Applied Soft Computing 117 (2022) 108397

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Contrastive learning based self-supervised time-series analysis

Johannes Pöppelbaum *, Gavneet Singh Chadha, Andreas Schwung

South Westphalia University of Applied Sciences, Lübecker Ring 2, Soest, 59494, Germany

ARTICLE INFO

Article history:
Received 8 April 2021
Received in revised form 8 December 2021
Accepted 24 December 2021
Available online 4 January 2022

Keywords:
Self-supervised learning
Time-series analysis
Convolutional neural networks
Data augmentation

ABSTRACT

Deep learning architectures usually require large scale labeled datasets for achieving good performance on general classification tasks including computer vision and natural language processing. Recent techniques of self-supervised learning have opened up new a research frontier where deep learning architectures can learn general features from unlabeled data. The task of self-supervised learning is usually accomplished with some sort of data augmentation through which the deep neural networks can extract relevant information. This paper presents a novel approach for self-supervised learning based time-series analysis based on the *SimCLR* contrastive learning. We present novel data augmentation techniques, focusing especially on time-series data, and study their effect on the prediction task. We provide comparison with several fault classification approaches on benchmark Tennessee Eastman dataset and report an improvement to 81.43% in the final accuracy using our contrastive learning approach. Furthermore we report a new benchmark of 80.80%, 81.05% and 81.43% for self-supervised training on Tennesee Eastman where a classifier is only trained with 5%, 10% or 50% percent of the available training data. Hence, we can conclude that the contrastive approach is very successful in time-series problems also, and further suitable for usage with partially labeled time-series datasets.

1. Introduction

Deep neural networks (DNN) have been known to provide superior performance in a number of application fields such as computer vision, natural language processing and also recently in the field of natural sciences [1]. However, DNN also require large scaled labeled dataset for achieving such remarkable performance. The availability of such labeled data-sets is not always guaranteed, especially in an industrial application where labeling is either too expensive or not possible at all. In such cases, semi-supervised or self-supervised learning approaches are the only alternative for the prediction task.

The advent of the deep learning era started with unsupervised pre-training and supervised fine-tuning [2,3], but since then deep supervised models have proven to be the standard in image classification [4] and machine translation [5] tasks. However, there has been a recent surge in pre-training of DNN for machine translation tasks in [6,7]. Such pre-training techniques are possible for language models since the natural sequential nature of text data allows for predicting the occurrence of the next most probable words given the surrounding words or from the context. Similar approaches are however not transferable to other sequential or time-series data, as a natural sequence of occurrences or context is not always present.

Therefore, to overcome the challenge of learning a good representation from unlabeled data, recent approaches for self-supervised learning have been used mainly for visual representation learning. As surveyed in [8], the self-supervised learning models can be categorized into generative, contrastive and a hybrid of generative–contrastive methods. In generative self-supervised learning, the model is tasked to predict part of an input, from a given observed part of the input. There are various tasks predicting from the input without any labels, which are termed as pretext tasks [9], which help in learning a good representation for the relevant downstream task. The most commonly used pretext tasks include pixel image generation [10,11]. A common pretext task for time-series analysis is to train an AE for reconstruction, and use the final hidden representation as an input for the downstream task of classification or regression, as has been previously reported in [12,13].

In this work, we propose a SimCLR [14] based self-supervised learning based industrial time-series (SimCLR-TS) analysis framework, where we leverage the existing methodologies of self-supervised learning in image recognition, in order to achieve better performance on time-series data-sets. Recently, contrastive learning approaches have achieved astonishing performance on certain downstream tasks like image recognition [14–16]. The main idea for these approaches is to learn a higher level feature representation where the data itself provides supervision. In the SimCLR framework, a composition of multiple data-augmentation

* Corresponding author.
E-mail address: poeppelbaum.johannes@fh-swf.de (J. Pöppelbaum).

https://doi.org/10.1016/j.asoc.2021.108397
1568-4946/

techniques is used for characterizing the similarities and dissimilarities between images in a batch. Specifically, similar samples should be encoded by the trained network to have similar representations, while the dissimilar samples should be encoded to have dissimilar representations. The similar and dissimilar samples are obtained by data augmentation techniques, with the assumption that augmentation samples from the same source image are similar while all the other images being dissimilar. The data-augmentation presented in [14] are standard image transformations such as rotation, crop and resize and color distortion, which have been proposed in previous studies [17,18]. Such transformations cannot be applied as they are for time-series signals because of the inherent characteristics of temporal and dynamic dependencies in a multivariate time-series setting. Therefore, we propose multiple augmentation techniques for time-series signal and test their effectiveness on the contrastive learning task.

By adapting SimCLR to SimCLR-TS, we intent to create a representative and distinguishable intermediate representation of the raw data to allow a more accurate fault classification in time-series problems and to enable the usage of only partially labeled data.

We test the proposed framework on the benchmark Tennessee Eastman Process (TEP) dataset for fault classification leading to a significant increase in fault classification accuracy as compared to standard supervised learning approaches. In addition, we only use fractions of the training data for training the classifier to simulate a partially labeled dataset and to further confirm our approach. We provide a thorough comparison on the effectiveness of each of the data-augmentation techniques for fault classification. The results constitute a new state-of-the art in the multiclass classification on the TEP dataset and a new benchmark for only using fractions of the TEP training data.

Major contributions: The major contributions of the paper can be summarized as follows:

- We present SimCLR-TS, a new self-supervised learning based approach for industrial time-series analysis which uses data-augmentation techniques to learn feature representation for downstream classification tasks. The framework is based on data-augmentation techniques on raw time-series data.
- We present multiple novel data-augmentation techniques for time-series analysis such as bidirectional flipping, channel permutation, crop- and resize for time-series and random smoothing, also we utilize some standard techniques like noise injection or blockout for maximizing the contrastive learning ability among the different sequences.
- We provide a thorough comparison on the effectiveness of both the proposed data augmentation techniques as well as the proposed contrastive learning framework and its variants on the benchmark TEP dataset for supervised as well as semi-supervised fault classification.

The paper is organized as follows. Section 2 presents the related work. In Section 3 we present the SimCLR-TS framework along with the different data-augmentation techniques. In Section 4 we provide a thorough comparison of the convolutional neural networks and compare the performance with existing methods from the literature. Section 5 concludes the paper.

2. Related work

In this section, we discuss the various research fields related to our paper. This includes existing approaches for supervised and semi-supervised deep learning based time-series fault classification of industrial processes, contrastive learning based modeling methods and data augmentation techniques for image and time-series data.

Supervised learning multi-class classification: Various approaches exist in the field of deep learning based fault classification, as has been surveyed in [19,20]. Fundamentally, the previous works can be categorized into supervised, semi-supervised and unsupervised learning based methods. Most of the semi-supervised and unsupervised methods do not consider time-series based fault classification, but rather just point based fault classification. Long Short Term Memory are a type of recurrent neural network which can model such time dependencies, and have been applied for fault diagnosis of wind turbine in [21] and fault detection in the TEP in [22]. Convolutional Neural Networks (CNN) are the most common applied architecture for time-series based fault classification. They have been applied for identifying different health conditions of wind turbine gearbox in [23], for fault diagnosis of an induction motor in [24], for fault diagnosis of a motor bearing, self-priming centrifugal pump and axial piston hydraulic pump in [25] and fault diagnosis of two rotating machinery in [26]. A fault classification technique for the TEP with Principal Component Analysis (PCA) and Support Vector Machines (SVM) has been proposed in [27]. A similar fault classification approach with slow feature analysis and Relevance Vector Machine (RVM), although without the normal class classification, has been proposed in [28]. Recently, CNN along with wavelet transform techniques have been applied to the benchmark TEP, also without the normal class classification, in [29] for fault classification. A similar approach with CNN and LSTM has been proposed in [30] but the benchmark test dataset has been manipulated in this study, such that training and testing data was combined and a new testing data-set was formed by randomly selecting 20% of the combined samples, yielding results which are not comparable. Additionally, a stacked supervised AE approach for binary fault classification of the TEP has been presented in [31]. Finally, random forest have been investigated for fault classification of the TEP in [32], but also without the normal class classification.

Semi-supervised multi-class classification: In the case of semi-supervised learning methods, usually a reconstruction based deep AE or a Restricted Boltzmann Machine (RBM) is trained in an unsupervised manner followed by a fine-tuning step of training the penultimate layer in a supervised manner with labels. A stacked RBM or a deep belief network based fault diagnosis of the TEP has been presented in [33]. A similar deep belief network approach for rolling bearing fault diagnosis was investigated in [34]. A fault diagnosis scheme for motor bearings and a locomotive bearings based on sparse AE has been presented in [35]. Similar approaches with sparse AE for induction motor faults classification and bearing fault diagnoses has been presented in [36,37]. Our approach differs from the above methods such that, we employ a contrastive learning approach to learn a higher level representation, rather than a reconstruction learning based approach.

Another approach for SSL in the context of image classification is MixMatch, ReMixmatch and Fixmatch [38–40], where augmented samples from unlabeled data are utilized to predict/create pseudo-label or artificial label with slightly different strategies. These labels are then used to calculate an auxiliary loss for the unlabeled data in addition to the loss from the labeled samples. Here, too, our approach differs as we target a two-staged approach where our pre-training does not rely on any kind of labels at all as it is fully unsupervised and is intended to learn meaningful representations first instead of directly targeting classification.

Representation Learning: Also coming from image tasks are the approaches of [41,42]. They do a pre-training by bringing a model to solve tasks for which they assume a detailed understanding about the content of the image is necessary. Specifically, [41] applies the task of predicting the position of one extracted patch in an image to another extracted patch. [42] on the other hand rotates an image either 0, 90, 180 or 270 degree and the model has to predict the respective applied rotation. The learned representation can then be used for downstream tasks. A further pretext task is introduced by [43] where representations are learned by solving a jigsaw puzzle of 9 patches extracted from an image. All share the commonality that the labels are inherently defined by the action taken on the input image. They differ from our contrastive learning approach as they still rely on labels, despite not needing an initially labeled dataset. Also, different patches/parts in images are related to each other whereas in time-series data the channels cannot generally be assumed to contain spatial relationships and are permutation invariant. Hence, the approaches relying on patch position are not applicable. For predicting the rotation this is similar as the time-series data cannot be assumed to be square such that only a 180 degree rotation can be applied without padding. Furthermore, the 1D-CNN would stride over the channel and not over the time-dimension anymore in case of a 90 or 270 degree rotation such that the data seen from the model is completely different in this cases.

Contrastive learning: The original idea for contrastive learning was proposed in [44,45] wherein comparison of different samples, without any labels was investigated. There have been a number of studies since then using a similar principle, which have been surveyed in [8,46]. The proposed contrastive learning methods can be categorized into Context-Instance and Context-Context contrast methods. The methods falling under the Context-Instance branch usually work with e.g. the principle of predicting relative position [41] or with maximizing mutual information [47]. The Context-Context methods propose representation learning in a discriminative fashion from individual instances. A deep clustering approach for jointly learning neural network parameters and cluster assignments has been proposed in [48]. A Swapping Assignment between multiple Views (SwAV) approach which uses a set of trainable "code" vectors to compute encoded features from different augmentations and clustering assignment for each of the augmentations has been presented in [49]. An instance discrimination approach with a memory bank network has been proposed in [50] for learning visual representations. A Momentum Contrast (MoCo) [15] approach has been investigated with a special momentum-updated encoder for learning representations that transfer well to downstream tasks. None of the above studies consider contrastive learning approach for time-series analysis

Similar works to our proposed are CLAR [51] and SeqCLR [52]. However, both distinguish from our work. First of all, both deal with very specific signals whereas we deal with multi-variate time-series data of a arbitrary and mixed physical units which can affect the choice of suitable augmentations. CLAR does classification of audio samples based on SimCLR, but they also utilize time–frequency domain whereas we work only on the raw data and they do not need to deal with multiple channels of different signals. Furthermore, their pre-training is not fully unsupervised but instead incorporates a supervised training component. Hence, this is aimed for boosting the raw performance, we instead aim to create the most meaningful latent representation in a fully unsupervised way. Also, they train their evaluation head always on the fully labeled data, despite using a fraction of training data for training their encoder. [51] We, on the contrary, train our final classifier with only fractions of the labeled data which enables our approach to work with partially labeled datasets. SeqCLR is specifically concerned with EEG signals and they aim to extract features from a single channel and to learn a corresponding representation for this single channel, so they feed their data sequentially whereas we feed and process multichannel data of different physical units simultaneously. In addition to that, they combine channels to form new ones and recombination of different datasets as a significant performance booster, which is not applicable with arbitrary multichannel time-series data [52]

Data-augmentation for time-series data: Standard data-augmentation techniques for sequential data that are normally used in the field of NLP can be summarized as either replacing a token with its synonym [53] or by generating new data samples with back-translation [54]. However, such data-augmentation techniques cannot be transferred to the general time-series data signals.

The basic time-series data augmentation techniques can be categorized into time-domain, frequency domain and hybrid approaches as surveyed in [55]. In this study, we will focus on the time-domain data augmentation techniques. A speeding up (up-sampling) and slowing down (down-sampling) of the time-series signal in various UCR [56] datasets, termed as window warping has been proposed in [57]. A window slicing technique wherein a larger time-series signal is splitted into multiple smaller signals and a moving average smoothing with different window sizes has been presented in [58] for time-series classification with multi-scale CNN. [59] introduces multiple augmentations for wearable sensor data including jittering, scaling, rotating, permutating, magnitude warping and time warping.

Authors in [60] propose a weighted version of the time series averaging method Dynamic Time Warping (DTW) Barycentric Averaging (DBA) for generating new samples. The authors propose three weighting methods to choose the weights to assign to the series of the dataset. Authors in [61] apply the same technique for training deep CNNs for time-series classification reporting a drastic increase in performance. It should be however, noted here that DBA is dataset extension technique which is not explicitly the goal of this study. We aim to achieve augmentation strategy to have uniform features that preserves maximal information and aligned features for similar examples [62]. Moreover, an on-the-fly dynamic realization the DBA technique is not possible since it is an iterative process. Therefore, we do not consider DBA and other dataset extension techniques. We also do not consider supervised data-augmentation technique where a label is required as presented in [63], since it also does not fit into our goal of self-supervised time-series classification. In addition to previously applied approaches, we propose further data augmentation techniques which are shown to improve performance in the contrastive learning setting.

Time-series analysis and deep learning: [64,65] provide an overview about different methods and approaches to time-series data whereas [66] expresses the enormous potential and capabilities of deep learning methods. [67] presents a detailed survey on datasets and deep learning methods used in research in the context of recommender systems and is suitable supplemented by [68,69].

The following Table 1 sums up all the mentioned references.

3. Self-supervised time-series analysis

In this section, we state the considered problem of self-supervised learning for time-series analysis and present the proposed approach to solve it.

Table 1
Summary of the related works.

Type	Reference
Supervised learning multi-class classification	[21–32]
Semi-supervised multi-class classification	[33–40]
Representation learning	[41–43]
Contrastive learning	[8,15,44–52]
Data-augmentation for time-series data	[53–63]
Time-series analysis and deep learning	[64–69]

3.1. Problem statement

Since there are various approaches for time-series analysis, some clarifications and assumptions have to be stated.

1. It is assumed that we have a dataset of time-series signals from arbitrary sources. These sources can generate either discrete or continuous values.
2. No feature extraction techniques have been used except for normalization. The raw time-series data is used as input for the learning algorithm.
3. No additional statistical test have been undertaken for evaluating the model performance.

Hence, given a multivariate a time series signal $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ where $\mathbf{x}_i \in \mathbb{R}^m$, with m denoting the number of variables and T the length of the time series signal. Then, self-supervised deep learning refers to the problem of learning a representative embeddings using some sort of deep neural network architecture and training procedure solely based on a loss function derived from the above given unlabeled time series signal and potentially data augmentation techniques.

The performance of the self supervised learning is most often measured on a downstream task, e.g. a classification problem. To this end, a certain amount of labels $\{y_1, y_2, \ldots, y_T\}$ is used for performance evaluation. Hence, the employed contrastive learning set-up operates as follows:

- Train a (fully convolutional) neural network $f(\cdot)$, to optimize a contrastive loss function in a self-supervised manner by means of data augmentation techniques.
- Keeping the neural network $f(\cdot)$ fixed, train a linear classifier $g(\cdot)$ on the latent representation of $f(\cdot)$ in a supervised learning setting with standard classification or regression loss functions.

3.2. SimCLR

SimCLR is a framework for contrastive learning proposed in [14] and intended for the usage in image classification. The proposed approach allowed them to reach state of the art performance on the ImageNet [70] Top 1 and Top 5 accuracy and to compete with fully supervised trained models, although they utilized a self-supervised training approach.

The SimCLR approach combines various data augmentations techniques with contrastive learning. As illustrated in Fig. 1, starting from an input image $\mathbf{X}$, two representations $\tilde{\mathbf{X}}_i$ and $\tilde{\mathbf{X}}_j$ of $\mathbf{X}$ are created by applying randomly selected augmentation techniques. The augmented data is subsequently encoded in the representations $\mathbf{h}_i$ and $\mathbf{h}_j$ using a shared CNN architecture $f(\cdot)$, yielding a two headed approach. The SimCLR study uses a ResNet-50 [71] implementation as $f(\cdot)$. $\mathbf{h}_i$ and $\mathbf{h}_j$ are further processed with an additional non-linearity $g(\cdot)$ in the form of a two layer Multi Layer Perceptron (MLP) to form the embeddings $\mathbf{z}_i$ and $\mathbf{z}_j$. These embeddings are then finally used for the self-supervised contrastive learning which can be formally explained as follows:

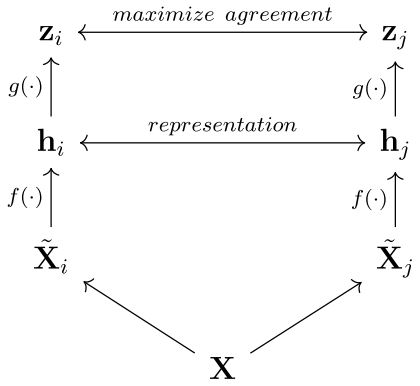


Fig. 1. Illustration of the contrastive learning process [14].

- Starting with a batch of N inputs, the augmentation leads to a total number of $2N$ training items, whereby $\tilde{\mathbf{X}}_i$ and $\tilde{\mathbf{X}}_j$ are defined as a positive pair and the other $2N-2$ items are considered as negative values, respectively.
- Then, for each positive pair $\tilde{\mathbf{X}}_i$ and $\tilde{\mathbf{X}}_j$, the similarity to the negative values after applying $f(\cdot)$ and $g(\cdot)$, is calculated using

$$l_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k\neq i} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \tag{1}$$

  where $\text{sim}(\cdot)$ represents the cosine similarity and $\mathbb{1}_{k\neq i} \in [0, 1]$ corresponds to 1 if $k \neq i$, otherwise 0. The loss is further tunable with an additional temperature parameter τ. The final overall loss is then composed of all positive pairs (i, j) as well as (j, i).
- After the contrastive training, the additional non-linearity $g(\cdot)$ is removed and only the encoding of $f(\cdot)$ is used for the downstream task, i.e. the classification. To this end, a simple linear layer or another MLP, respectively, is trained to classify based on the output encoding of the pre-trained contrastive model $f(\cdot)$ [14].

3.3. Contrastive Learning for Time-Series - SimCLR-TS

Our approach is based on the Sim-CLR method, however there are certain subtle differences as we consider time series data instead of images. First, for $f(\cdot)$, instead of a ResNet-50 architecture, we use a network suitable for time-series data. Specifically, we use 1D convolutions as described in the next section. Second, we refrain from using the additional non-linearity $g(\cdot)$ during contrastive training since we experienced no benefit by using the additional network and hence, we avoid unnecessary computation time and memory consumption. Therefore, we directly train to maximize agreement on the latent representations h_i and h_j yielding a slightly adapted loss function

$$l_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{h}_i, \mathbf{h}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k\neq i} \exp(\text{sim}(\mathbf{h}_i, \mathbf{h}_k)/\tau)}. \tag{2}$$

An overview of the SimCLR-TS structure is illustrated in Fig. 2. Third, multiple augmentations used in Sim-CLR are applicable to image data only. Hence, we propose a novel set of data augmentations particularly designed for the time-series character of the data.

After pre-training this model, the respective learned representation h_i, produced with only one head and without augmentation, is fed into either a linear layer or a MLP which serves as a classifier $g(\cdot)$.

The overall process is further expressed in algorithm 1:

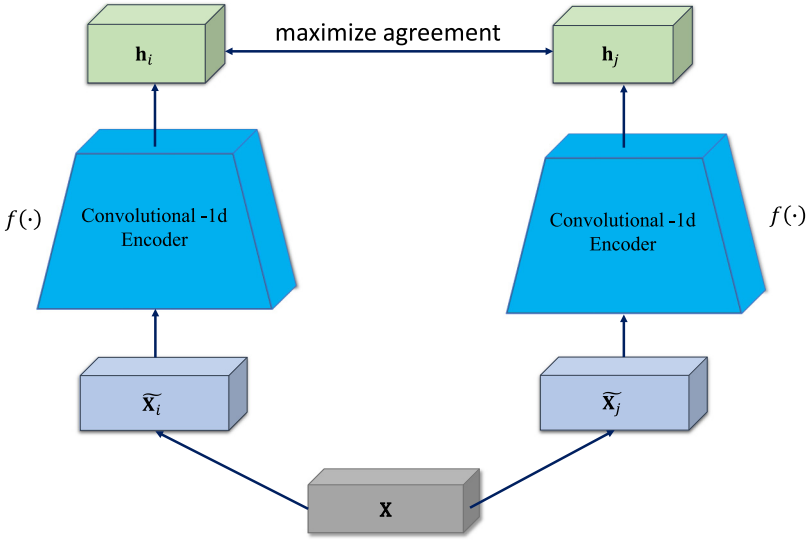


Fig. 2. SimCLR-TS structure.

Algorithm 1: SimCLR-TS Algorithm

Input: labeled or partially labeled Dataset $\mathcal{D} = \mathcal{D}_{labeled} \cup \mathcal{D}_{unlabeled}$, Set of Augmentations $\mathcal{A}$, models $f_\psi(\cdot), g_\phi(\cdot)$, contrastive loss $l(\cdot)$, cross-entropy loss $XE(\cdot)$

for *epoch in* $epochs_{cl}$ **do**
  for *batch in* $\mathcal{D}$ **do**
    $\tilde{x}_i, \tilde{x}_j = augment(batch),\ augment(\cdot) \in \mathcal{A}$
    $h_i = f_\psi(\tilde{x}_i)$
    $h_j = f_\psi(\tilde{x}_j)$
    $\psi = \psi + \nabla l(h_i, h_j)$
  end
end
$f(\cdot)$ =freeze_weights($f_\psi(\cdot)$)
for *epoch in* $epochs_{classifier}$ **do**
  for *batch. label in* $\mathcal{D}_{labeled}$ **do**
    $y_i = g_\phi(f(batch))$
    $\phi = \phi + \nabla XE(y_i, label_i)$
  end
end

3.4. CNN-1D encoder

We propose to employ CNN-1D encoder networks for time-series analysis. The rational for 1D CNN lies in the lack of spatial relation within the sensor channels which renders the standard 2D convolutions in CNNs unreasonable. At the same time, CNNs have been shown to provide superior performance also for time series data analytics. The 1D convolution operation is performed over a part of the complete input space, which is referred to as the receptive field. We denote the receptive field of size $n_r \times m$, which strides over $T \times m$ sequences, accounting for each of the variables. The pth convolution 1D kernel in the first layer can be denoted with a 2-dimensional tensor $K^{(p)} = [k_{i,j}^{(p)}] \in \mathbb{R}^{n_r \times m}$. The indices i, j denote the dimension along the time and variable axis respectively. The outputs or feature maps extracted from the convolution operation with 1 convolution kernel is a 1-dimensional tensor $H = [h_i]$. Usually multiple convolution kernels are used in each convolution layer leading multiple feature maps which subsequently make the feature maps a 2-dimensional tensor $H = [h_{i,p}]$. Each convolution kernel is responsible for extracting different features from the input data. Formally, the convolution 1D operation can be summarized as follows:

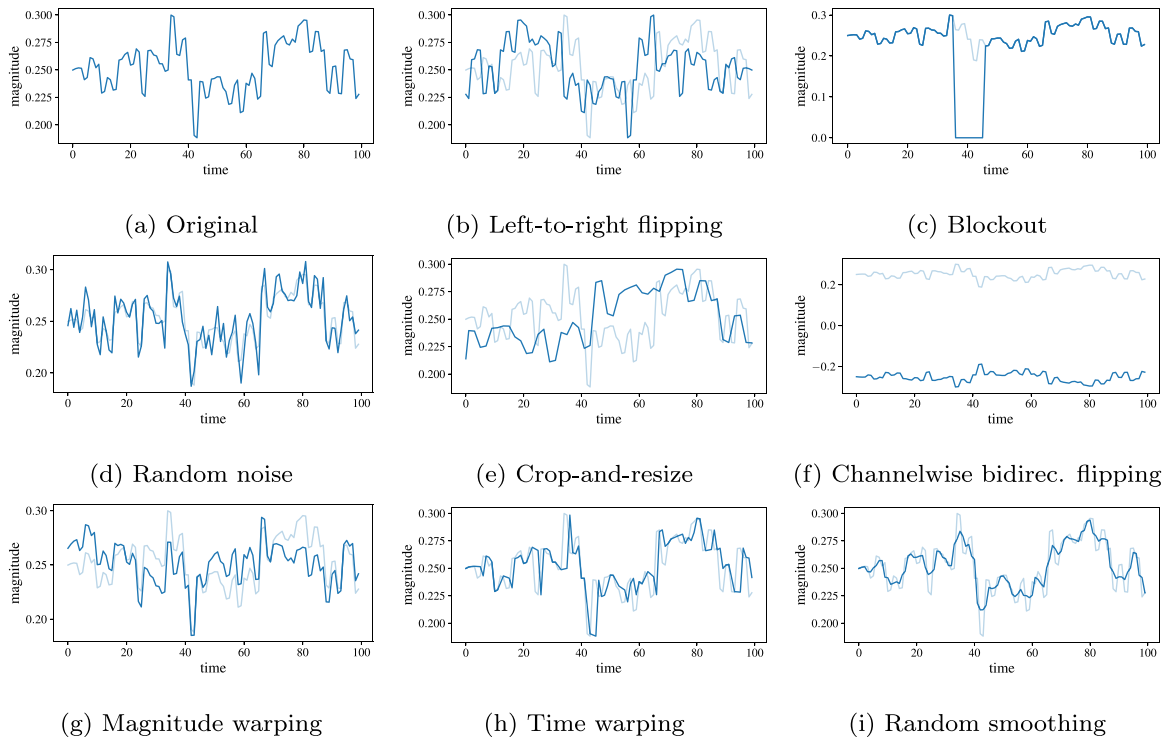
$$h_{i,p} = (x * k)_i = \sum_{g=1}^{n_r} \sum_{f=1}^{m} x_{i+g-1,f} \cdot k_{g,f}^{p}$$
$$\forall i \in \{1, \ldots, T - n_r + 1\}$$
$$\forall p \in \{1, \ldots, d_{q+1}\}, \quad (3)$$

where $h_{i,p}$ denotes the output of the $(i)^{th}$ receptive field and the pth convolution kernel, $x_{i+g-1,f}$ are the elements in the receptive field of the input variable, $k_{g,f}$ is the convolution kernel and d_{q+1} denotes the number of convolution kernels in the given layer.

3.5. Data augmentation

In this section, we derive various data augmentations for time series data which we use in our experiments. Although [14] augments image-data, some of the augmentations can be also transferred to the time-series data domain. Particularly, we can define Cutout or Blockout as well as Noise also for time series data. We also derive a new variant of crop and resize specifically designed for the time-series character of the data. As novel data augmentation techniques, we propose Left-to-right flipping and Bidirectional flipping. We further propose Magnitude Warping, comparable to [59], however using sine-waves, and Time Warping, i.e. stretching and squeezing the time-series data with a randomly sampled rectangular pattern. Finally, we propose Random smoothing using a Finite Impulse Response (FIR) filter based on randomly sampled parameters, to keep the overall trend and shape of the time-series data but to achieve varying representations. This is in contrast to smoothing through applying a fixed moving average used for de-noising as proposed in [58].

In the following section we present each of the data augmentation strategies in detail. An example of the proposed augmentations applied on a single channel is shown in Fig. 3.



(a) Original (b) Left-to-right flipping (c) Blockout

(d) Random noise (e) Crop-and-resize (f) Channelwise bidirec. flipping

(g) Magnitude warping (h) Time warping (i) Random smoothing

Fig. 3. Example of the used data augmentations. Dark blue corresponds to the augmented data and the lighter blue indicates the original data. Note that bidirectional flipping and random permutation is omitted here as this only makes sense when considering or showing multiple channels.

3.5.1. Left-to-right flipping

Left-to-right flipping can be seen as a rotation of the data around the y-axis (channel-axis), resulting in the most right sensor values becoming the most left and vice-versa. In matrix representation, this can be written as

$$\tilde{\mathbf{X}} = \mathbf{X}\Upsilon \text{ where}$$
$$\Upsilon_{i,j} = \begin{cases} 1, & \text{if}(j = T - i + 1) \\ 0, & \text{if}(j \neq T - i + 1) \end{cases} \tag{4}$$

with $\mathbf{X}$ denoting the original time-series data, $\tilde{\mathbf{X}}$ the augmented data and $\Upsilon \in \mathbb{R}^{T \times T}$ is an antidiagonal identity matrix. Particularly, this augmentation shall make the 1D CNN filter invariant to permutations of the sensor channels.

3.5.2. Bidirectional flipping

Bidirectional flipping is comparable to left-to-right flipping with the distinction that this time the data is mirrored on the x-axis (time-axis). This formulates to

$$\tilde{\mathbf{X}} = \Gamma\mathbf{X} \text{ where}$$
$$\Gamma_{i,j} = \begin{cases} 1, & \text{if}(j = m - i + 1) \\ 0, & \text{if}(j \neq m - i + 1) \end{cases} \tag{5}$$

with $\mathbf{X}$ denoting the original time-series data, $\tilde{\mathbf{X}}$ the augmented data and $\Gamma \in \mathbb{R}^{m \times m}$ is an antidiagonal identity matrix, sometimes referred to as an exchange matrix.

Furthermore, instead of a global flipping, this can be also done channelwise. In particular, this results in simply inverting each channel by multiplying with -1:

$$\tilde{\mathbf{X}} = -\mathbf{X} \tag{6}$$

3.5.3. Random channel permutation

As the name already says, for the random channel permutation we bring the channels in a randomly selected new order, the time-dimension remains unchanged during this augmentation. This yields

$$\tilde{\mathbf{X}} = \Psi\mathbf{X} \text{ where}$$
$$\Psi_{i,j} = \begin{cases} 1 & \text{if } i = \boldsymbol{\psi}(j) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

with $\boldsymbol{\psi}$ being a randomly permuted version of the vector $[1, \ldots, m]$ and Ψ the resulting permutation matrix. Random channel permutation can be seen as the generalization of bidirectional flipping or the other way around bidirectional flipping as a special case of random channel permutation.

3.5.4. Blockout

For the blockout augmentation, we randomly select an area of $\lambda \cdot m \in \mathbb{N}$ neighboring elements inside the time-series signal and set all values to zero. Starting in row k and column l, then

$$\tilde{x}_{i,j} = \begin{cases} 0 & \text{if } k \leq i \leq k + \lambda m \text{ and } l \leq j \leq l + \lambda m \\ x_{i,j} & \text{otherwise} \end{cases} \tag{8}$$

3.5.5. Random noise

To take care of the time-series character of the data, we apply noise individually per channel i based on the standard deviation σ_i of the values. Therefore, we use a matrix $\mathbf{R}$ of the size of the data with random sampled values between -1 and 1, multiply each channel with the per-channel standard deviation σ_i and multiply an additional factor λ for further tuning possibility of the noise level. This yields to

$$\tilde{\mathbf{X}} = \mathbf{X} + \mathbf{R} \circ \lambda\boldsymbol{\Sigma},$$
$$\boldsymbol{\Sigma} = \{\Sigma, \ldots, \Sigma\} \in \mathbb{R}^{m \times T} \tag{9}$$

where $\mathbf{X}$ denotes the original data and $\tilde{\mathbf{X}}$ the augmented data, Σ a column-vector composed of the per-channel standard deviations σ_i and $\circ$ and element-wise multiplication.

3.5.6. Crop and resize

Since we are working with time-series data, the cropping and resizing is only done in the time-dimension of the data. For this, we do a linear interpolation between the successive values, resulting in $T-1$ intermediate values and a new image with the size $m \times (2T-1)$. Then we randomly select a starting column k within the first half of the time-series signal and the following $T-1$ columns to form the cropped and resized data. Interpolating between channels makes no sense here as this usually mixes two physically completely quantities and data which is not related to each other. The described procedure can also be expressed as

$$\tilde{x}_{i,j} = \begin{cases} x_{i,k+\frac{j}{2}-0.5} & \text{if } j \text{ odd} \\ \frac{x_{i,k+\frac{j}{2}}+x_{i,k+\frac{j}{2}+1}}{2} & \text{otherwise.} \end{cases} \tag{10}$$

3.5.7. Magnitude warping

Magnitude warping changes the amplitude of the signal channel by slightly increasing it in some areas and decreasing it in others. Specifically we use a sine wave matrix Ψ with an amplitude of λ added with one and n full sine periods to scale the signal. Each sine wave for one channel $i \in [1, \ldots, m]$ is further phase shifted with an individual, randomly selected value θ_i. The whole process can be expressed as

$$\begin{aligned} &\tilde{\mathbf{X}} = \mathbf{X} \circ \lambda\Psi, \\ &\Psi = 1^{m\times T} + \sin\left(\begin{bmatrix} \frac{2\pi nt}{T} + \theta_1 \\ \vdots \\ \frac{2\pi nt}{T} + \theta_m \end{bmatrix}\right) \text{where} \\ &t \in [0, \ldots, T],\ \theta_i \in [0, \ldots, 2\pi]. \end{aligned} \tag{11}$$

3.5.8. Time warping

Time warping resembles the idea of crop and resize, but instead of creating a cutout of the data, it gets stretched and compressed. Compression in this case means simply discard some data items and stretching utilizes the same interpolation mechanism as crop and resize. The spots for the stretching and compressing are randomly sampled in equal numbers to maintain the overall shape of the data. Specifically, we randomly sample a vector of length T, consisting of the values $-1, 0, 1$. Here, -1 represents discarding the elements, 0 represents no action and 1 represents interpolating. To scale the strength of the augmentation, we fill this vector with 10%, 20%, 30%, 40% or 50% ones and minus ones respectively. This process can also be described by the following algorithm:

Algorithm 2: Time warping

Result: Augmented data $\tilde{\mathbf{X}}$ out of the initial data $\mathbf{X}$

```
X̃ = zeros(m, T)
ctr = 0
for i in range(T) do
    if pattern[i] == 0 then
        X̃[:, ctr] = X[:, i]
        ctr += 1
    else if pattern[i] == 1 then
        X̃[:, ctr] = X[:, i]
        X̃[:, ctr + 1] = (X[:, i] + X[:, i+1])/2
        ctr += 2
    else
        continue
    end
end
return X̃
```

3.5.9. Random smoothing

For smoothing of the time-series signal trajectories, we use a FIR filter [72]. We randomize the smoothing capability to increase the variation by randomly sampling a value $\lambda \in [0; 1]$ and use the transfer function:

$$H(z) = \frac{\lambda}{2}z^1 + (1-\lambda) + \frac{\lambda}{2}z^{-1}. \tag{12}$$

3.6. Training strategies/augmentation strategies

For the contrastive training, several strategies for the usage of the augmentations with the two-headed architecture are conceivable. Basically, a distinction can be made between two variants: augmenting only on one head while the other head performs a simple identity function or using augmented data on both heads. The former has a stronger connection to the original data, which is what is later used for the fault classification, whereas the latter offers more variety in the training data. We implement and test both variants in our experiments.

Furthermore, a combination of augmentations is possible. For example either using a fixed augmentation followed by a second, randomly chosen augmentation, or two augmentations which are applied based on individual probabilities. This however leads to a intractable large search space for different augmentation combinations, which is why we focus on data which is augmented only once throughout this work.

4. Experimental results

In this section, we first describe the setup for our experiments, followed by the experimental results. Specifically, we define a baseline for comparison throughout the experiments, compare and analyze our proposed augmentation methods, provide a comparison among the different training strategies to achieve the overall best classification accuracy and finally train the classifier with only a subset or a fraction of the training data. We end this section with a detailed comparison of our results with the literature.

4.1. Experimental setup

For an experimental evaluation of the proposed methodology we use the TEP dataset [73]. This dataset consists of 21 fault cases and an additional case 0 for normal operation whereby each of the 22 cases consists of 52 sensor channels with sampled time series data. It is divided into training and testing data with 480 items per channel in the training data and 960 items in the test data. An exception is the regular operation case 0 where the training data consists of 500 items. Furthermore, the first 160 items of the test data for cases 1–21 correspond to the normal operation, the respective error is introduced after this samples. [74] The dataset can be downloaded here[1]:

For our experiments, we use the Pytorch [75] machine-learning framework. To form the time series input signal, we extract T consecutive values of each channel resulting in an input signal of size $52 \times T$. In the following we refer to T as the window length. Furthermore, we use a sliding window, meaning the next image starts at data sample $s+1$ and not $s+T$. This yields $480-T$ training items for the cases 1–21 and $500-T$ for case 0. Since the

[1] http://web.mit.edu/braatzgroup/links.html.

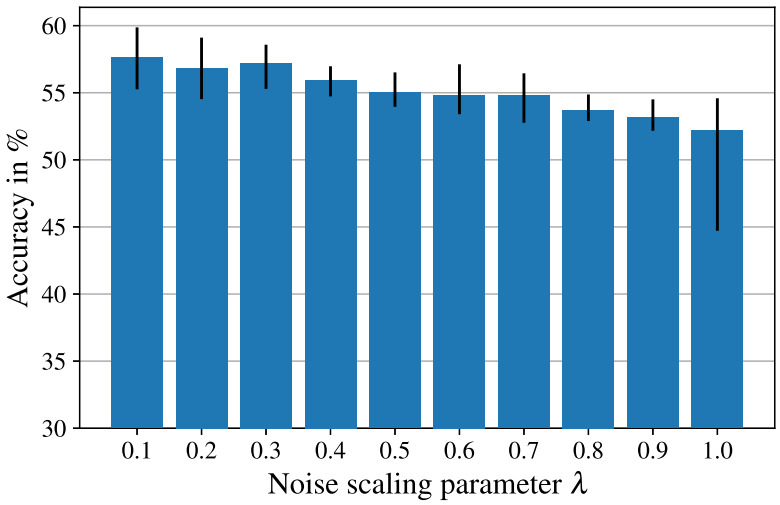


Fig. 4. Effect of varying the augmentation strength on random noise on the classification accuracy. It can be observed that the behavior is robust against changes in augmentation strength.

testing data of cases 1–21 also consists of samples corresponding to the regular operation, this has to be taken care of. Hence, we start at sample $s = 161$ for the creation of the testing images, yielding $800 - T$ test images. For case 0 this is not necessary, therefore this results in $960-T$ images. Furthermore, this number could be increased with the $(160 - T) \times 21$ additional images which can be created from the cases 1–21. Since this would create a huge imbalance between case 0 and the other cases, we refrain from this procedure.

During this work we concentrate on the augmentations and their effect on the contrastive learning, hence we use a fixed architecture for the contrastive model throughout all the experiments. This architecture is the following:

- Three layers of a Conv1d, LeakyReLU [76] and BatchNormalization-1d [77] combination
- A kernel size of 3 in all convolution layers
- A stride of 2 in the first two convolution layers and a stride of 1 in the last convolution layer
- The number of convolution channels doubling with each layer, starting with 64 channels
- The ϵ and momentum parameters in the BatchNorm1d layers are set as 1×10^{-5} and 0.1 respectively.

When using the pre-trained contrastive model for the final supervised evaluation, we follow the standard linear evaluation protocol, i.e. we use a linear layer with inputs according to the flattened output of the contrastive model and 22 classification neurons.

The training of the contrastive model is done for 400 epochs and the linear evaluation layer is trained for 4 epochs with the restriction that we stop the contrastive training earlier if the test-loss rises over a longer period. In both training processes the ADAM optimizer [78] and a batch-size of 64 is used. Unless otherwise stated, we tune on the learning rate for both, the contrastive-learning and the linear evaluation, and the weight-decay rate of the ADAM optimizer as a regularization mechanism. We use a window length $T = 100$ throughout this work.

During our experiments, we calculate several performance metrics. Specifically, these are the overall micro-F1 score and the class-wise F1 score [79], calculated with scikit-learn [80], the overall accuracy which is the relation of correct classified samples to overall samples and furthermore a class-wise accuracy or correct classification rate which corresponds to the ratio of correct classification to labels per class. These are used for comparisons throughout our experiments and also for comparison with several other classification approaches.

Table 2
Achieved accuracies with only one augmented head.

Augmentation	Accuracy
none	55.23%
Left-to-right flipping	71.59%
Bidirectional flipping	43.12%
Bidirectional flipping channelwise	43.17%
Blockout	32.07%
Noise	61.48%
Crop-and-resize	69.74%
Permute channels	20.17%
Magnitude warping	27.05%
Time warping	56.62%
Random smoothing	56.00%

4.2. Baseline supervised classification

The baseline model is the same as the model used for contrastive learning, but trained in fully supervised manner. Specifically, three Conv1d-LeakyRelU-BatchNorm1d combinations with a linear layer reducing the flattened output to the desired 22 classes. The training was done for 300 epochs with the ADAM optimizer, weight-decay regularization and a batch-size of 64.

With this classification approach, a baseline accuracy of 48.10% and a micro F1-score of 0.4727 could be achieved. Note that the results were only tuned on the used learning rate and weight-decay, and not on model architecture to keep the number of parameters for the evaluation protocol comparable.

4.3. Augmentation comparison and analysis

We experimented with the training strategy in a way such that the most promising augmentations out of the introduced function pool can be identified and to rule out the least promising ones. In particular, we use one fixed augmentation on the one head while the other head performs an identity function throughout the whole training loop resulting in the accuracies shown in Table 2.

As a result, augmentations like channel permutation, magnitude warping and blockout can be discarded since they perform even worse than the baseline and hence do not aid the training process but instead hinder it as they perform worse than without

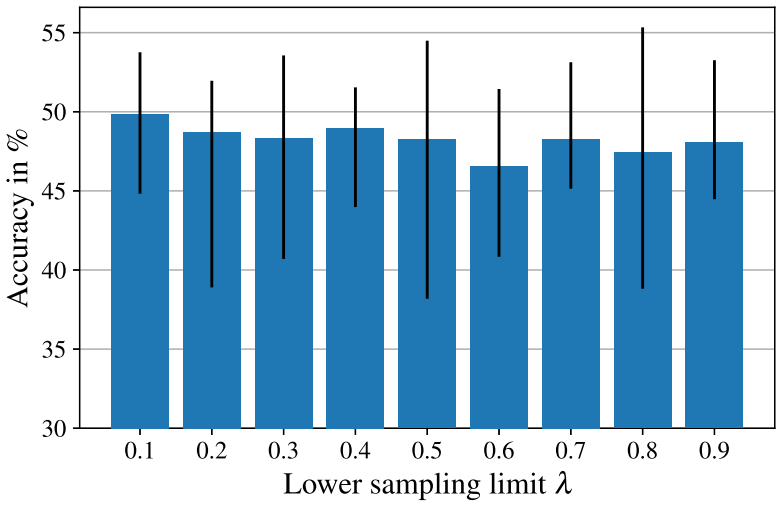


Fig. 5. Effect of varying the augmentation strength on random smoothing on the classification accuracy. Again, a robustness against changing the augmentation strength is observable.

applying any augmentation at all. Further, the bidirectional flipping approaches perform comparably worse and are discarded. Contrarily, left-to-right flipping followed by crop-and-resize and noise appear to be the frontrunner approaches and exceed the baseline by a significant amount, but also time warping and random smoothing performs comparably good and yield at least a slight increasement to applying no augmentation.

To further investigate the effect of the augmentation parameters and strength, we conduct a hyperparameter sweep corresponding to the results from Table 2 by testing over a parameter space on two selected examples. We chose Random Noise and Random Smoothing, the results are visualized in Figs. 4 and 5 where the bar heights indicate the mean classification accuracies throughout multiple runs and the vertical line the obtained min- and max-values. The x-axis corresponds to the augmentation strengths with increasing strength from the left to the right.

It can be observed that both augmentations seem to be relatively unaffected by the selection of the tuning parameter in a certain range, yielding results that appear to be robust about a parameter choice. Specifically, too high noise ratios tend to worsen results while sampling limit for random smoothing leaves the results mostly unaffected. Consequently, we conclude that tuning these parameters should not be a top priority during the general tuning process.

4.4. Random augmentation

In this section, we present the results from the training setup where randomly chosen augmentations are used. Particularly we use the four best working augmentations from the previous section: left-to-right flipping, random noise, crop and resize and random smoothing in three different combinations. Furthermore, we augment on one head as well as on both heads as discussed in Section 3.6. Table 3 shows the respective achieved best results. The only exception is the two-head augmentation using only left-to-right flipping and crop and resize: here we use the former on the first head and the latter on the second one. The randomness is introduced by the selected window in crop and resize.

With this setup, we could achieve a maximum classification accuracy of 80.24% and a micro F1-score of 0.8024 over all 22 cases of the TEP on the linear evaluation protocol. The trajectory of the loss and accuracy of the best classification training is shown in Fig. 6. In addition, the class-wise accuracies and F1-scores are displayed and compared with our baseline in Table 4, where the overall results for the F1-score indicates the micro F1-score.

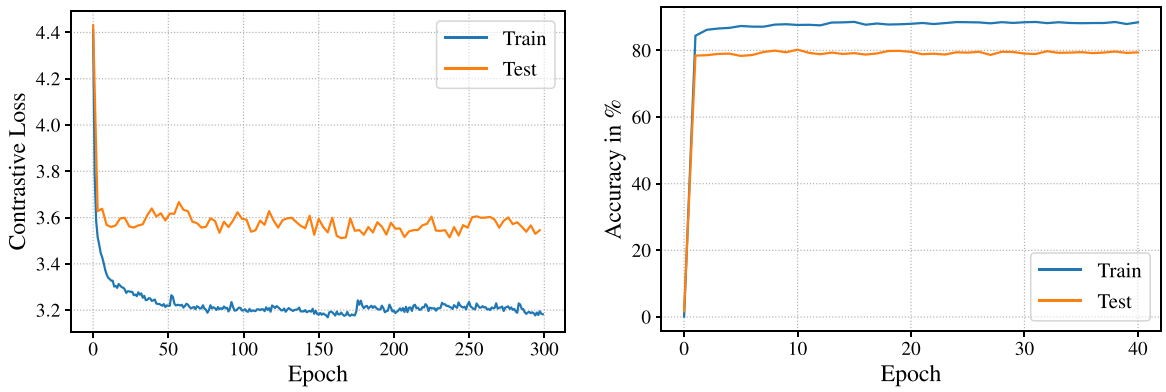
Table 3
Results when using different augmentation combinations and either one or two head augmentation.

Used augmentations	One head	Two heads
Left-to-right flipping + crop and resize	75.03%	80.24%
Left-to-right flipping + random noise + crop and resize	73.74%	73.33%
Left-to-right flipping + time warping + crop and resize	73.53%	75.46%
Left-to-right flipping + random smoothing + crop and resize	69.83%	73.75%

Table 4
Achieved accuracies and F1-scores per class and overall.

	Baseline		SimCLR-TS	
Case	F1-score	Accuracy	F1-score	Accuracy
Case 0	0.1102	10.93	0.3866	46.86
Case 1	0.9777	100.0	0.9857	98.43
Case 2	0.9825	100.0	0.9866	100.0
Case 3	0.1335	12.00	0.3375	40.43
Case 4	0.6949	65.71	0.9758	97.86
Case 5	0.3688	37.86	0.8866	81.00
Case 6	0.9928	98.57	0.9746	95.86
Case 7	0.9894	100.0	0.9986	99.86
Case 8	0.5003	51.86	0.7576	88.43
Case 9	0.1033	11.71	0.1502	12.14
Case 10	0.2080	18.86	0.9115	89.71
Case 11	0.0742	5.14	0.9922	99.57
Case 12	0.5104	51.00	0.9640	95.57
Case 13	0.4277	29.57	0.6463	54.29
Case 14	0.0037	0.29	0.9979	99.71
Case 15	0.1436	14.71	0.2139	19.14
Case 16	0.3110	38.86	0.8919	98.43
Case 17	0.8197	84.43	0.9993	100.0
Case 18	0.8027	77.00	0.9641	94.00
Case 19	0.3377	44.14	0.9550	100.0
Case 20	0.9676	98.14	0.9884	97.71
Case 21	0.0224	2.86	0.7740	63.86
Overall	0.4727	48.10	0.8024	80.24

As illustrated in the accuracy graph in Fig. 6(b), very few training epochs for the linear evaluation layer are necessary to reach the maximum achievable accuracy. This indicates that the output produced by the contrastive learning model yields a nearly linear separable problem and hence further confirms the approach of a pre-training to enhance differences in a latent representation.



(a) Contrastive pre-training loss trajectory (b) Linear evaluation accuracy trajectory

Fig. 6. Graphs of the minimized contrastive loss and maximized linear evaluation accuracy. (a) shows the effect of the pre-training emphasizing the similarity on the positive pairs and dissimilarity to the negative pairs whereas (b) corresponds to the downstream task where a classifier is trained on the representation produced by the pre-trained and frozen encoder.

Also, by comparing different classification accuracies and the corresponding confusion matrices we observed that a high accuracy in most cases and therefore a good overall result usually came at the expense of accuracy in separating the cases 0, 3, 9, 15 and 21 which is in accordance with previously reported results.

4.5. Label fraction training

In this section, we report the classification accuracies when only a fraction of the labeled training data is used for linear evaluation of the pre-trained contrastive model. This casts a real world scenario where only partially labeled data is available while fully labeling the dataset might be too time-consuming or not possible at all. Additionally, we use this scenario to compare the classification using a linear layer with a more capable 2-layer MLP.

We chose the fractions of 5%, 10% and 50%, in which we simply take every 20th, 10th or 2nd item of the training dataset to form the reduced one. We use the same pre-trained contrastive model and the same training hyperparameter throughout all experiments and run each setup 10 times. The used batch size during training remains 64. This yields the achieved accuracies visualized in Fig. 7. We remark that in contrast to the usual experimental setup, the linear evaluation was done with 200 training episodes in comparison to the 40 episodes used earlier to make up for the loss of optimization steps due to the smaller amount of batches in the reduced dataset. This poses no drawbacks as the training of the single linear layer is not nearly as computational expensive as the contrastive learning. For the contrastive model, we use the best performing one from the previous section and the corresponding learning-rate for the classifier training. We also trained the classifier from our baseline-experiments with the same training data fractions for further comparison.

The obtained results from these experiments are shown in Fig. 7. Again, the bar-height indicate the mean values throughout multiple runs and the vertical line the obtained min- and max-values. Furthermore, the additionally stated accuracy also corresponds to the max-values.

As visible in Fig. 7, the pre-training using the contrastive learning approach has the most impact on the overall classification performance, as only a very small subset of the training data used in the following second training process results in a very high accuracy, which is not considerably inferior to the training with the complete dataset. The impact of only using fractions on the classification accuracy is also a way higher on the comparative values produced utilizing a fully supervised training. Furthermore, the usage of a MLP yields only a minor improvement to a simple linear layer. We see both as a further indication that the contrastive pre-training yields an almost linear separable latent representation. The capabilities of the MLP to be able to solve non-linear separable problems only enable marginal improvements and hence for the vast majority of fault cases it is not necessary. Instead, this shows the quality of the representation gained by the contrastive pre-training. Moreover, the ability to train the classifier with only a fraction of the data makes the proposed approach suitable also for only partially labeled datasets or reduces the necessity to fully label a newly created dataset.

Interestingly, the best overall accuracy, namely 80.37% for the linear layer and 81.43% for the MLP, were achieved by only using 50% of the training data even though the mean is highest for using all training data.

4.6. Literature comparison

Despite being a often used benchmark dataset, TEP bears two problems concerning the comparability of the obtained results: The first one is that a majority of comparable works is based on binary classification. This usually results in better results as it breaks a big problem into many small sub-problems which are easier to solve. This is however, not applicable to our approach as it rules out the training with only partially labeled data, which is an important aspect of our work. For the fully binary approach, the data needs to be split into 21 sub-groups including the normal operation case 0 and the respective error case 1–21. Assuming only e.G. 10% of the labels would be known, this could not be done since for 90% of the data it cannot be said to which sub-group they belong. Only a mixing of multi-class pretraining and binary classification would be possible, but we want to avoid this mixing.

The second one is that some error cases are often omitted, likewise hurting the comparability since the overall problem itself is simplified because fewer classes or cases need to be distinguished. Most of the time the most difficult cases 3, 9 and 15 are ignored which significantly boosts the overall classification accuracy. Hence, in the following we adapt our obtained results if necessary for best possible comparability.

Table 5 gives an overview of several classification approaches with their achieved per-class accuracy. Here we provide the results from the label fraction training with 50% of training data and using a MLP as they were our best overall results. For better comparability we also added a second set of accuracies, achieved with

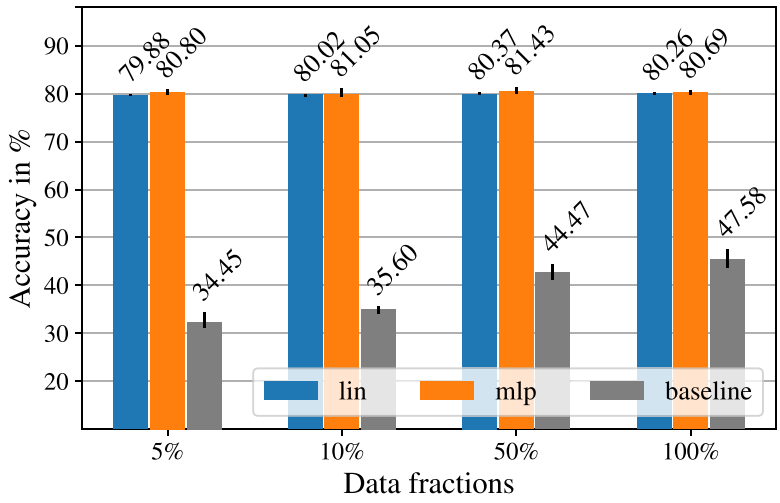


Fig. 7. Achieved classification accuracies from the experiments where only fractions of the overall training data was used to train the three classifiers. The linear layer and the MLP used the pre-trained and frozen encoder whereas the baseline results were obtained utilizing a fully-supervised training.

Table 5
Comparison of the achieved accuracies with other approaches.

	All cases			Without 3, 9, 15		Other		
Case	SVM [27]	PCA [27]	SimCLR-TS	SSAE [31]	SimCLR- TS	MCNN-LSTM [81]	Enhanced RF [32]	MC1-DCNN [29]
Case 0	19.27	19.69	**55.12**	96.48	**98.37**	93.5	–	–
Case 1	88.44	88.54	**96.29**	**100.0**	99.43	99.9	99	100.0
Case 2	86.04	89.06	**100.0**	**100.0**	99.86	99.3	98	100.0
Case 3	15.73	21.25	**25.57**	–	–	–	35	83.48
Case 4	58.02	81.35	**96.57**	**100.0**	87.71	100	97	99.22
Case 5	64.79	**87.81**	83.86	100.0	79.71	–	100	90.40
Case 6	71.88	89.58	**99.14**	**100.0**	98.86	–	100	93.10
Case 7	88.13	88.75	**100.0**	100.0	100.0	–	100	100.0
Case 8	45.10	83.96	**86.71**	**96.32**	95.29	98.8	76	99.27
Case 9	12.92	22.60	**29.43**	–	–	–	23	74.60
Case 10	27.08	76.98	**91.71**	62.16	**97.14**	99.9	81	93.75
Case 11	14.90	70.21	**99.29**	95.93	**98.57**	–	76	95.45
Case 12	52.40	87.08	**95.14**	99.87	**100.0**	97.0	89	100.0
Case 13	35.10	**69.58**	57.86	**89.88**	71.29	98.2	30	99.14
Case 14	62.19	88.23	**99.57**	100.0	100.0	100	100	100.0
Case 15	22.19	**26.98**	02.71	–	–	–	28	73.50
Case 16	16.77	73.65	**98.57**	54.66	**99.57**	–	–	100.0
Case 17	53.65	75.94	**100.0**	**100.0**	99.86	100	–	100.0
Case 18	30.94	73.54	**99.00**	**93.56**	47.29	99.7	–	98.43
Case 19	51.25	85.73	**100.0**	79.37	**99.57**	–	–	100.0
Case 20	44.90	79.69	**99.14**	92.51	**100.0**	–	–	94.12
Case 21	8.65	**85.63**	81.71	78.06	**93.29**	–	6	70.59
Overall	44.11[a]	71.17[a]	**81.43**	91.52	**93.00**	98.8[a]	71.46	94.08

[a]The original work does not state this value, hence it is self calculated.

our approach when omitting the cases 3, 9 and 15. Furthermore, we convert to percentages if necessary and round the accuracies to two decimal places if the original source allows that.

The work in [27] does a classification on all 22 cases of the TEP using SVM and PCA. For SVM, a one against one approach is chosen to serve for multi-classification. When comparing the class-wise accuracies with SimCLR-TS, it can be observed that our approach is significantly more accurate in almost all cases, the only exceptions are cases 5, 13, 15 and 21. Unfortunately, no overall accuracy is stated, hence we calculated the mean accuracy out of the given values. This yields an improvement of more than 37% over SVM and 10% of PCA.

[31] achieves an overall accuracy of 91.52%, but without the inclusion of the cases 3, 9 and 15. To do so, they use stacked supervised auto-encoder (SSAE). This is close to our achieved accuracy of 93.00%, while our final performance is still higher.

In [32], an enhanced random forest approach is used for fault classification in the TEP achieving an accuracy of 71.46%. However, they used only 16 fault cases and omitted the cases 16–20 as well as the case 0 for normal operation, yielding no direct comparability since we used all 22 cases. Despite having to classify six less classes, this approach results in an almost 10% lower accuracy.

For the sake of completeness we have also listed the results of [29] in Table 5 but we refrain from a further comparison as they used a fully supervised training and additional wavelet transformation which prevents a fair comparison with our semi-supervised approach using raw data. The same applies to the MCNN-LSTM approach from [81] as they only used a selected subset of the TEP dataset with classes which seem to be predictable very well. However, if comparing the class-wise accuracies, our results are not much worse than the MCNN-LSTM ones, despite resulting from a way more complex classification task.

We also compare results from using only fractions of the TEP data. [28] uses a combination of slow feature analysis and multiclass relevance vector machines for fault classification on TE. Specifically, they train their classifier with only 100 samples of the training data, yielding comparability with our experiments

Table 6
Classification accuracies achieved with only using fractions of the training dataset.

Case	SFA-RVM [28]	SimCLR-TS 10%
Case 0	–	**0.57**
Case 1	**0.98**	0.97
Case 2	0.93	**1.00**
Case 3	**0.47**	0.19
Case 4	0.87	**0.98**
Case 5	**0.97**	0.82
Case 6	**1.00**	0.93
Case 7	0.99	**1.00**
Case 8	0.75	**0.84**
Case 9	**0.25**	0.24
Case 10	0.68	**0.97**
Case 11	0.66	**0.99**
Case 12	0.77	**0.92**
Case 13	**0.75**	0.60
Case 14	0.96	**1.00**
Case 15	0.11	**0.15**
Case 16	0.77	**0.98**
Case 17	0.87	**1.00**
Case 18	0.88	**0.96**
Case 19	0.82	**1.00**
Case 20	0.76	**0.99**
Case 21	0.61	**0.78**
Overall	0.761	**0.8105**

where the classifier was only trained with fractions of the training data. For comparison, we use the results where we used 10% of the training dataset and a MLP for classification, which corresponds to 38 time-series windows in classes $1-21$ and 40 time-series windows in class 0. This is a significantly smaller number of training items than the 100 items used in [28], but our approach still outcompetes the SFA-RVM by 81.05% to 76.1%. The class-wise results are further shown in Table 6. Moreover, [28] omitted the normal operation case 0, hence one class less needed to be classified.

5. Conclusion

We proposed a novel self-supervised learning approach for time-series data based on contrastive learning and data-augmentation techniques. This was supplemented by an investigation of the effectiveness of these data-augmentations for the used methodology. The overall approach was tested on the TEP benchmark dataset and achieved very good results, surpassing many other multi-classification approaches. Furthermore, our approach allows for a very accurate classification when training the classifier with only a subset of labeled data, indicating that this approach is also suitable for applications where only partially labeled time-series data is available. Likewise, it can lower the necessary work when labeling a dataset.

In the future, the proposed approach can be applied to other time-series datasets for fault classification to corroborate and confirm our findings. Furthermore, experiments and improvements on the used neural network architecture for the model trained with contrastive learning have the potential to further increase the classification accuracy on the TEP. Also, more advanced augmentation methods, e.g. using GAN-networks, can be tried and rated. Likewise, instead of augmenting all channels, the effect of only augmenting several channels or mixing augmentations on a channel-level and not sample-wise could be investigated. In the same way, the previously mentioned chain of augmentations can also be examined. Finally, the classification using a linear layer or MLP could instead be done with a SVM.

CRediT authorship contribution statement

Johannes Pöppelbaum: Conceptualization, Implementation, Validation, Resources, Writing – original draft, Writing – review and editing, Visualization. **Gavneet Singh Chadha:** Conceptualization, Methodology, Validation, Formal analysis, Resources, Writing – original draft, Visualization. **Andreas Schwung:** Conceptualization, Writing – original draft, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P. Sadowski, P. Baldi, Deep learning in the natural sciences: Applications to physics, in: Braverman Readings in Machine Learning, in: LNCS Sublibrary: SL7 - Artificial intelligence, Vol. 11100, Springer, Cham, Switzerland, 2018, pp. 269–297.
[2] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[3] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[4] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems 25, Curran Associates, Inc, 2012, pp. 1097–1105.
[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[6] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[7] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, 2018, arXiv preprint arXiv:1802.05365.
[8] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, J. Tang, Self-supervised learning: Generative or contrastive, 2020, ArXiv arXiv–2006.
[9] L. Jing, Y. Tian, Self-supervised visual feature learning with deep neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. (2020).
[10] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al., Conditional image generation with pixelcnn decoders, in: Advances in Neural Information Processing Systems, 2016, pp. 4790–4798.
[11] A. van den Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, in: Proceedings of The 33rd International Conference on International Conference on Machine Learning-Volume 48, 2016, pp. 1747–1756.
[12] Z. Chen, W. Li, Multisensor feature fusion for bearing fault diagnosis using sparse autoencoder and deep belief network, IEEE Trans. Instrument. Measur. 66 (7) (2017) 1693–1702.
[13] R. Thirukovalluru, S. Dixit, R.K. Sevakula, N.K. Verma, A. Salour, Generating feature sets for fault diagnosis using denoising stacked auto-encoder, in: 2016 IEEE International Conference on Prognostics and Health Management (ICPHM), 2016, pp. 1–7.
[14] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning, 2020, pp. 1597–1607.
[15] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of The IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.
[16] O. Henaff, Data-efficient image recognition with contrastive predictive coding, in: Proceedings of The 37th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, Vol. 119, PMLR, Virtual, 2020, pp. 4182–4192.
[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[18] C. Shorten, T.M. Khoshgoftaar, A survey on image data augmentation for deep learning, J. Big Data 6 (1) (2019) 60.
[19] L. Zhang, J. Lin, B. Liu, Z. Zhang, X. Yan, M. Wei, A review on deep learning applications in prognostics and health management, IEEE Access 7 (2019) 162415–162438.

[20] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, A.K. Nandi, Applications of machine learning to machine fault diagnosis: A review and roadmap, Mech. Syst. Signal Process. 138 (2020) 106587.
[21] J. Lei, C. Liu, D. Jiang, Fault diagnosis of wind turbine based on Long Short-term memory networks, Renew. Energy 133 (2019) 422–432.
[22] G.S. Chadha, A. Panambilly, A. Schwung, S.X. Ding, Bidirectional deep recurrent neural networks for process fault classification, ISA Trans. 106 (2020) 330–342.
[23] G. Jiang, H. He, J. Yan, P. Xie, Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox, IEEE Trans. Ind. Electron. 66 (4) (2019) 3196–3207.
[24] R. Liu, G. Meng, B. Yang, C. Sun, X. Chen, Dislocated time series convolutional neural architecture: An intelligent fault diagnosis approach for electric machine, IEEE Trans. Ind. Inf. 13 (2017) 1310–1320.
[25] L. Wen, X. Li, L. Gao, Y. Zhang, A new convolutional neural network-based data-driven fault diagnosis method, IEEE Trans. Ind. Electron. 65 (7) (2018) 5990–5998.
[26] M. Xia, T. Li, L. Xu, L. Liu, C.W. de Silva, Fault diagnosis for rotating machinery using multiple sensors and convolutional neural networks, IEEE/ASME Trans. Mechatron. 23 (1) (2018) 101–110.
[27] C. Jing, X. Gao, X. Zhu, S. Lang, Fault classification on Tennessee Eastman process: PCA and SVM, in: 2014 International Conference on Mechatronics and Control (ICMC), 2014, pp. 2194–2197.
[28] J. Huang, X. Yang, Y. Shardt, X. Yan, Fault classification in dynamic processes using multiclass relevance vector machine and slow feature analysis, IEEE Access 8 (2020) 9115–9123.
[29] J. Yu, C. Zhang, S. Wang, Multichannel one-dimensional convolutional neural network-based feature learning for fault diagnosis of industrial processes, Neural Comput. Appl. (2020).
[30] N. Wang, F. Yang, R. Zhang, F. Gao, Intelligent fault diagnosis for chemical processes using deep learning multimodel fusion, IEEE Trans. Cybernet. (2020) 1–15.
[31] Y. Wang, H. Yang, X. Yuan, Y.A. Shardt, C. Yang, W. Gui, Deep learning for fault-relevant feature extraction and fault classification with stacked supervised auto-encoder, J. Process Control 92 (2020) 79–89.
[32] Z. Chai, C. Zhao, Enhanced random forest with concurrent analysis of static and dynamic nodes for industrial fault classification: IEEE transactions on industrial informatics, 16(1), 54-66, IEEE Trans. Ind. Inf. 16 (1) (2020) 54–66.
[33] Z. Zhang, J. Zhao, A deep belief network based fault diagnosis model for complex chemical processes, Comput. Chem. Eng. 107 (2017) 395–407.
[34] J. Tao, Y. Liu, D. Yang, Bearing fault diagnosis based on deep belief network and multisensor information fusion, Shock Vib. 2016 (7) (2016) 1–9.
[35] Y. Lei, F. Jia, J. Lin, S. Xing, S.X. Ding, An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data, IEEE Trans. Ind. Electron. 63 (5) (2016) 3137–3147.
[36] W. Sun, S. Shao, R. Zhao, R. Yan, X. Zhang, X. Chen, A sparse auto-encoder-based deep neural network approach for induction motor faults classification, Measurement 89 (2016) 171–178.
[37] Y. Qi, C. Shen, D. Wang, J. Shi, X. Jiang, Z. Zhu, Stacked sparse autoencoder-based deep network for fault diagnosis of rotating machinery, IEEE Access 5 (2017) 15066–15079.
[38] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, C. Raffel, Mixmatch: A holistic approach to semi-supervised learning, in: Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.
[39] D. Berthelot, N. Carlini, E.D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, C. Raffel, ReMixMatch: Semi-supervised learning with distribution matching and augmentation anchoring, in: International Conference on Learning Representations, 2020.
[40] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C.A. Raffel, E.D. Cubuk, A. Kurakin, C.-L. Li, Fixmatch: Simplifying semi-supervised learning with consistency and confidence, in: Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 596–608.
[41] C. Doersch, A. Gupta, A. Efros, Unsupervised visual representation learning by context prediction, in: 2015 IEEE International Conference on Computer Vision (ICCV), IEEE Computer Society, Los Alamitos, CA, USA, 2015, pp. 1422–1430.
[42] S. Gidaris, P. Singh, N. Komodakis, Unsupervised representation learning by predicting image rotations, in: International Conference on Learning Representations, 2018.
[43] M. Noroozi, P. Favaro, Unsupervised learning of visual representations by solving jigsaw puzzles, in: Computer Vision – ECCV 2016, Springer International Publishing, Cham, 2016, pp. 69–84.
[44] S. Becker, G.E. Hinton, Self-organizing neural network that discovers surfaces in random-dot stereograms, Nature 355 (6356) (1992) 161–163.
[45] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a "siamese" time delay neural network, in: Proceedings of The 6th International Conference on Neural Information Processing Systems, in: NIPS'93, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 1993, pp. 737–744.
[46] P.H. Le-Khac, G. Healy, A. Smeaton, Contrastive representation learning: A framework and review, IEEE Access 8 (2020) 193907–193934.
[47] A. van den Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, 2018, URL http://arxiv.org/pdf/1807.03748v2.
[48] M. Caron, P. Bojanowski, A. Joulin, M. Douze, Deep clustering for unsupervised learning of visual features, in: Proceedings of The European Conference on Computer Vision (ECCV), 2018, pp. 132–149.
[49] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, A. Joulin, Unsupervised learning of visual features by contrasting cluster assignments, Adv. Neural Inf. Process. Syst. 33 (2020).
[50] Z. Wu, Y. Xiong, S.X. Yu, D. Lin, Unsupervised feature learning via non-parametric instance discrimination, in: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 3733–3742.
[51] H. Al-Tahan, Y. Mohsenzadeh, CLAR: Contrastive learning of auditory representations, in: Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, in: Proceedings of Machine Learning Research, Vol. 130, PMLR, 2021, pp. 2530–2538.
[52] M.N. Mohsenvand, M.R. Izadi, P. Maes, Contrastive representation learning for electroencephalogram classification, in: Proceedings of The Machine Learning For Health NeurIPS Workshop, in: Proceedings of Machine Learning Research, Vol. 136, PMLR, 2020, pp. 238–253.
[53] W.Y. Wang, D. Yang, That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets, in: Proceedings of The 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 2557–2563.
[54] R. Sennrich, B. Haddow, A. Birch, Improving neural machine translation models with monolingual data, in: Proceedings of The 54th Annual Meeting of The Association For Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, 2016, pp. 86–96.
[55] Q. Wen, L. Sun, X. Song, J. Gao, X. Wang, H. Xu, Time series data augmentation for deep learning: A survey, 2020, arXiv preprint arXiv:2002.12478.
[56] H.A. Dau, E. Keogh, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, Hexagon-ML, The UCR time series classification archive, 2018.
[57] A. Le Guennec, S. Malinowski, R. Tavenard, Data augmentation for time series classification using convolutional neural networks, in: ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data, Riva Del Garda, Italy, 2016.
[58] Z. Cui, W. Chen, Y. Chen, Multi-scale convolutional neural networks for time series classification, 2016, arXiv preprint arXiv:1603.06995.
[59] T.T. Um, F.M.J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, D. Kulić, Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks, in: Proceedings of The 19th ACM International Conference on Multimodal Interaction, 2017, pp. 216–220.
[60] G. Forestier, F. Petitjean, H.A. Dau, G. Webb, E. Keogh, Generating synthetic time series to augment sparse datasets, in: 2017 IEEE International Conference on Data Mining (ICDM), 2017, pp. 865–870.
[61] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Data augmentation using synthetic data for time series classification with deep residual networks, 2018, arXiv preprint arXiv:1808.02455.
[62] T. Wang, P. Isola, Understanding contrastive representation learning through alignment and uniformity on the hypersphere, in: Proceedings of The 37th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, Vol. 119, PMLR, 2020, pp. 9929–9939.
[63] J. Gao, X. Song, Q. Wen, P. Wang, L. Sun, H. Xu, Robusttad: Robust time series anomaly detection via decomposition and convolutional neural networks, 2020, arXiv preprint arXiv:2002.09545.
[64] R. Katarya, T. Prasad, A survey on time series online sequential learning algorithms, 2017.
[65] R. Katarya, S. Rastogi, A study on neural networks approach to time-series analysis, in: 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018, pp. 116–119.
[66] R. Katarya, A. Lal, A study on combating emerging threat of deepfake weaponization, in: 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2020, pp. 485–490.
[67] G. Gupta, R. Katarya, Research on understanding the effect of deep learning on user preferences, Arab. J. Sci. Eng. 46 (2020).
[68] R. Katarya, Y. Arora, Capsmf: a novel product recommender system using deep learning based text analysis model, Multimedia Tools Appl. (2020) 1–22.
[69] G. Gupta, R. Katarya, A study of deep reinforcement learning based recommender systems, in: 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC), 2021, pp. 218–220.
[70] J. Deng, W. Dong, R. Socher, L. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[71] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[72] M. Meyer, Signalverarbeitung : Analoge Und Digitale Signale, Systeme Und Filter, Springer Fachmedien Wiesbaden, Wiesbaden, 2014.
[73] J. Downs, E. Vogel, A plant-wide industrial process control problem, Comput. Chem. Eng. 17 (3) (1993) 245–255, Industrial challenge problems in process control.
[74] L.H. Chiang, E.L. Russell, R.D. Braatz, Fault Detection and Diagnosis in Industrial Systems, in: Advanced Textbooks in Control and Signal Processing, Springer London, London, 2001.
[75] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.
[76] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. Icml, Vol. 30, 2013, p. 3.
[77] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of The 32nd International Conference on International Conference on Machine Learning - Volume 37, in: ICML'15, JMLR.org, 2015, pp. 448–456.
[78] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017, http://arxiv.org/abs/1412.6980.
[79] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Inf. Process. Manage. 45 (2009) 427–437.
[80] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learning Res. 12 (2011) 2825–2830.
[81] J. Yuan, Y. Tian, A multiscale feature learning scheme based on deep learning for industrial process monitoring and fault diagnosis, IEEE Access 7 (2019) 151189–151202.