *Proc. Int. Conf. Neural Comput. Adv. Appl.*, 2021, pp. 319–332.
[30] T. Von Hahn and C. K. Mechefske, "Self-supervised learning for tool wear monitoring with a disentangled-variational-autoencoder," *Int. J. Hydromechatronics*, vol. 4, no. 1, pp. 69–98, 2021.
[31] J. S. L. Senanayaka, H. Van Khang, and K. G. Robbersmyr, "Toward self-supervised feature learning for online diagnosis of multiple faults in electric powertrains," *IEEE Trans. Ind. Inform.*, vol. 17, no. 6, pp. 3772–3781, Jun. 2021, doi: 10.1109/TII.2020.3014422.
[32] W. Zhang, D. Chen, and Y. Kong, "Self-supervised joint learning fault diagnosis method based on three-channel vibration images," *Sensors*, vol. 21, no. 14, Art. no. 4774, 2021, doi: 10.3390/s21144774.
[33] B. Gunel, J. Du, A. Conneau, and V. Stoyanov, "Supervised contrastive learning for pre-trained language model fine-tuning," 2020, *arXiv:2011.01403*.
[34] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.
[35] A. Bagnall et al., "The UEA multivariate time series classification archive," 2018, *arXiv:1811.00075*.
[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations, Conf. Track Proc.*, 2015, pp. 1–14.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

Yoon Sang Cho received the B.E. degree in industrial and management engineering from the Hankuk University of Foreign Studies and the Ph.D. degree in the same field from Korea University, Seoul, South Korea, in 2017 and 2022, respectively.

He is currently a Postdoctoral Fellow with the Division of Biostatistics, Department of Population Health, New York University, New York, NY, USA. His research focuses on artificial intelligence reliability and its applications in real-world industrial contexts.

Seoung Bum Kim received the M.S. degree in industrial and systems engineering in 2001 and the Ph.D. degree in industrial and systems engineering in 2005 from the Georgia Institute of Technology, Atlanta, GA, USA.

From 2005 to 2009, he was an Assistant Professor with the Department of Industrial and Manufacturing Systems Engineering, University of Texas at Arlington. He is a Professor with the School of Industrial and Management Engineering, Korea University, Seoul, South Korea. He serves as the Director of Artificial Intelligence Engineering Center, Korea University. He has authored or coauthored more than 180 internationally recognized journals and refereed conference proceedings. He is actively involved with the INFORMS, serving as the President for the INFORMS Section on Data Mining. His research interests utilize machine learning algorithms to create new methods for various problems appearing in engineering and science.