IEEE Open Journal of the
Industrial Electronics Society

Received 28 August 2024; accepted 3 September 2024. Date of publication 6 September 2024;
date of current version 19 September 2024. The review of this article was arranged by Associate Editor Luis Ribeiro.

Digital Object Identifier 10.1109/OJIES.2024.3455264

Unsupervised and Semisupervised Machine Learning Frameworks for Multiclass Tool Wear Recognition

MARYAM ASSAFO [1] AND PETER LANGENDOERFER [1,2]

[1]Chair of Wireless Systems, BTU Cottbus-Senftenberg, 03046 Cottbus, Germany
[2]IHP - Leibniz-Institut für Innovative Mikroelektronik, 15236 Frankfurt (Oder), Germany

CORRESPONDING AUTHOR: MARYAM ASSAFO (e-mail: maryam.assafo@b-tu.de).

This work was supported by the Federal Ministry of Education and Research of Germany (BMBF) within the iCampus Cottbus project, under Grant 16ES1128K.

ABSTRACT Tool condition monitoring (TCM) is crucial to ensure good quality products and avoid downtime. Machine learning has proven to be vital for TCM. However, existing works are predominately based on supervised learning, which hinders their applicability in real-world manufacturing settings, where data labeling is cumbersome and costly with in-service machines. Additionally, the existing unsupervised solutions mostly handle binary decision-based TCM which is unable to fully reflect the dynamics of tool wear progression. To address these issues, we propose different unsupervised and semisupervised five-class tool wear recognition frameworks to handle fully unlabeled and partially labeled data, respectively. The underlying methods include Laplacian score, sparse autoencoder (SAE), stacked SAE (SSAE), self-organizing map, Softmax, support vector machine, and random forest. For the semisupervised frameworks, we considered designs where labeled data influence only feature learning, classifier building, or both. We also investigated different training configurations of SSAE regarding the supervision level. We applied the frameworks on two run-to-failure datasets of milling tools, recorded using a microphone and an accelerometer. Single sensor and multisensor data under different percentages of labeled training data were considered in the evaluation. The results showed which of the frameworks led to the best predictive performance under which data settings, and highlighted the significance of sensor fusion and discriminative feature representations in combating the unavailability and scarcity of labels, among other findings. The highest macro-F1 achieved for the two datasets with fully unlabeled data reached 87.52% and 75.80%, respectively, and over 90% when only 25% of the training observations were labeled.

INDEX TERMS Autoencoder, feature learning, Laplacian score (LS), machine learning (ML), multiclass classification, predictive maintenance (PdM), semisupervised learning, self-organizing map, tool condition monitoring (TCM), unsupervised learning.

I. INTRODUCTION

Cutting tools are key components in the manufacturing industry, and their health condition plays a significant role in the quality of finished products as well as the availability of production lines. However, tool wear is inevitable during machining due to the thermal-force coupling effect, necessitating a suitable maintenance plan [1]. To this end, three maintenance strategies exist, namely, corrective maintenance, preventive maintenance, and predictive maintenance (PdM) [1], [2]. With corrective maintenance, tool replacement occurs only after failure, causing unplanned downtime, workpiece damage, etc. On the other hand, preventive maintenance aims at avoiding failure by replacing the tool based on a predefined time schedule, leading to an inefficient resource utilization and financial losses. As a tradeoff solution, PdM is based on continuously monitoring the tool condition and replacing it in a timely manner, preventing failure while maximizing the tool usage and ensuring good quality products. Implementing PdM mainly involves deploying sensors to collect data relevant to the tool state. This data are subsequently

used to develop the monitoring models which will eventually be deployed for online tool condition monitoring (TCM) [1], [2], [3].

Machine learning (ML) has been extensively employed to build data-driven TCM models, with the supervised learning being predominately used. Despite the state-of-the-art performance of many existing supervised solutions, e.g., [4], their heavy reliance on labeled training data limits their applicability in real-world manufacturing scenarios where labeled data are hard to obtain. This is particularly the case with the supervised deep learning which needs a massive amount of labeled data to ensure a high predictive performance [5]. Data labeling is a laborious and costly task, and often needs experts. Moreover, considering the TCM task, labeling the sensor data collected during machining along the tool lifetime requires frequent interruptions of the machining process so that the tool health state can be inspected and the corresponding data are labeled accordingly. This inspection is usually carried out by an expert with the assistance of special equipment, e.g., microscope, cameras, etc., that captures some physical quantities and/or qualities related to the tool health condition, e.g., flank tool wear [6], surface roughness of workpiece [7], characteristics of the resulting chips [8], etc. While data labeling might not be extremely challenging in laboratory settings, it is tremendously costly and cumbersome in real-world settings with in-service machines on the shop floor. These interruptions adversely affect the productivity of production lines, among other consequences. Therefore, the supervised TCM solutions are not quite appealing to manufacturers in terms of cost effectiveness and seamlessness of implementation. In addition, given the massive amount of unlabeled data facing the industry sectors [9], relying only on supervised solutions not only hinders the adoption of PdM in real-world industry applications but also prevents reaching the potential of the available unlabeled data. The aforementioned constraints pose an urgent need for fully unsupervised solutions that can develop monitoring models using unlabeled data. Nevertheless, such solutions are receiving little interest in TCM literature.

Concerning the existing fully unsupervised TCM approaches, they mostly deal with the task of anomaly detection modeled as a one-class classification, e.g., in [1], [10], [11], and [12], where a model is trained using only normal data and the tool is then assessed in the online phase as normal or abnormal. Unlike such classification problems with a binary decision, a multiclass classification enables the recognition of the multiple wear states evolved along the tool lifetime, allowing a more granular and timely TCM, which in turn helps to reach a better maintenance plan and enhances the real-time response by not only detecting the failure but also anticipating it through the identification of the different degradation states leading to it. However, to the best of our knowledge, fully unsupervised multiclass classification for tool state recognition has been scarcely studied in literature, and the existing insights are quite limited especially with respect to feature learning, e.g., [13], [14]. Compared to the supervised learning, one major challenge of using the unsupervised learning is that it usually leads to a lower predictive performance [15]. This is to be expected theoretically since critical processes of the supervised ML pipeline, such as feature learning and model building, can be considerably driven by the labels. As such, the lack of studies in this regard will hinder the growth of this challenging research area.

Some existing TCM frameworks include both supervised and unsupervised methods, constituting semisupervised frameworks. Unsupervised methods are usually employed for various purposes, e.g., dimensionality reduction [6]. However, the input data considered for these frameworks are fully supervised data, i.e., when it comes to applying the supervised methods, all the training samples along with their labels are used, which does not meet the real-world scenarios where labeled data, if available, are usually scarce.

In light of the aforementioned discussion, the main limitations of the existing works and the research gaps that we address in this article can be summarized as follows.

1) Existing literature on supervised TCM solutions is increasingly growing, whereas fully unsupervised solutions are receiving little interest despite being urgently needed in real-world industry settings.
2) The existing fully unsupervised solutions mostly handle the anomaly detection task, and thus, are confined to a binary decision. On the other hand, fully unsupervised multiclass classification works are limited.
3) Fully labeled data were considered in many existing semisupervised frameworks, which does not reveal their performance with scarce labeled data.

The main aim of this article is developing multiclass tool wear recognition frameworks that can deal with different levels of data constraints with respect to the amount of labels available, more specifically, fully unlabeled data and partially labeled data. We focus on feature learning and classifier building as they are two integral tasks affecting the performance of TCM. Feature learning, which is a subterm of the broader term "feature engineering," refers to the automated algorithms aiming at feature selection (FS) or feature construction (transformation) [16]. Both latter methods were considered and used to provide a useful, lower dimensional feature representation that can be subsequently used in building ML models. The proposed frameworks were experimentally evaluated for a five-class tool wear recognition in milling, using two run-to-failure milling tool datasets recorded by a microphone and an accelerometer under two different sets of operating conditions, i.e., rotation speed, feed rate, etc. Each dataset includes both up-milling and down-milling operations. The contributions of this article can be summarized as follows.

1) Proposing different fully unsupervised multiclass tool wear recognition frameworks that can handle fully unlabeled data. A self-organizing map (SOM) was employed for building the monitoring models, while different approaches for feature learning were considered and

compared, namely, Laplacian score (LS), shallow sparse autoencoder (SAE), and deep stacked SAE (SSAE). The impact of the number of selected features on the predictive performance was also investigated.
2) Proposing different semisupervised frameworks that can handle partially labeled data. Different degrees of supervision were induced across the different frameworks. The underlying methods include: unsupervised SSAE and semisupervised SSAE for feature learning, support vector machine (SVM), Softmax, random forest (RF) as supervised classifiers, and SOM as an unsupervised classifier. These frameworks were experimentally tested on different percentages of labeled training data.
3) Investigating different training configurations of SSAE with respect to the supervision degree involved during its deep feature learning, and analyzing their impact on the multiclass tool wear recognition. To the best of our knowledge, this performance aspect was neglected in the related works that used SSAE, e.g., [17], [18].
4) Single sensor data as well as multisensor data were considered in the evaluation.

We experimentally compared the different frameworks, showing their relative strengths/weaknesses with respect to the task at hand, and which framework led to the best predictive performance under which data settings. Further, we considered various baselines to assess their utility and robustness (e.g., performance with fully labeled data, without feature learning, etc.). The results highlighted the significance of sensor fusion and discriminative feature spaces in improving multiclass TCM under label constraints. In addition, they showed two distinctive ways of reaching discriminative feature representations: 1) implicitly by solely relying on some data properties without assistance of any labels (the case of LS under fully unsupervised settings); 2) explicitly by using labels, even if few, to fine-tune nondiscriminative feature spaces pretrained on unlabeled data (the case of SSAE under semisupervised settings), among other new findings that can aid researchers to further advance this significant yet overlooked research area.

The rest of this article is organized as follows. In Section II, the related works are reviewed and their limitations are highlighted. Section III presents an overall methodology of developing TCM models. In Section IV, the proposed unsupervised and semisupervised frameworks are detailed. The motivation behind the methods adopted and some backgrounds are also presented. Section V presents the implementation details, including the experimental datasets, the different processes, training parameters, the results and discussion of the performance evaluation. Finally, Section VI concludes this article.

II. RELATED WORKS

ML frameworks have been extensively employed for tool wear recognition. Based on the degree of supervision involved in feature learning and classifier building, we categorize the existing frameworks into three main categories: 1) fully supervised, 2) partially supervised or semisupervised, and 3) fully unsupervised. It is worth mentioning that the framework category presented here should be distinguished from the type of data available (e.g., the training dataset might be fully supervised yet the framework is semisupervised), and should also be distinguished from the category of the ML algorithm used to build the predictive model (e.g., the model might be trained in a supervised manner yet the framework is semisupervised, for example, due to an unsupervised feature learning being adopted in the framework), as it will be shown in more examples later.

Fully supervised frameworks, where both feature learning and classifier training are performed in a fully supervised fashion. These frameworks are widely implemented in literature, e.g., in [4] and [8], but their large dependence on labeled data hinders their industrial applicability. Hence, as mentioned previously, they are not the focus of this article.

Semisupervised frameworks, where data labels are used by at least one part of the ML pipeline. For example, when feature learning (or classifier building) is fully supervised while the classifier building (or feature learning) is fully unsupervised, e.g., in [6], [19], [20], [21], and [22]. Alternatively, the feature learning and/or classification task might be partially (un)supervised, e.g., in [17], [18], [23], and [24]. It should be noted that, even when the training data are fully labeled, unsupervised methods can still be incorporated into the framework, e.g., to benefit from the advantages of both supervised and unsupervised methods, making it a semisupervised framework.

In [6], feature learning was realized through principle component analysis (PCA) (unsupervised), while the tool condition recognition was performed by SVM (supervised). On the other hand, the framework in [20] used t-Test (supervised) as an FS and SOM (unsupervised) as a classifier. In [19], the labels were used to aid the FS, while different fully unsupervised classifiers were built and compared using clustering algorithms, namely, K-means, Gaussian mixture models (GMM), density-based spatial clustering, and the extended hierarchical version of the latter. Similarly, in [21], feature learning-related settings were selected based on the Pearson correlation coefficient between some health indicators and the actual wear values (supervised), while unsupervised classifiers based on k-means and Jenks natural breaks were employed.

Some recent semisupervised frameworks leverage the powerful capability of unsupervised deep learning architectures to learn deep representations that can be subsequently coupled with supervised classifiers. To this end, different deep architectures based on autoenocder (AE) variants were commonly utilized [17], [18], [22], [23], [24]. An SSAE and Softmax classifier were employed in [17] and [18] for four-class and five-class tool wear recognition problems, respectively. In both works, the SSAE network underwent a two-stage training procedure: an unsupervised pretraining followed by a supervised fine-tuning. In [23], a framework comprising a stacked denoising autoencoder, PCA, t-distributed random neighbor embedding (t-SNE), and a Softmax classifier was proposed for a four-class TCM. Similarly to [17] and [18], the deep network

training also involves unsupervised and supervised phases. In [24], a parallel unsupervised learning model was built using three independent SSAE trained on three feature domains, respectively. The learned representations were concatenated into a single vector that was further used to build a feature-fusion SSAE model. Both the parallel learning model and fusion model were then fine-tuned using the data labels and a modified loss function. In [25], unsupervised SAE was also part of the feature learning performed and Softmax was used for classification. In [22], a four-class tool wear recognition was performed using an unsupervised variational autoencoder and a supervised extreme learning.

Despite the semisupervised nature of the aforementioned frameworks, the training data considered were fully labeled, i.e., when it comes to the supervised processes of the framework, all the training data were used along with their labels, which is not consistent with real-world scenarios where there is only a limited amount of labels, if any. Hence, the performance of these solutions with partially labeled data was not investigated.

Fully unsupervised frameworks, where both feature learning and classification tasks are performed in a fully unsupervised fashion, i.e., no labels are introduced at any stage of the pipeline. Most of the existing unsupervised TCM solutions deal with the anomaly detection task modeled as a one-class classification problem [1], [10], [11], [12]. Feature learning can be achieved here through unsupervised feature reduction methods (e.g., PCA), traditional FS (e.g., LS), task-customized FS (e.g., multicriteria decision making-based FS) [1], or deep feature learning (e.g., AE variants) [10], [11], [26]. A model representing the normal state is usually built using an unsupervised data-driven method, e.g., SOM [1], generative adversarial neural network [10], hybrid robust convolutional AE [11], long short-term memory (LSTM)-AE and GMM [26], and finally some health indicators and thresholds are derived accordingly. In the online phase, a binary decision as whether the tool is normal or abnormal is made, with the objective mostly being the detection of an excessive tool wear [1], [10], [11] or a tool breakage [12], [26]. However, given the gradual and dynamic process of tool wear progression, the binary decision-based solutions do not give sufficiently granular information about the exact tool condition.

Literature on the fully unsupervised multiclass TCM frameworks is scarce and the insights presented are limited. In [27], SAE was adaptively trained on the current segment comprising multiple raw sensor data, and the next segment was reconstructed using the trained model, and so on. The mean of reconstruction errors is sequentially taken. The filtered signal of the mean sequence was shown to be correlated with the tool wear, and upon some imposed thresholds, suitable for a four-class tool wear classification. However, standard classification performance metrics, e.g., F1-score, were not presented to clearly quantify the performance. In addition, the approach included many parameters affecting the performance, e.g., filtering settings, thresholds, sampling interval, etc. Moreover, the use of multiple raw sensor data poses a great burden in terms of computation and storage, and can be largely influenced by the interference and noise. In [13], a 6-D feature space was created by applying PCA on Fourier-transformed data, and K-means was used to cluster the resultant space into seven classes representing the normal state, and six different faults of the tool. However, the experiments presented did not involve any comparison with other feature learning or classification algorithms, limiting the insights that can be gained from the study. Moreover, the six faulty classes considered represent separate tool faults, e.g., notch wear, crater face, etc., rather than the different consecutive degradation stages experienced by the tool along its lifetime. In [14], three time-domain features were extracted from the cutting force signal, and then clustered using an algorithm based on adjacent grids searching. Based on the observation that the distribution of force features is correlated with the tool wear, a cluster density factor was defined to reflect the predicted tool wears, which were then converted into four classes. However, this approach is quite related to the force signal characteristics, and its effectiveness with other sensors is not clear. In addition, the role of feature learning in the performance was not investigated.

III. METHODOLOGY OF TCM MODEL DEVELOPMENT

As mentioned previously, this work mainly aims to develop unsupervised and semisupervised frameworks for multiclass tool wear recognition, with the focus being on feature learning and classifier building. Before delving into the frameworks, we will present the workflow followed in this article to develop TCM models. It is worth mentioning that the focus here is on the main steps and processes. Other details related to the experimental implementation and evaluation will be thoroughly covered in Section V. Fig. 1 illustrates the overall workflow in the offline phase, starting with training data ‘ collected during run-to-failure experiments of milling tools, and ending with a classification model. The highlighted area in this figure marks where along the process chain our proposed frameworks lie. The main steps are as follows.

Acquiring Raw Sensor Data: Sensor signals representing the sound and vibration generated during milling were collected using a microphone and an accelerometer, respectively. This data cover five different consecutive states along the tool wear progression, starting from when the tool is completely new up to its failure state.

Signal Preprocessing: It includes processes aiming at improving the data quality and preparing it for the subsequent analysis, such as segmenting, reducing noise, keeping only the relevant parts of collected signals, i.e., here the parts corresponding to active milling.

Feature Extraction: It aims to extract meaningful information from raw data, which also reduces the computational, storage, and communication burden associated with the large number of samples contained in raw signals [28]. As in [8], we used the following methods to extract multidomain features from each signal segment: 1) Time-domain statistical analysis,

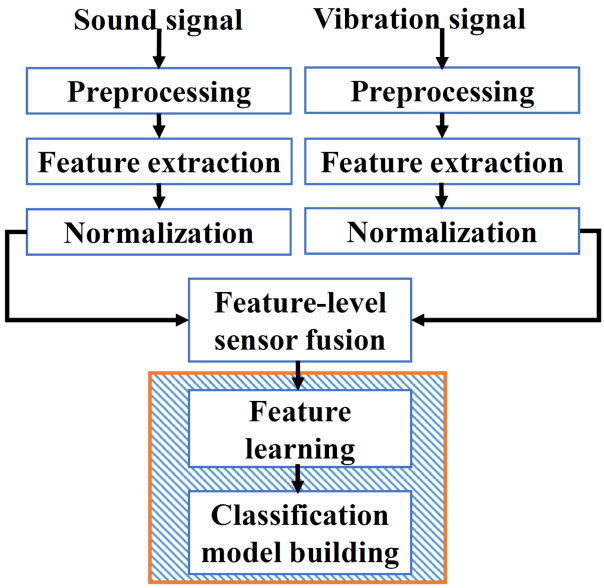

FIGURE 1. Methodology of developing TCM models in this article. The highlighted area indicates which steps our proposed frameworks cover.

2) nondecimated discrete wavelet transform, and 3) short-time Fourier transform (STFT). The specific features extracted using these methods are as follows [8].

1) *Eight Time-Domain Features:* Mean, variance, skewness, kurtosis, impulse factor, crest factor, root mean square, and range.
2) *56 Wavelet Features:* A six-level signal decomposition was performed using the following two mother wavelet functions: db1 and db3. Each decomposition generates one approximation and six details from which the four following features were extracted: 1) mean, 2) variance, 3) skewness, and 4) kurtosis. Thus, 28 features were extracted for each of these two decompositions.
3) *One STFT Feature:* mean peak frequency which is calculated by averaging the peak frequencies determined at the different time instances of the signal. The peak frequency is the frequency with the maximum power at the corresponding time instance.

Thus, a total of 65 features were extracted from each of the microphone and accelerometer signals.

Feature Normalization: To account for different ranges of different features, feature values were normalized with respect to the training data. We used z-score for this purpose, i.e., a feature value f_i is standardized by subtracting the mean, and then dividing by the standard deviation of the training samples corresponding to the feature f.

Feature-Level Sensor Fusion: To integrate the information from both sensors, the two 65-feature vectors corresponding to individual sensors are concatenated into a single vector, forming a 130-feature vector. This concatenation operation serves as a simple feature-level sensor fusion. It should be noted that, if only one sensor is to be used, this fusion step is skipped and the respective 65-feature vector proceeds to the next step.

Feature Learning: The purpose of feature learning in this article is not only to reach an informative feature representation of the tool wear progression but also to reduce dimensionality, which is crucial for realizing an effective and efficient TCM model as well as a better real-time response in the online phase. Hence, let M be the dimension of the original feature space containing the total extracted sensory features, then the output of feature learning is an N-dimension feature space, where $N < M$. As mentioned previously, FS and feature construction (transformation) methods are both considered in this study. FS selects a subset of N features from the full feature set, whereas feature transformation constructs new features by transforming the original M features into another N-dimensional feature space.

Classification Model Building: The N-dimensional feature samples generated from the previous step will be used by the classifier algorithm in building the diagnosis model for tool wear recognition.

Our frameworks do not only focus on the classifier building task but also on feature learning. While feature learning greatly influences the predictive performance of ML models regardless of their supervision type, its role is particularly significant under unsupervised settings due to the absence of labels which would otherwise guide the model building process. Since the learning algorithm will rely solely on the input features to detect hidden data patterns, we gave special care to devising and investigating different feature representations in our frameworks, aiming at revealing which feature representation properties and/or methods are promising with complex multiclass tool wear progression data where differences between distinct classes, i.e., five classes in our case, can be quite subtle. Further, under semisupervised settings, we considered designs that exhibit different levels of supervision over the two tasks, aiming at investigating different ways of harnessing both labeled and unlabeled data, while also considering different combinations of feature learning and classification methods.

IV. PROPOSED UNSUPERVISED AND SEMISUPERVISED FRAMEWORKS FOR MULTICLASS TOOL WEAR RECOGNITION

A. GENERAL OUTLINE

Different fully unsupervised and semisupervised frameworks for multiclass tool wear recognition are proposed in this article. With respect to the underlying methods, the former frameworks include only unsupervised methods, whereas the latter include both supervised and unsupervised methods. With respect to the input data, considering the real-world data constraints, we assume fully unlabeled data for the unsupervised frameworks, and partially labeled data for the semisupervised frameworks, i.e., the training dataset contains unlabeled observations and a certain fraction of labeled observations.

Fig. 2 shows an outline of the methods employed in the proposed frameworks. As can be seen from Fig. 2(a)

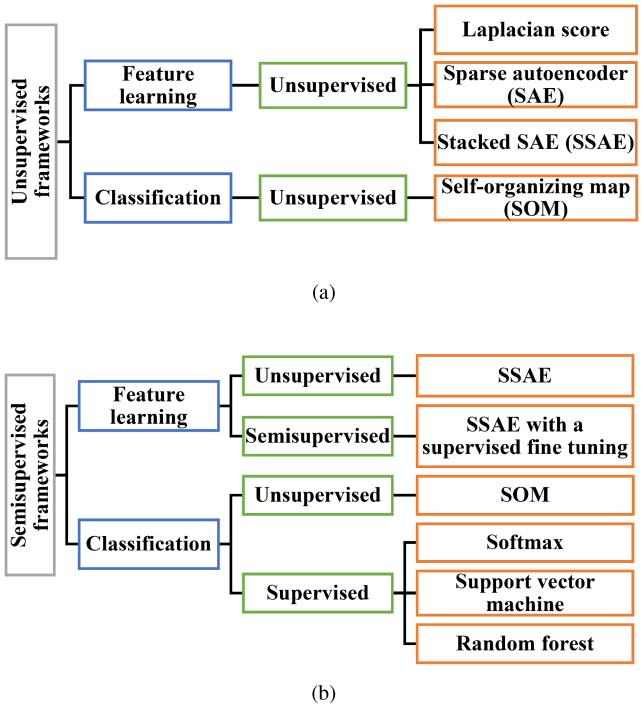

FIGURE 2. Outline of the methods employed in the proposed frameworks. (a) For unsupervised frameworks. (b) For semisupervised frameworks.

concerning the unsupervised frameworks, SOM is used for building the classification models. As for feature learning, two main approaches are employed: 1) LS-based FS, and 2) AE-based feature transformation/extraction, in which case SAE and SSAE are considered. Regarding the outline of the semisupervised frameworks shown in Fig. 2(b), two versions of SSAE are considered for deep feature learning, namely, SSAE, and a fine-tuned SSAE which involves a supervised fine-tuning phase following the typical unsupervised training of SSAE. As such, hereafter in this article, we will refer to the former version as an unsupervised SSAE, and to the latter version as a semisupervised SSAE. For the classification task, we use SOM as an unsupervised classifier, and the following supervised classifiers: Softmax, SVM, and RF. The details of the proposed frameworks are presented in Section IV-C and Section IV-D.

B. BACKGROUND AND MOTIVATION

In this section, we will present a brief background of the unsupervised methods employed in our work, as well as the motivation behind adopting them.

1) LAPLACIAN SCORE

LS is an unsupervised filter FS method that is essentially based on Laplacian Eigenmaps and locality preserving projection [29]. We utilize LS as an FS-based feature learning method for the following main reasons: 1) It is one of the most popular well-established unsupervised FS methods. 2) LS is independent of any ML algorithm as it evaluates the features based solely on the intrinsic data characteristics, leading to purely generic outputs that can be coupled with any ML algorithm. 3) It can yield a predictive performance similar to that of the supervised Fisher score method, as experimentally shown in [29].

The rationale of LS is that data points under the same class tend to be nearly spaced in the original data space. As such, given a certain feature, LS reflects the extent to which it preserves the local structure of the data. This structure is determined by constructing a weighted nearest neighbor graph with its nodes being the input data points. Two data points are connected by an edge if and only if one of them is among the k nearest neighbors of the other, and this connection is further weighted by a distance-based similarity function S_{ij}, where i and j are the two data points under consideration. The LS of feature f, L_f, is given as [29]

$$L_f = \frac{\sum_{ij} \left(x_{fi} - x_{fj}\right)^2 S_{ij}}{\text{var}(x_f)} \tag{1}$$

where x_{fi} and x_{fj} are the feature f's samples corresponding to the data points i and j, respectively. The denominator in (1) represents the sample variance of feature f. The smaller L_f, the more important the feature.

2) SPARSE AUTOENCODER

Autoencoder is an artificial neural network that is commonly used for nonlinear feature learning and dimensionality reduction. We employ it in our work for two main reasons: 1) it is the most popular unsupervised neural architecture for feature learning under the umbrella of deep learning [16]; 2) traditional unsupervised techniques, such as PCA, have been intensively studied with different TCM tasks. On the other hand, the AE performance with multiclass classification under data constraints is yet to be explored.

AE consists of an encoder and a decoder. The encoder transforms the input, $x \in R^M$, into a hidden (latent) feature representation $y \in R^N$; where M and N are the dimensionality of the input and encoded vectors, respectively. The encoding process is expressed as

$$y = f(x) = s_1\left(W^{(1)}x + b^{(1)}\right) \tag{2}$$

where $W^{(1)} \in R^{N\times M}$, $b^{(1)} \in R^N$, and s_1 are the weight matrix, bias vector, and activation function of the encoder, respectively. The encoded vector is subsequently mapped by the decoder into a reconstructed version of the original input vector r as follows:

$$r = g(y) = s_2\left(W^{(2)}y + b^{(2)}\right) \tag{3}$$

where $W^{(2)} \in R^{M\times N}$, $b^{(2)} \in R^M$, and s_2 are the weight matrix, bias vector, and activation function of the decoder, respectively [16].

W and b are the learning parameters of the encoder and decoder. The training process is unsupervised and aims to replicate the encoder's input at the decoder's output through

the optimization of a cost function measuring the error between the input and output, also referred to as a reconstruction error. The typical cost function used for the basic AE is the mean squared error (MSE), as follows:

$$\text{MSE} = \frac{1}{T}\sum_{i=1}^{T}\sum_{j=1}^{M}\left(r_{ij} - x_{ij}\right)^2 \tag{4}$$

where T is the number of training observations, and M is the number of input features.

Some mathematical properties can be introduced to the AE-learned features by adding certain regularization terms to the cost function [16]. One such property is the "sparsity" which can be imposed on the encoded feature vector by adding a penalty term R_{sparsity} to minimize the average activation of each of the hidden neurons. One common choice of this term is the Kullback–Leibler (KL) divergence function. L2 regularization R_{L2} is another penalty term that can be added to the cost function to constrain the weights, and thus, improving the generalizability and avoiding overfitting [30].

The overall cost function for training the SAE is given in

$$I = \text{MSE} + \lambda R_{\text{l2}} + \beta R_{\text{sparsity}} \tag{5}$$

where λ and β are coefficients for the L2 and sparsity regularization terms, respectively.

An SAE with a basic architecture comprising the input, hidden (encoding), and output layers is called a shallow SAE. A deeper architecture can be formed by stacking multiple SAEs successively, hence the name SSAE. In this case, the architecture is particularly characterized by the number of hidden layers. The input of the second SAE is the output of the encoding layer of the first SAE, and so on [16].

3) SELF-ORGANIZING MAP

SOM is an unsupervised neural network that has proven to be a powerful tool for data analysis, clustering, and visualization [31]. Moreover, it has many other advantages that are particularly desirable in practical applications, e.g., understandability, ease of optimization, low requirements of memory, computations, and training data, etc [1], [9]. SOM consists of a grid of neurons, usually a 2-D grid. Each neuron (a map unit) is characterized by its spatial position in the grid as well as its weight vector in the input space. SOM is capable of mapping high-dimensional input data into the neuron map in a way that preserves the topological relations in the input space, i.e., the closer the neurons in the grid, the more similar the patterns they represent in the input space, and vice versa. This special property, that distinguishes SOM from other clustering algorithms, such as K-means, is thanks to the fact that the spatial relations between different SOM neurons are taken into account when updating the neurons' weights during training. These relations are governed by the so-called neighboring function. The main principle of SOM training is based on finding the best matching unit (BMU), among the map neurons, for the input data sample. This is determined by the metric adopted to measure the similarity between the input samples and neurons in the input space. The weights of BMU as well as its spatial neighbors are updated so as to better match the input sample. In this article, we adopt the Euclidean distance as a similarity measure, as it is the most commonly used metric with SOM [31]. As such, the BMU is identified as follows:

$$c = \arg\min_{i}\{||x(t) - m_i(t)||\} \tag{6}$$

where c is the index of the BMU, $m_i(t)$ is the current weight vector of the neuron i, and $x(t)$ is the input vector. $||.||$ is the Euclidean distance.

We adopt the batch learning algorithm for training SOM, as it does not involve a learning parameter and it allows the network to reach a better and faster convergence compared to the sequential learning [31]. In batch learning, the training samples are fed to SOM all at once, and the weights are updated in each iteration as follows [1]:

$$m_i(t+1) = \frac{\sum_{j=1}^{T} h_{ci}(t)x_j}{\sum_{j=1}^{T} h_{ci}(t)} \tag{7}$$

where T is the number of training samples, and $h_{ci}(t)$ is the value of the neighborhood function, centered around the winning neuron c, for the neuron i when the input vector is x_j. This function takes higher values as the spatial distance between the neurons c and i gets smaller. After the completion of training, the neurons' weights act as prototypes representing the training data.

The map size is a significant parameter affecting the generalization and complexity of the SOM model. Too many neurons or too few neurons lead to overfitting or underfitting, respectively [9]. In addition, the computation and memory requirements increase with the map size. To yield a tradeoff, the following equations are commonly used to determine the size and dimensions of a 2-D map based on the training data [1]:

$$U \approx 5\sqrt{T} \tag{8}$$

$$\frac{u_1}{u_2} \approx \sqrt{\frac{e_1}{e_2}} \tag{9}$$

where U is the number of map neurons, u_1 and u_2 are the numbers of rows and columns of the map, respectively, e_1 and e_2 are the largest and second largest eigenvalues of the training data's covariance matrix, respectively.

C. PROPOSED UNSUPERVISED FRAMEWORKS

These frameworks use training observations without any labels. As mentioned previously, the different unsupervised frameworks proposed here differ in the feature learning task. The following approaches are considered: 1) An LS-based FS scheme. 2) SAE-based feature learning. 3) SSAE-based deep feature learning. The feature space resulting from the feature learning step is used to build an SOM model. Fig. 3 shows the workflow of the proposed unsupervised frameworks.

Fig. 3(a) illustrates the framework with LS. LS ranks the extracted sensory features according to the extent to which

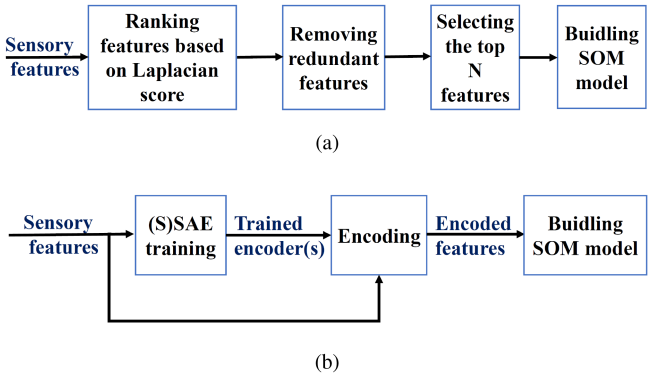

FIGURE 3. Workflow of the proposed unsupervised frameworks using self-organizing map and two feature learning approaches. (a) With LS. (b) With (stacked) sparse AE.

they preserve the local structure of the input feature space (see Section IV-B1). To obtain a feature subset, the top-N ranked features are selected. However, LS does not consider the interfeature correlation when scoring the features, i.e., highly correlated features might exist among the selected top-N features, leading to redundant, superfluous information that increases the dimensionality without necessarily enriching the representation. Moreover, redundant features might degrade the performance of the learning models. To circumvent this issue, eliminating redundant features can be carried out prior to selecting the top features [1], [8], [32]. We adopt the following iterative procedure presented in [32]: Starting from the top ranked feature, all the features with which they exhibit a strong correlation (equal or exceeding a predefined threshold value Th) are removed from the feature list. The same process continues iteratively down the remaining ranked feature list until no feature is remaining. Finally, the top-N features are selected from the remaining features set. As in [8], the interfeature correlation is measured in this article by the absolute value of Pearson's coefficient correlation (the values range from 0 to 1), and Th is set to 0.9.

The feature learning in the other frameworks is based on SAE. The number of hidden layers in the SAE network is a significant design parameter affecting the performance. An architecture with two hidden layers was a common choice in several PdM studies of multiclass recognition tasks, e.g., [17], [30], and [33]. In this article, we consider the following: 1) A shallow SAE (with only one hidden layer); 2) An SSAE with two hidden layers. Deeper architectures with over two hidden layers are not considered in this article since they might increase the complexity and training time without necessarily improving the accuracy, as experimentally shown in [33].

For the SAE-based feature learning, it is performed through the training process explained in Section IV-B2, where a mapping from the M-dimension original feature space into an N-dimension latent feature space is reached. It should be noted that, both the encoder and decoder are involved in the training process. However, when it comes to extracting the new features from the input, only the encoder part of the trained SAE is used to encode the input, as illustrated in Fig. 3(b).

Regarding the SSAE, it learns a multilevel feature representation through a greedy layerwise training, i.e., only a single SAE is trained at a time [16]. In our case with two hidden layers, the first SAE learns to reconstruct the original training observations, and then the second SAE learns to reconstruct the latent feature representation extracted by the first SAE's encoder.

In all the unsupervised frameworks, the feature vectors resulting from feature learning are passed to SOM. Recall that SOM is trained in an unsupervised fashion, with the resulting model comprising a set of prototypes representing the training data, and an input sample can then be mapped to one of these prototypes by finding its BMU. However, in order to use the SOM model for classification, given a finite set of possible classes, labeling SOM units should first take place, then a new data sample will be given the same class as that of its BMU [31]. Labeling SOM can be performed manually by domain experts, automatically, or both.

D. PROPOSED SEMISUPERVISED FRAMEWORKS

We propose different semisupervised frameworks that deal with partially labeled data. Generally speaking, the more a framework depends on labeled data for its underlying processes, the more likely for its performance to be affected by label scarcity. On the other hand, limiting the extent to which available labels are utilized might also limit the predictive performance of resulting models, since significant information can be carried by the labels no matter how few. In order to gain broad insights on different ways of harnessing partially labeled data, we designed the frameworks in such a way that, across different frameworks, the labeled data have different levels of influence over the functionality of feature learning and classifier building. In this regard, three scenarios are distinguished as follows.

1) The labeled data influence only feature learning. Here, the framework comprises a semisupervised SSAE (the supervision is induced here in the fine-tuning phase) and an unsupervised classifier.
2) The labeled data influence only the classifier building. Here, the framework comprises an unsupervised SSAE and a supervised classifier.
3) The labeled data influence both feature learning and classifier building. Here, the framework comprises a semisupervised SSAE and a supervised classifier.

As mentioned in Section IV-A, SOM is adopted as an unsupervised classifier, whereas SVM, RF, or Softmax was used in the frameworks involving a supervised classifier.

To facilitate the discussion, we use the following notations in the different steps described as follows. Let X be a set of training observations, consisting of unlabeled observations X_u, and labeled observations X_l. Let Y_l be the set of labels corresponding to the labeled observations X_l. Recall that a single observation here is an M-dimension feature vector consisting of the extracted sensory features.

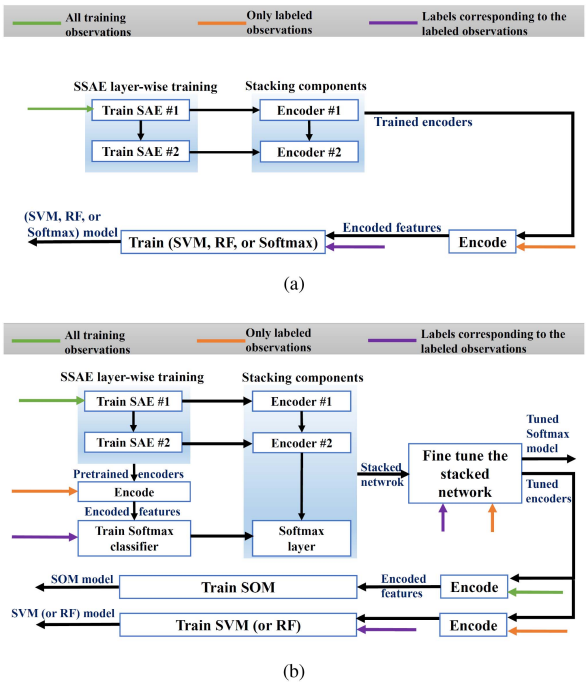

FIGURE 4. Workflow of the proposed semisupervised frameworks. (a) With unsupervised SSAE (undergoing only an unsupervised layerwise training) and a supervised classifer (SVM, RF, or Softmax). (b) With semisupervised SSAE (SSAE undergoing an unsupervised layerwise pretraining followed by a supervised global fine-tuning) and a supervised classifier (SVM, RF, or Softmax) or an unsupervised classifier (SOM).

Fig. 4(a) illustrates the main workflow of the semisupervised frameworks that use unsupervised SSAE and supervised classifiers. SSAE training is performed here through the typical greedy layerwise training, as explained previously for the unsupervised frameworks. For this training, the full training dataset is used, but without any labels, i.e., only X. Next, only the labeled observations X_l are encoded by the trained stacked encoders. Then, the encoded feature vectors are fed, together with their respective labels Y_l, to train the supervised classifier, i.e., SVM, RF, or Softmax. Thus, being fully unsupervised, the feature learning in these frameworks leverages the total training observations. However, the unlabeled observations X_u remained unexploited when it comes to building the supervised classifiers.

Fig. 4(b) illustrates the main workflow of the semisupervised frameworks that use semisupervised SSAE with (un)supervised classifiers. The training process of the semisupervised SSAE encompasses two successive phases, namely, an unsupervised pretraining and a supervised fine-tuning. The first unsupervised training uses all the training observations, i.e., X, and is based on the same greedy layerwise unsupervised training described previously. On the other hand, the fine-tuning phase aims to globally tune the previously initialized learning parameters in accordance with the supervised classification task at hand. However, SSAE does not inherently possess the capability of classifying data, as it only transforms the input into a latent representation. Thus, in order to enable the supervised fine-tuning phase, a classification layer should be added to the SSAE. This is commonly tackled by adding a Softmax classification layer, as in [17] and [18], which is also the solution employed in our frameworks. Yet, it should be noted that, in existing works [17], [18], all the training observations and labels were also used in the fine-tuning phase as fully supervised data were considered, and this added Softmax layer served also as the end classifier of the tool wear. On the other hand, since we consider partially labeled data in our frameworks, the fine-tuning phase uses only the labeled observations along with their labels, i.e., X_l and Y_l. Moreover, across our proposed frameworks that use semisupervised SSAE, one of the two following cases can be noticed with respect to the role of the added Softmax layer: 1) It is only involved in the feature learning task (more specifically, in the fine-tuning phase of SSAE), which is the case corresponding to the frameworks in which other classifiers were employed to classify the tool wear (i.e., SOM, SVM, or RF). 2) It serves as the end classifier of the tool wear, besides its involvement in the fine-tuning phase of SSAE.

In light of the discussion above, after the pretraining of SSAE, the initial training of the Softmax layer is carried out in a supervised fashion, with the input being the features generated from passing the labeled observations X_l through the pretrained SSAE encoders (encoding process), along with their corresponding labels Y_l. Next, the pretrained encoders of the first and second SAEs, as well as the pertained Softmax layer are successively stacked in one network. Using X_l and Y_l as input, the stacked network is then fine-tuned in a supervised manner, where the parameters of the whole stacked network are updated simultaneously based on a global supervised cost function, which is similar to training the multilayer perceptron.

After the fine-tuning, the stacked network can be used for feature extraction (the stacked fine-tuned encoders) and tool wear classification (the fine-tuned Softmax classification layer), which constitutes one of the proposed semisupervised frameworks. Moreover, in order to benefit from other state-of-the-art supervised classifiers, the role of Softmax layer can be limited to the fine-tuning phase of SSAE, i.e., the encoders which were fine-tuned with the assistance of the Softmax can now be coupled with other classifiers, which is the idea behind the proposed frameworks including SVM and RF. Alternatively, to leverage the full training observations in building the classification model, an unsupervised classifier can be trained on the features resulting from applying X to the fine-tuned stacked encoders, which corresponds to the proposed framework containing SOM. However, no labels are guiding the model building here. Among the other proposed semisupervised frameworks, the labeled data are involved the least in this latter framework containing SOM.

V. IMPLEMENTATION DETAILS

This section covers the experimental implementation of the proposed frameworks, including the datasets, training parameters, results and discussion, etc. As mentioned

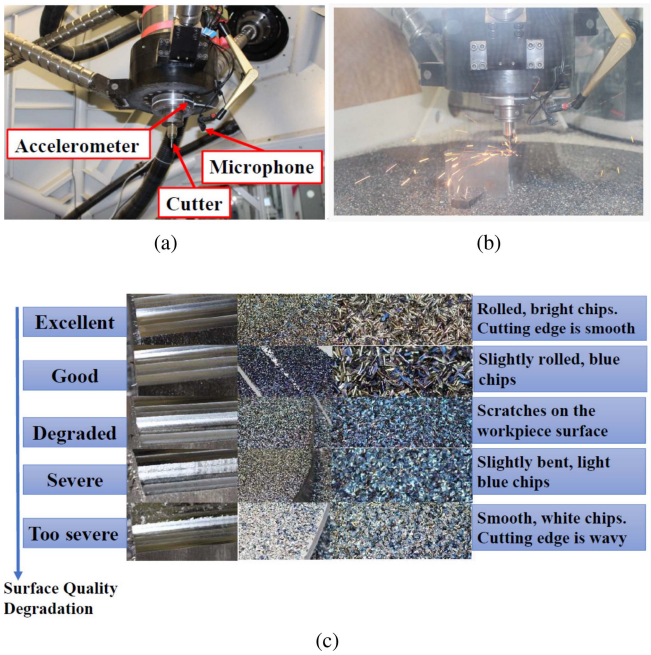

FIGURE 5. Experimental setup. (a) Deployed sensors. (b) Milling process. (c) Finished surface quality and chips generated under the five different health states (classes) of milling tools. [8].

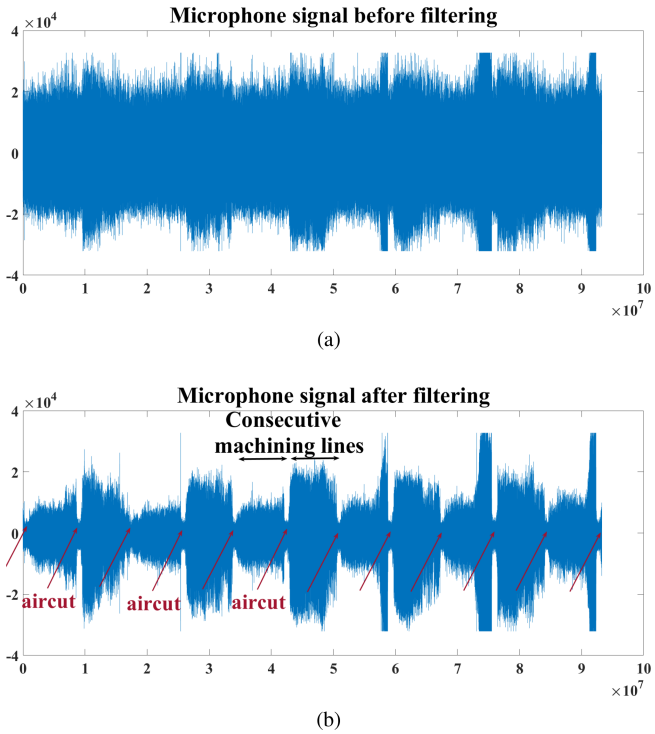

FIGURE 6. Example of microphone signals. (a) before filtering. (b) after Median filtering. [8].

previously, the specific PdM task handled by the proposed unsupervised and semisupervised frameworks is a five-class tool wear recognition in milling. It should be mentioned that all the software implementations related to the ML pipeline, starting from signal preprocessing, were performed in MATLAB R2020b.

A. MILLING DATASETS

Two run-to-failure datasets of milling tools were used in this article, with each tool having two teeth and a diameter of 8 mm. Each milling dataset represents sound and vibration signals recorded by a microphone and an accelerometer, respectively, during dry milling a steel workpiece. Fig. 5(a) shows the sensors deployed in the machine center. Each dataset comprises both up-milling and down-milling operations. The down-milling (or up-milling) refers to the milling process in which the cutter rotates against (or with) the feed direction. Alternating between these two operations, the milling process type changes with every cutting line of the workpiece surface. A snapshot of the milling process is shown in Fig. 5(b). Five different health states (classes) of the milling tools were used to label the collected sensor data, as shown in Fig. 5(c). The labeling was based on various factors, namely, the color and shape of the resulting chips, quality of the finished surface, wear on the cutting edge, and operator experience [8].

The sensor signals were passed through an amplifier into a Red Pitaya board, where they were sampled with a frequency of about 1.95 MHz, and then transferred to a PC for storage. As in [8], the following preprocessing were applied:

TABLE 1. Description of Milling Datasets [8]

Dataset (DS)	Rotation speed (rpm)	Feed rate (mm/min)	Depth of cut (mm)	Dataset size (per class 1/ ... /class C)
DS1	8000	360	2.5	12680 (2247/2599/2585/2639/2610)
DS2	5600	252	2.5	15665 (2865/3133/3181/3198/3288)

1) Median filtering to remove the high-frequency noise. Fig. 6 shows an example of the microphone signal before and after this step, showcasing the importance of filtering in revealing the different events encountered during the milling process. 2) Removing the parts of signals corresponding to the entry cut (the tool is engaging in the workpiece), exit cut (the tool is disengaging from the workpiece), and air cut (the tool is in the air), while keeping only parts corresponding to active milling. 3) Reducing the sampling frequency to about 400 KHz. The operating conditions of the two milling experiments as well as the size of the two datasets used in the subsequent analysis are listed in Table 1. More details about the datasets and the experimental setup can be found in [8] and [34], and about the preprocessing steps in [8].

B. TRAINING PARAMETERS

The distance metric used in SOM and LS is the Euclidean distance. Regarding LS, we set k [for the K nearest neighbors (KNN)] to be $k = ln(T)$, where T is the number of training observations. In our implementation of SOM, a 2-D hexagonal

grid was considered, where the map size and dimensions were determined based on the training data, as in (8) and (9), respectively. The batch learning algorithm was used for training (see Section IV-B3) with 300 epochs.

Concerning the SAE-based architectures, we used the Logistic sigmoid function as an activation function of the neurons. To yield a dimensionality reduction, we adopted an undercomplete configuration of the AE structure, i.e., the dimensionality of the encoding layer's output is smaller than that of the input layer, which will cause the network to learn a compressed representation of the input. When it comes to training, two sets of parameters were used. For the unsupervised training, we used an L2-weight-regularization coefficient of 0.001, a sparsity proportion of 0.05, a sparsity regularization of 1, a scaled conjugate gradient descent as a training algorithm, a loss function as in (5), and a maximum of 1000 epochs. For the supervised fine-tuning, we used a scaled conjugate gradient as a training algorithm, the cross entropy as a loss function, a maximum of 300 epochs. The training parameters of the Softmax layer are the same as those aforementioned for the fine-tuning of SSAE.

Regarding the dimensionality of the hidden (encoding) layer of SAE N, defined by the number of neurons in this layer, was set specific to each experimental case, as it will be shown in Section V-E2. The same holds true for the second hidden layer in the SSAE. As for the size of the first hidden layer in SSAE, N_{l1}, it was set as follows:

$$N_{l1} = \text{ceil}\left(M - \frac{M-N}{2}\right) \tag{10}$$

where M and N are the sizes of the input layer and the last hidden layer, respectively. Considering the datasets used here, M is 65 and 130 for the single-sensor feature vector and multisensor feature vector, respectively.

For SVM, we used a polynomial kernel of order 2, and the one-versus-one approach to construct the five-class SVM classifier. Thus, the SVM classifier comprises ten SVM binary classifiers ($C(C-1)/2$; where C is the number of classes). The penalty parameter was set to 1, and the kernel scale was determined automatically using a subsampling-based heuristic approach applied in MATLAB on the training data. As for the RF classifier, it consists of 50 classification trees, the minimal leaf size is 1, and the number of the randomly selected features for each decision split is the square root of the number of total input features, as in [8].

C. LABELING SOM FOR EVALUATION

As mentioned in Section IV-C, a built SOM model can be used for classification by labeling the map neurons, and then an input data sample is given the same class as that of its BMU. The existing approaches of labeling SOM are mainly based on associating each map unit i of the trained model with the class corresponding to the majority class of those training samples whose BMU is the unit i [9], [31]. Some considerations can also be taken to account for particular scenarios, e.g., a tie, neurons with zero or too few hits [31].

In this work, we present a scheme for labeling SOM which is largely inspired by [31]. In this scheme, we also adopt the majority class voting as a main approach to label the SOM units. However, we impose some rules to determine which of the training samples the voting is applied on. To this end, the main criterion we consider is the number of sample hits of map units. Given a neuron i, the number of sample hits H_i refers to the number of training samples whose BMU is the neuron i. If H_i is larger than a defined threshold th, then the majority voting is applied on all the training samples associated with the neuron i. Otherwise, the voting is applied on those training samples corresponding to the KNN of the neuron i. In any case, when a tie is encountered, KNN is also applied, as described above. If the tie was not resolved, K is increased incrementally until the tie is resolved. We set th to be the 0.5 quantile (median) of the sample hits of the total map neurons, and we set K to be $ln(T)$, where T is the number of training samples.

It should be emphasized here that training SOM was purely unsupervised, and this latter stage is only to label SOM for evaluation purposes. Thus, a test sample will be given the same class as that of its BMU, and this predicted class will then be compared with the ground-truth class of that sample for calculating the metric of the predictive performance.

D. PERFORMANCE METRICS AND VALIDATION

To evaluate the predictive performance of the TCM models built at the final stage of the different proposed frameworks, we use macro F1-score as a performance metric. It is calculated by taking the average of the individual F1-scores corresponding to the respective classes, as expressed in (14). Given a class c, the F1-score is given in (13), and it represents the harmonic mean of the recall (11) and precision (12) corresponding to this class [8].

$$\text{Recall}_c = \frac{\text{TP}}{\text{TP+FN}} \tag{11}$$

$$\text{Precision}_c = \frac{\text{TP}}{\text{TP+FP}} \tag{12}$$

$$(\text{F1-score})_c = \frac{2 \times \text{Recall}_c \times \text{Precision}_c}{\text{Recall}_c + \text{Precision}_c} \tag{13}$$

$$\text{macro-F1} = \frac{1}{C}\sum_{c=1}^{C}(\text{F1-score})_c \tag{14}$$

where TP and TN denote the number of the test observations that were correctly classified as positive and negative, respectively. FP and FN represent the number of the test observations that were wrongly classified as positive and negative, respectively. C is the number of classes, i.e., five classes in our work.

A five-fold cross validation was performed for all the experiments. As such, given a dataset, each of the resulting five-folds will serve once as test data while the remaining four folds form the training data. Hence, each macro-F1 reported in this article is the average of the individual macro-F1 values corresponding to the five test folds.

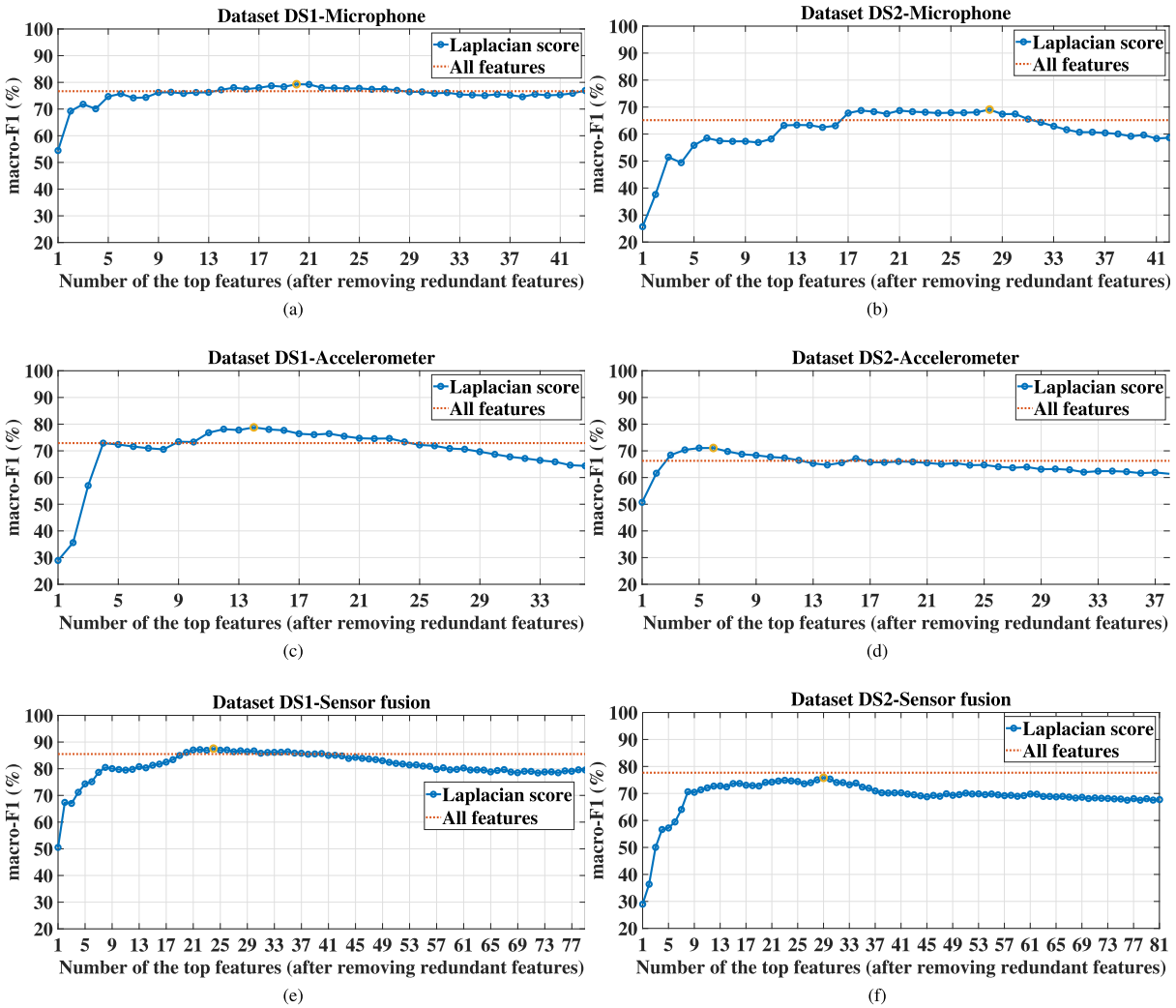

FIGURE 7. Macro-F1 variation of SOM over different numbers of features selected using the LS-based FS scheme (the highest value is marked in yellow), and the case where all the extracted features are used, for two milling datasets. (a) and (b) Microphone. (c) and (d) Accelerometer. (e) and (f) Sensor fusion.

E. RESULTS AND DISCUSSION OF UNSUPERVISED FRAMEWORKS

As mentioned previously, SOM is used with different feature learning methods across these frameworks.

1) EFFECT OF NUMBER OF SELECTED FEATURES

Starting with the LS-based FS scheme, we will evaluate the predictive performance of SOM over all the possible numbers of selected features, i.e., N in Fig. 3(a). For this purpose, N is increased incrementally one at a time, starting from $N = 1$, i.e., only the top-ranked feature, through the total number of features remaining after removing redundant features. For each resulting feature subset, an SOM model is built using the corresponding training observations, then tested on the test observations. Figs. 7(a)–(f) illustrate the macro-F1 variation over N for all the six combinations of dataset-sensor configurations. The highest macro-F1 of each curve was highlighted with a yellow circle. It can be seen that, as more features are added, macro-F1 initially tends to increase until a certain point, then it slightly decreases. The initial rising trend is a result of gaining more useful information by adding more relevant features. The slight degradation afterward can be attributed to the decreased distinction between different data patterns as more less-relevant features are added, especially, given the unsupervised training of SOM, there are no labels to guide the training and the model is built merely based on the data patterns observed in the feature space of the training data. Fig. 7 also shows the case where all the extracted features are used without applying FS, i.e., 65 features for the individual sensors and 130 features for the sensor fusion. This latter case helps to assess the utility of the LS-based FS scheme. As it can be seen from Figs. 7(a)–(f), different feature subsets, constituted out of different numbers of top features, achieved a higher macro-F1 at considerably lower dimensionalities compared to when all the features were used. This is particularly evident for the accelerometer, in which

TABLE 2. Feature Vector Dimensionality for the Subsequent Experiments

Milling dataset	Microphone	Accelerometer	Sensor fusion
DS1	20	14	24
DS2	28	6	29

case the FS scheme with 14 and 6 selected features for DS1 and DS2, respectively, led to a maximum macro-F1 improvement of 5.89% and 4.81% along with a reduction of 51 and 59 features, respectively, compared to the case of no FS. The only case where a better performance was obtained without FS was the sensor fusion of dataset DS2 [see Fig. 7(f)], where the macro-F1 achieved with all the 130 features was higher by 1.88% than the highest value achieved by the FS scheme (with 29 features). However, the performance difference in this case can be considered insignificant given the much higher number of features needed (101 more features). Therefore, given the overall results, the proposed LS-based FS scheme is suitable for building effective and efficient unsupervised multiclass tool wear recognition models.

2) COMPARISON OF LS AND (S)SAE

We will compare the different unsupervised frameworks that differ in feature learning. To ensure a fair comparison given a certain dataset-sensor, the dimensionality of the feature learning's output feature vectors is kept constant across all the frameworks, allowing the performance variation of SOM to be solely linked to the quality of feature vectors. As such, given a certain dataset-sensor, we set the unified dimensionality N of the learned feature vectors to be equal to the dimensionality that achieved the best performance using the LS selected features, i.e., based on the results shown in Fig. 7. These sizes are listed in Table 2 and will be used for all the subsequent experiments presented in this article regarding both the unsupervised and semisupervised frameworks.

Table 3 shows the macro-F1 results of SOM models with the different feature learning methods. Overall, SSAE slightly improved the performance compared to the SAE. The highest improvement reached 3.37%, which was achieved for the sensor fusion of DS2. However, for all the experimental cases, both SAE and SSAE led to a lower performance than that achieved with all the original features (the case of no feature learning). On the other hand, the macro-F1 attained with the LS-based FS scheme was substantially higher than those with SAE and SSAE. The average outperformance (over all the sensor configurations and milling datasets) reached 9.67% and 8.23%, respectively. Thus, despite the deep feature representation constructed by training the SSAE, it was inferior to the LS-based FS scheme when it comes to the unsupervised multiclass tool wear recognition. This can be attributed to the fact that the unsupervised SSAE learns to replicate the input, which might not necessarily yield a latent feature space that is sufficiently discriminative with respect to the different wear states. On the other hand, LS selects the features with the highest locality preserving power, implicitly favoring the most discriminative features despite the absence of labels.

Regarding the sensors, as to be expected, the sensor fusion led to a better performance compared to the individual sensors, for all the feature learning methods and datasets. This demonstrates the impact of sensor fusion in boosting the performance under the challenging fully unlabeled data.

F. RESULTS AND DISCUSSION OF SEMISUPERVISED FRAMEWORKS

For the semisupervised frameworks that deal with partially labeled data, the amount of labeled training observations is a critical factor affecting the performance. As such, we will evaluate the proposed semisupervised frameworks for the following percentages of labeled training observations (out of the total training observations): 100%, 50%, 25%, 10%, and 1%. The labeled training observations corresponding to each percentage were determined by downsampling the original training data accordingly. The case when 100% of the training observations are labeled represents data under no constraints regarding the labels (which is the case considered in the existing semisupervised frameworks reviewed in Section II). In contrast, the other percentages represent data under different levels of label constraints. Thus, the 100%-case can also serve here as a baseline for the other cases. As mentioned previously, the different proposed frameworks are influenced differently by the labeled data, based on how the labeled data involve in the feature learning and/or classifier building (See Fig. 4).

1) EFFECT OF DIFFERENT SUPERVISION CONFIGURATIONS ON SSAE FEATURES

Fig. 8 shows an illustrative example of the different neural network architectures involved in training the semisupervised SSAE, as well as the different input/output dimensionalities corresponding to this example case (DS1-sensor fusion). The input is a 130-feature vector, the dimensionality of the second hidden layer is 24 (as determined from Table 2), and the dimensionality of the first hidden layer is 77 [as determined from (10)].

We will study how different supervision configurations of training SSAE can influence the quality of resulting feature representations, and thus, SOM performance. Fig. 9 illustrates the different macro-F1 values achieved by SOM with the semisupervsied SSAE under different percentages of the labeled training observations involved in the fine-tuning of SSAE, as well as that achieved with the unsupervised SSAE which does not involve any supervision. This latter case serves as a significant reference to assess how the performance evolves with adding more labeled data. Recall that SOM always exploits the total training observations due to its unsupervised training, regardless of the framework, and that the influence of labeled data in the semisupervised framework containing SOM is only restricted to the semisupervised SSAE-based feature learning. Thus, the performance variation over the different cases in this figure is only attributed to the quality of the corresponding feature representation reached.

TABLE 3. Evaluation of Unsupervised Multiclass Tool Wear Recognition Frameworks Using 5-Fold Cross-Validation: Macro-F1 (%) of Self-Organizing Map Models Built With Different Feature Learning Approaches

Feature learning	Milling dataset DS1			Milling dataset DS2		
	Microphone	Accelerometer	Sensor fusion	Microphone	Accelerometer	Sensor fusion
Laplacian Score	**79.29**	**78.79**	**87.52**	**69.08**	**71.07**	75.80
SAE	71.83	67.95	77.42	56.80	61.89	67.65
SSAE	72.75	69.57	78.32	56.30	63.86	71.38
No feature learning	76.69	72.90	85.50	65.16	66.26	**77.68**

Bold values indicate the highest macro-F1 value for the corresponding dataset and sensor(s).

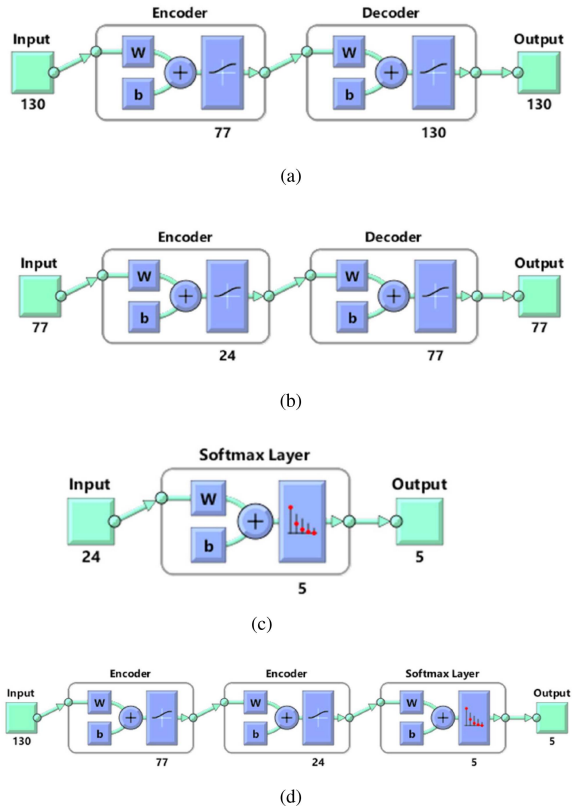

FIGURE 8. Neural network architectures for the unsupervised pretraining and supervised fine-tuning phases of the semisupervised SSAE. Input and output dimensionalities in this illustration correspond to the DS1-sensor fusion case. (a) First SAE. (b) Second SAE. (c) Softmax layer for a five-class classification. (d) Stacked network comprising the two trained encoders and Softmax layer.

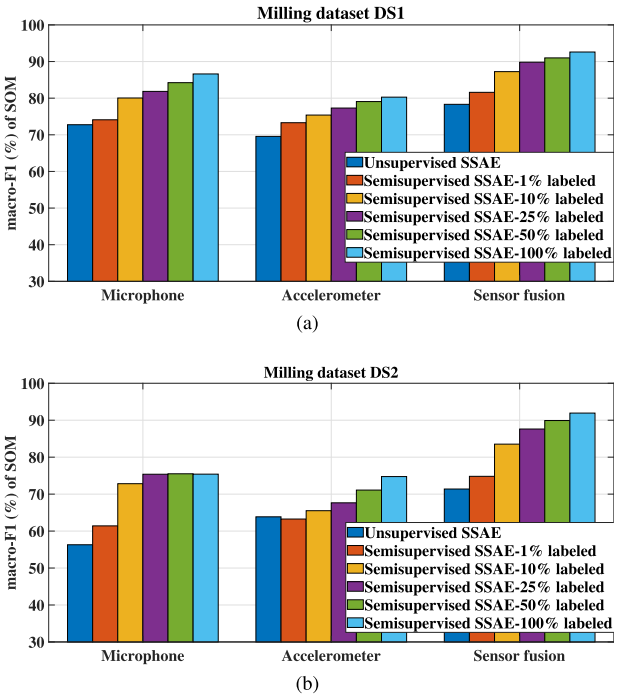

FIGURE 9. Macro-F1 variation of SOM over different feature representations learned by the unsupervised SSAE, and the semisupervised SSAE under different percentages of labeled training observations. In all the cases, SOM was trained in an unsupervised fashion using the total training observations. The performance variation is linked to the quality of feature representation. Overall, SOM was able to detect more useful data patterns for the five-class classification task as more labeled observations were involved in the fine-tuning of SSAE, indicating an increasingly discriminative feature representation. (a) Milling dataset DS1. (b) Milling dataset DS2.

It can be seen that SOM benefits from the features learned by the semisupervised SSAE as it achieved higher macro-F1 values compared to those achieved with the unsupervised SSAE in the fully unsupervised frameworks. This indicates that, with the semisupervsied SSAE, SOM was able to detect more useful data patterns for the tool state classification task, despite the absence of labels in the SOM training in all the cases. It also emphasizes our previous explanation related to the incapability of the unsupervised SSAE to yield a feature representation that is sufficiently discriminative, owing to its reliance on the unsupervised reconstruction error. On the other hand, adding the fine-tuning phase that incorporates a supervised criterion, i.e., in the semisupervised SSAE cases, allows tuning the latent feature representation in accordance with the classification task at hand, which led to a more discriminative feature representation. Overall, the macro-F1 of SOM gets higher as more labeled data are involved in the feature learning, owing to an improved fine-tuning of SSAE. An exception can be seen with the DS2-microphone, where macro-F1 saturated after adding 25% of the labeled training data, and did not improve thereafter.

Even though SSAE was also used with supervised classifiers in the other proposed semisupervised frameworks, the framework with SOM is more suitable to reveal the impact of the supervised fine-tuning phase on SSAE, as well as the role that the size of labeled data plays in this process. This is due to two reasons: 1) SOM training is unsupervised, and thus, is solely based on the feature space characteristics, and unlike the supervised classifiers, is not biased by the labels. Hence, the macro-F1 variation of SOM is strongly related

TABLE 4. Evaluation of Semisupervised Multiclass Tool Wear Recognition Frameworks Using Five-fold Cross-Validation: Macro-F1 (%) of Different Supervised and Unsupervised Classifiers With Different SSAE-Based Feature Learning Configurations

Milling dataset	Sensor(s)	Semisupervised framework	Percentage of labeled training observations				
			100%	50%	25%	10%	1%
DS1	Microphone	SOM with semisupervised SSAE	86.61	84.23	81.85	**80.05**	**74.10**
		Softmax with semisupervised SSAE	89.98	86.17	82.92	79.66	69.91
		Softmax with unsupervised SSAE	72.14	72.21	71.90	71.89	68.91
		SVM with semisupervised SSAE	**90.15**	**86.62**	**82.99**	80.03	70.14
		SVM with unsupervised SAE	74.46	74.06	73.69	72.09	65.14
		RF with semisupervised SSAE	89.80	86.45	82.82	79.17	70.43
		RF with unsupervised SSAE	77.01	76.68	75.72	73.98	67.16
	Accelerometer	SOM with semisupervised SSAE	80.28	79.07	77.30	**75.39**	**73.30**
		Softmax with semisupervised SSAE	84.63	81.34	77.36	74.74	70.52
		Softmax with unsupervised SSAE	68.53	68.51	68.87	69.10	66.41
		SVM with semisupervised SSAE	**84.80**	81.33	77.95	75.24	69.11
		SVM with unsupervised SSAE	60.95	63.50	66.94	68.33	57.93
		RF with semisupervised SSAE	84.75	**81.65**	**78.76**	75.02	70.51
		RF with unsupervised SSAE	75.28	74.76	74.19	72.80	67.82
	Sensor fusion	SOM with semisupervised SSAE	92.60	91.00	89.82	87.24	**81.59**
		Softmax with semisupervised SSAE	93.65	92.54	91.31	88.82	77.87
		Softmax with unsupervised SSAE	80.71	80.68	80.55	80.22	74.91
		SVM with semisupervised SSAE	**93.91**	**93.00**	**91.50**	**88.97**	76.82
		SVM with unsupervised SSAE	85.39	84.44	83.23	80.51	71.29
		RF with semisupervised SSAE	93.54	92.23	90.75	88.63	78.98
		RF with unsupervised SSAE	86.55	85.42	84.00	82.66	75.99
DS2	Microphone	SOM with semisupervised SSAE	75.41	75.51	75.39	72.83	**61.41**
		Softmax with semisupervised SSAE	85.56	80.87	77.01	74.48	59.27
		Softmax with unsupervised SSAE	54.75	54.75	54.88	54.59	52.20
		SVM with semisupervised SSAE	**85.75**	**81.04**	**77.44**	**74.62**	58.53
		SVM with unsupervised SSAE	62.12	63.66	63.24	61.28	50.30
		RF with semisupervised SSAE	85.27	80.81	76.91	73.78	59.27
		RF with unsupervised SSAE	69.24	67.59	65.97	63.23	54.57
	Accelerometer	SOM with semisupervised SSAE	74.75	71.10	**67.65**	**65.52**	**63.26**
		Softmax with semisupervised SSAE	77.44	**71.83**	67.12	63.51	59.92
		Softmax with unsupervised SSAE	63.14	63.01	63.24	62.61	61.28
		SVM with semisupervised SSAE	**77.48**	**71.83**	67.13	62.62	56.12
		SVM with unsupervised SSAE	29.11	34.79	38.40	46.74	57.08
		RF with semisupervised SSAE	77.22	71.48	66.65	61.27	57.49
		RF with unsupervised SSAE	62.11	61.83	61.29	60.16	57.68
	Sensor fusion	SOM with semisupervised SSAE	91.94	89.92	87.62	83.53	**74.82**
		Softmax with semisupervised SSAE	93.51	91.77	**90.18**	**86.86**	70.74
		Softmax with unsupervised SSAE	77.71	77.60	77.53	77.15	65.98
		SVM with semisupervised SSAE	**93.70**	**91.89**	90.08	86.74	69.48
		SVM with unsupervised SSAE	81.77	80.41	78.67	75.79	63.28
		RF with semisupervised SSAE	93.32	91.65	89.76	86.38	72.88
		RF with unsupervised SSAE	82.96	81.99	80.88	79.29	71.29

Bold values indicate the best performance for the corresponding case.

to the SSAE performance; 2) in the frameworks containing supervised classifiers with semisupervised SSAE, the labeled data not only affect the quality of feature learning but also the size of training data fed to the classification algorithm. Thus, the macro-F1 variation over different percentages of labeled data in these frameworks can also be related to the impact of training data size on the classifier training. Therefore, Fig. 9 is not only important for illustrating the impact of feature learning quality on SOM but also for observing the SSAE performance without/with supervised fine-tuning under different sizes of labeled data. The impact of the fine-tuning phase on the feature learning quality is particularly evident when comparing the performance obtained with the following two extreme cases: 1) unsupervised SSAE and 2) the semisupervised SSAE under 100% labeled training observations. For example, with the multisensor data of DS2, the macro-F1 of SOM was 71.38% and 91.94%, for the two aforementioned cases, respectively.

2) COMPARISON OF SEMISUPERVISED FRAMEWORKS

Table 4 shows the predictive performance of the classification models for all the semisupervised frameworks. Given a certain supervised classifier, the impact of fine-tuning phase of SSAE can be revealed by comparing the performance attained with the unsupervised SSAE and that with the semisupervised SSAE, at the same percentage of labeled training data. Considering the overall trend over all the datasets and sensors, the supervised classifiers achieved a higher macro-F1 with the semisupervised SSAE compared to the unsupervised SSAE, indicating a higher quality feature representation reached by the former. Thus, even with the supervised classifiers that use data labels, the features learned by the unsupervised SSAE cannot lead to a better or comparable performance to that of the semisupervised SSAE. The performance difference between the cases of unsupervised and semisupervised SSAEs gets more significant as more labeled data are involved in the frameworks. Taking the accelerometer of the milling dataset

DS1 as an example, the macro-F1 difference between the two SSAE configurations under the 100% percentage of labeled data reached 16.10%, 23.85%, and 9.47% for the Softmax, SVM, and RF, respectively, whereas the difference with 1% labeled observations reached 4.11%, 11.18%, and 2.69%, for the aforementioned three classifiers, respectively. Even though the Softmax classifier was used in the fine-tuning phase of the semisupervised SSAE, all the classifiers benefit from the enhanced feature representation (to an extent that exceeded the Softmax in some cases), reflecting a generic utility of the fine-tuning approach. With the semisupervised-SSAE features, the performance difference between the supervised classifiers is marginal, especially as the percentage of labeled data gets larger. However, with the unsupervised SSAE, the difference is more significant, with RF being the superior in most of the cases. For example, in the case of unsupervised SSAE of DS1-accelerometer with 100% labeled data, RF outperformed Softmax and SVM by 6.75% and 14.33%, respectively, whereas the performance difference for any classifier pair in the case of semisupervised SSAE did not exceed 0.17%. This observation can also be seen in many other cases, indicating that the choice of the supervised classifier becomes more critical with less-discriminative features.

It is noteworthy that, with the multisensory features learned by the semisupervised SSAE, all the supervised classifiers maintained a macro-F1 of over or nearly 90% even when only 25% of the training data were labeled. Moreover, the multisensor data showed a considerably greater robustness to the decrease of labeled observations, compared to the single sensor data. For example, moving from 100% to 25% labeled training data for DS1 and DS2, the macro-F1 of SVM corresponding to the sensor fusion dropped by 2.41% and 3.62%, respectively, whereas it decreased by 7.16% and 8.31%, respectively, for the microphone, and by 6.85% and 10.35%, respectively, for the accelerometer.

3) CHOICE OF SEMISUPERVISED FRAMEWORK

As mentioned previously, the overall results suggest that, with partially labeled data, a better predictive performance can be attained with the semisupervised SSAE-based feature learning that uses the available labeled data for fine-tuning the unsupervised SSAE, regardless of the size of labeled data and the subsequent classifier. However, when it comes to determining the best classifier type to be used with this feature learning method, it is quite dependent on the size of labeled training data. Based on our results, the supervised classifiers outperformed SOM when the size of labeled data seemed to still be sufficient for them to build well-generalized models. However, this performance difference diminishes as the labeled observations get fewer, until it reaches a point where SOM became the superior. More specifically, when only 1% of the training data are labeled, SOM consistently outperformed the other supervised classifiers for all the milling datasets and sensors. Even for larger labeled data percentages, SOM showed a comparable or better performance in some cases, e.g., accelerometer of DS2. Thus, when the size of labeled data is not sufficient, using an unsupervised classifier that makes full use of the training observations without any labels is more promising than using a supervised classifier that can only use the few available labeled observations along with their labels. This holds true regardless of the quality of the feature learning, as it can also be seen that when only 1% of training data are labeled, the supervised classifiers with the unsupervised SSAE achieved a lower performance than that of SOM with the unsupervised SSAE (see Table 3). Moreover, it is noteworthy that even in the cases where the small labeled data were not sufficient to build supervised classification models with high macro-F1, they were still beneficial to fine-tune the latent feature space learned by the unsupervised SSAE.

4) FURTHER INSIGHTS ON FEATURE LEARNING QUALITY

Among the different considered percentages of labeled training data, some important insights can be derived from the case corresponding to 100% as it is the only case in which the size of training data is the same for SOM and the supervised classifiers, allowing for observing the impact of other factors affecting their performance differences. It can be seen that, for all the datasets and sensors corresponding to this case, the performance of SOM with semisupervised SSAE features substantially surpasses that of the supervised classifiers with the unsupervised SSAE, even though the supervised classifiers use the total labels. For example, considering the sensor fusion of DS2, the macro-F1 of SOM with semisupervised SSAE was 91.94%, which is larger than that of Softmax, SVM, and RF with unsupervised SSAE by 14.23%, 10.17%, and 8.98%, respectively. This proves that the feature learning quality can be a more significant factor of the predictive performance than the presence of labels during training the classifiers, i.e., high-discriminative features without labels are more powerful than low-discriminative features with their labels, under the same data dimensionality and size.

VI. CONCLUSION

This article addressed constraints that can limit the feasibility and applicability of the existing TCM solutions in real-world manufacturing settings. These include the heavy reliance on the supervised learning that requires labeled data, the focus on binary decision-based unsupervised solutions, and not considering scarce labeled data when it comes to the semisupervised solutions. To this end, we proposed different unsupervised and semisupervised multiclass tool wear recognition frameworks that can deal with fully unlabeled and partially labeled data, respectively. Concerning the unsupervised frameworks, SOM was used for model building, and three methods were considered for feature learning, namely LS, SAE, and SSAE. To gain broad insights across the different semisupervised frameworks, we considered scenarios where the labeled data influence only feature learning, classifier building, or both. The underlying methods include the two following SSAE-based feature learning methods:

1) unsupervised SSAE (trained in an unsupervised greedy layerwise fashion) and 2) a semisupervised SSAE (following the previous unsupervised training on the full data, the encoders undergo a supervised global fine-tuning phase using the available labeled data), and the following classifiers: SOM (unsupervised), Softmax, SVM, and RF (supervised). The frameworks were experimentally evaluated for a five-class tool wear recognition using two run-to-failure datasets of milling tools, recorded using a microphone and an accelerometer. Single sensor data and multisensor data, as well as different percentages of labeled training observations were considered in the evaluation. The main conclusions from the experimental results can be summarized as follows.

1) Regarding the fully unsupervised frameworks, the macro-F1 of SOM with the LS-based FS scheme was higher than those with SAE and SSAE by an average (over all the sensor configurations and milling datasets) of 9.67% and 8.23%, respectively. This would be attributed to the fact that LS selects the features based on their locality preserving power, a property that is significant for discrimination. On the other hand, the reconstruction error criterion considered in training (S)SAE does not necessarily lead to a discriminative feature space.
2) Compared to the unsupervised SSAE, the semisupervised SSAE led to a considerably better predictive performance for all the classifiers (e.g., the macro-F1 difference reached 20.56% for SOM in the sensor fusion of dataset DS2), owing to the fine-tuning phase that tunes the latent feature representation in accordance with the classification task, which led to a more discriminative feature representation. The improvement was more evident with more labeled data added. Even though a Softmax-based classification layer was used in the fine-tuning phase, the resulting features were beneficial for all the classifiers, indicating a generic utility of the approach.
3) High-discriminative features without labels are more powerful than low-discriminative features with their labels, under the same data dimensionality and size. This was indicated by comparing the performance achieved by SOM and the semisupervised SSAE, against that achieved by the supervised classifiers and unsupervised SSAE, over all the datasets and sensors.
4) For the semisupervised frameworks, the overall results suggest that the semisupervised SSAE was the superior for feature learning, regardless of the size of labeled data. As for the choice of the classifier, it depends on the labeled data size. When the labeled data are small and not sufficient to build well-generalized supervised models, particularly when only 1% of the training data were labeled, SOM outperformed the other supervised classifiers, owing to its use of the total training observations since no labels are needed during training. Otherwise, the supervised classifiers showed a better performance.
5) With the multisensor features learned by the semisupervised SSAE, all the supervised classifiers maintained a macro-F1 of over or nearly 90% when the percentage of labeled training data dropped to 25%. Moreover, multisensor data showed a greater robustness to the decrease of labeled observations, compared to the single sensor data.

For the future work, we will focus on two factors that were proven in this study to be promising for combating the unavailability or scarcity of labeled data, namely, sensor fusion and feature representation quality. In this regard, we consider investigating the combination of different multisensor feature representations, e.g., combining LS-selected features with SSAE-learned features in one representation, as well as other variants of AE, e.g., convolutional AE.

REFERENCES

[1] M. Assafo and P. Langendörfer, "A TOPSIS-Assisted feature selection scheme and SOM-based anomaly detection for milling tools under different operating conditions," *IEEE Access*, vol. 9, pp. 90011–90028, 2021, doi: 10.1109/ACCESS.2021.3091476.

[2] M. Abubakr, M. A. Hassan, G. M. Krolczyk, N. Khanna, and H. Hegab, "Sensors selection for tool failure detection during machining processes: A simple accurate classification model," *CIRP J. Manuf. Sci. Technol.*, vol. 32, pp. 108–119, Jan. 2021.

[3] G. Herrera-Granados, T. Misaka, J. Herwan, H. Komoto, and Y. Furukawa, "An experimental study of multi-sensor tool wear monitoring and its application to predictive maintenance," *Int. J. Adv. Manuf. Technol.*, vol. 133, pp. 3415–3433, Jun. 2024.

[4] M. Ahmed et al., "Tool health monitoring of a milling process using acoustic emissions and a ResNet deep learning model," *Sensors*, vol. 23, no. 6, 2023, Art. no. 3084.

[5] V. Nasir and F. Sassani, "A review on deep learning in machining and tool monitoring: Methods, opportunities, and challenges," *Int. J. Adv. Manuf. Technol.*, vol. 115, no. 9, pp. 2683–2709, 2021.

[6] W. J. Lee, G. P. Mendis, and J. W. Sutherland, "Development of an intelligent tool condition monitoring system to identify manufacturing tradeoffs and optimal machining conditions," *Procedia Manuf.*, vol. 33, pp. 256–263, Jan. 2019.

[7] T. Żabiński, Z. Hajduk, J. Kluska, and L. Gniewek, "FPGA-embedded anomaly detection system for milling process," *IEEE Access*, vol. 9, pp. 124059–124069, 2021.

[8] M. Assafo, J. P. Städter, T. Meisel, and P. Langendörfer, "On the stability and homogeneous ensemble of feature selection for predictive maintenance: A classification application for tool condition monitoring in milling," *Sensors*, vol. 23, no. 9, May 2023, Art. no. 4461, doi: 10.3390/s23094461.

[9] C. S. Wickramasinghe, K. Amarasinghe, and M. Manic, "Deep self-organizing maps for unsupervised image classification," *IEEE Trans. Ind. Inform.*, vol. 15, no. 11, pp. 5837–5845, Nov. 2019.

[10] C. Cooper, J. Zhang, R. X. Gao, P. Wang, and I. Ragai, "Anomaly detection in milling tools using acoustic signals and generative adversarial networks," *Procedia Manuf.*, vol. 48, pp. 372–378, Jan. 2020.

[11] S. Yan, H. Shao, Y. Xiao, B. Liu, and J. Wan, "Hybrid robust convolutional autoencoder for unsupervised anomaly detection of machine tools under noises," *Robot. Comput.- Integr. Manuf.*, vol. 79, Feb. 2023, Art. no. 102441.

[12] Y. Gui, Z.-Q. Lang, Z. Liu, Y. Zhu, H. Laalej, and D. Curtis, "Unsupervised detection of tool breakage: A novel approach based on time and sensor domain data analysis," *IEEE Trans. Instrum. Meas.*, vol. 72, 2023, Art. no. 3524813.

[13] A. Patange, R. N. Soman, S. Pardeshi, M. Kuntoglu, and W. Ostachowicz, "Milling cutter fault diagnosis using unsupervised learning on small data: A robust and autonomous framework," *Eksploatacja i Niezawodność-Maintenance Rel.*, vol. 26, no. 1, 2024, Art. no. 178274, doi: 10.17531/ein/178274.

[14] Z. Li, W. Zhong, W. Liao, Y. Cai, J. Zhao, and G. Wang, "A robust tool condition monitoring system based on cluster density under variable machining processes," *Appl. Sci.*, vol. 13, no. 12, 2023, Art. no. 7226.
[15] B. Brenner et al., "Better safe than sorry: Risk management based on a safety-augmented network intrusion detection system," *IEEE Open J. Ind. Electron. Soc.*, vol. 4, pp. 287–303, 2023.
[16] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Inf. Fusion*, vol. 44, pp. 78–96, 2018.
[17] L. E. E. Ochoa, I. B. R. Quinde, J. P. C. Sumba, A. V. Guevara Jr., and R. Morales-Menendez, "New approach based on autoencoders to monitor the tool wear condition in HSM," *IFAC-PapersOnLine*, vol. 52, no. 11, pp. 206–211, Jan. 2019.
[18] J. Ou, H. Li, G. Huang, and Q. Zhou, "A novel order analysis and stacked sparse auto-encoder feature learning method for milling tool wear condition monitoring," *Sensors*, vol. 20, no. 10, 2020, Art. no. 2878.
[19] T. Gittler, S. Scholze, A. Rupenyan, and K. Wegener, "Machine tool component health identification with unsupervised learning," *J. Manuf. Materials Process.*, vol. 4, no. 3, 2020, Art. no. 86.
[20] L. C. Brito, M. B. da Silva, and M. A. V. Duarte, "Identification of cutting tool wear condition in turning using self-organizing map trained with imbalanced data," *J. Intell. Manuf.*, vol. 32, no. 1, pp. 127–140, Jan. 2021.
[21] D. Mishra, U. Awasthi, K. R. Pattipati, and G. M. Bollas, "Tool wear classification in precision machining using distance metrics and unsupervised machine learning," *J. Intell. Manuf.*, pp. 1–25, Nov. 2023. [Online]. Available: https://doi.org/10.1007/s10845-023-02239-5
[22] B. Liu, H. Li, J. Ou, Z. Wang, and W. Sun, "Intelligent recognition of milling tool wear status based on variational auto-encoder and extreme learning machine," *Int. J. Adv. Manuf. Technol.*, vol. 119, pp. 4109–4123, 2022.
[23] J. Gao, J. Liu, and X. Yu, "Research on tool condition monitoring (TCM) using a novel unsupervised deep neural network (DNN)," *J. Vibroeng.*, vol. 26, no. 1, pp. 193–208, Feb. 2024.
[24] C. Shi, G. Panoutsos, B. Luo, H. Liu, B. Li, and X. Lin, "Using multiple-feature-spaces-based deep learning for tool condition monitoring in ultraprecision manufacturing," *IEEE Trans. Ind. Electron.*, vol. 66, no. 5, pp. 3794–3803, May 2019, doi: 10.1109/TIE.2018.2856193.
[25] B. Liu, C. H. Chen, P. Zheng, and G. Zhang, "An adaptive parallel feature learning and hybrid feature fusion-based deep learning approach for machining condition monitoring," *IEEE Trans. Cybern.*, vol. 53, no. 12, pp. 7584–7595, Dec. 2023.
[26] J. S. Nam and W. T. Kwon, "A study on tool breakage detection during milling process using LSTM-autoencoder and Gaussian mixture model," *Int. J. Precis. Eng. Manuf.*, vol. 23, no. 6, pp. 667–675, 2022.
[27] J. Dou, C. Xu, S. Jiao, B. Li, J. Zhang, and X. Xu, "An unsupervised online monitoring method for tool wear using a sparse auto-encoder," *Int. J. Adv. Manuf. Technol.*, vol. 106, pp. 2493–2507, 2020.
[28] Y. Jia and G. Li, "A two-stage feature selection method for hob state recognition," *Eng. Appl. Artif. Intell.*, vol. 133, Jul. 2024, Art. no. 108580.
[29] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2006, pp. 507–514.
[30] 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.
[31] T. Kohonen, "Essentials of the self-organizing map," *Neural Netw.*, vol. 37, pp. 52–65, Jan. 2013.
[32] K. Jemielniak, T. Urbański, J. Kossakowska, and S. Bombiński, "Tool condition monitoring based on numerous signal features," *Int. J. Adv. Manuf. Technol.*, vol. 59, pp. 73–81, Mar. 2012.
[33] G. Luo, C. Yao, Y. Liu, Y. Tan, J. He, and K. Wang, "Stacked auto-encoder based fault location in VSC-HVDC," *IEEE Access*, vol. 6, pp. 33216–33224, 2018, doi: 10.1109/ACCESS.2018.2848841.
[34] M. Jongmanns, P. Städter, and T. Meisel, "Dataset for AI-assisted detection of the wear level of a cutting tool on a CNC mill," 2023. [Online]. Available: https://fordatis.fraunhofer.de/handle/fordatis/348

MARYAM ASSAFO received the B.Sc. and M.Sc. degrees in communications engineering from the University of Aleppo, Aleppo, Syria, in 2013 and 2018, respectively. She is currently working toward the Ph.D. degree in computer science with Brandenburg University of Technology (BTU) Cottbus-Senftenberg, Cottbus, Germany.

She is currently a Research Assistant with the Chair of Wireless Systems, BTU Cottbus-Senftenberg. Her research interests include predictive maintenance, sensor fusion, data mining, signal processing, and artificial intelligence.

PETER LANGENDOERFER received the Diploma degree in computer science from the Technical University (TU) of Braunschweig, Braunschweig, Germany, in 1995, and the Ph.D. degree in computer science from the Brandenburg University of Technology (BTU), Cottbus, Germany, in 2001.

Since 2000, he has been with the Leibniz Institute for High Performance Microelectronics, Frankfurt (Oder), Germany, where he leads the Wireless Systems Department. From 2012 to 2020, he lead the Chair for Security in pervasive systems with the Technical University of Cottbus-Senftenberg, Cottbus, Germany, where he has owned the Chair of Wireless Systems, since 2020. He has authored or coauthored more than 150 refereed technical articles, filed 17 patents of which ten have been granted already. His reserach interests include security for resource constraint devices, low power protocols, efficient implementations of A means and resilience.

Dr. Langendoerfer was a Guest Editor for many renowned journals, e.g., *Wireless Communications and Mobile Computing* (Wiley) and *ACM Transactions on Internet Technology*.