Biegel, Tobias; Helm, Patrick; Jourdan, Nicolas; Metternich, Joachim

SSMSPC: self-supervised multivariate statistical in-process control in discrete manufacturing processes

Journal of Intelligent Manufacturing

Journal of Intelligent Manufacturing (2024) 35:2671–2698
https://doi.org/10.1007/s10845-023-02156-7

SSMSPC: self-supervised multivariate statistical in-process control in discrete manufacturing processes

Tobias Biegel[1] · **Patrick Helm**[1] · **Nicolas Jourdan**[1] · **Joachim Metternich**[1]

Received: 10 January 2023 / Accepted: 27 May 2023 / Published online: 29 June 2023

Abstract
Self-supervised learning has demonstrated state-of-the-art performance on various anomaly detection tasks. Learning effective representations by solving a supervised pretext task with pseudo-labels generated from unlabeled data provides a promising concept for industrial downstream tasks such as process monitoring. In this paper, we present SSMSPC a novel approach for multivariate statistical in-process control (MSPC) based on self-supervised learning. Our motivation for SSMSPC is to leverage the potential of unsupervised representation learning by incorporating self-supervised learning into the general statistical process control (SPC) framework to develop a holistic approach for the detection and localization of anomalous process behavior in discrete manufacturing processes. We propose a pretext task called Location + Transformation prediction, where the objective is to classify both, the type and the location of a randomly applied augmentation on a given time series input. In the downstream task, we follow the one-class classification setting and apply the Hotelling's T^2 statistic on the learned representations. We further propose an extension to the control chart view that combines metadata with the learned representations to visualize the anomalous time steps in the process data which supports a machine operator in the root cause analysis. We evaluate the effectiveness of SSMSPC with two real-world CNC-milling datasets and show that it outperforms state-of-the-art anomaly detection approaches, achieving 100% and 99.6% AUROC, respectively. Lastly, we deploy SSMSPC at a CNC-milling machine to demonstrate its practical applicability when used as a process monitoring tool in a running process.

Keywords Self-supervised learning · MSPC · Anomaly Detection · Discrete Manufacturing

Introduction

SPC is a well-known concept to monitor the condition of a process over time with the objective of detecting any anomalous process behavior that affects the performance of a process (Ferrer, 2007; Kourti & MacGregor, 1996; Zhang et al., 2015). Within the discrete manufacturing domain, the practical application of SPC is typically based on univariate measurements of predefined quality characteristics of manufactured parts that are sampled in equidistant time intervals from the process (Montgomery, 2009). Literature agrees that this univariate post-process SPC-scheme is outdated, since it ignores the large amount of available process data in today's data-rich manufacturing environments (Ferrer, 2014; Kourti & MacGregor, 1996; MacGregor, 1997; Woodall, 2017). Thus, in recent years, researchers in discrete manufacturing encouraged to shift this paradigm and transcend from univariate post-process SPC to MSPC by using sensor data collected from the machining process that are analyzed with machine learning (ML) methods to evaluate the process condition (Biegel et al., 2022a, 2022b; Li et al., 2020; Qiu & Xie, 2021). Figure 1 visualizes this paradigm-shift.
The general SPC framework consists of two distinct phases. Phase I represents the offline monitoring phase, where the

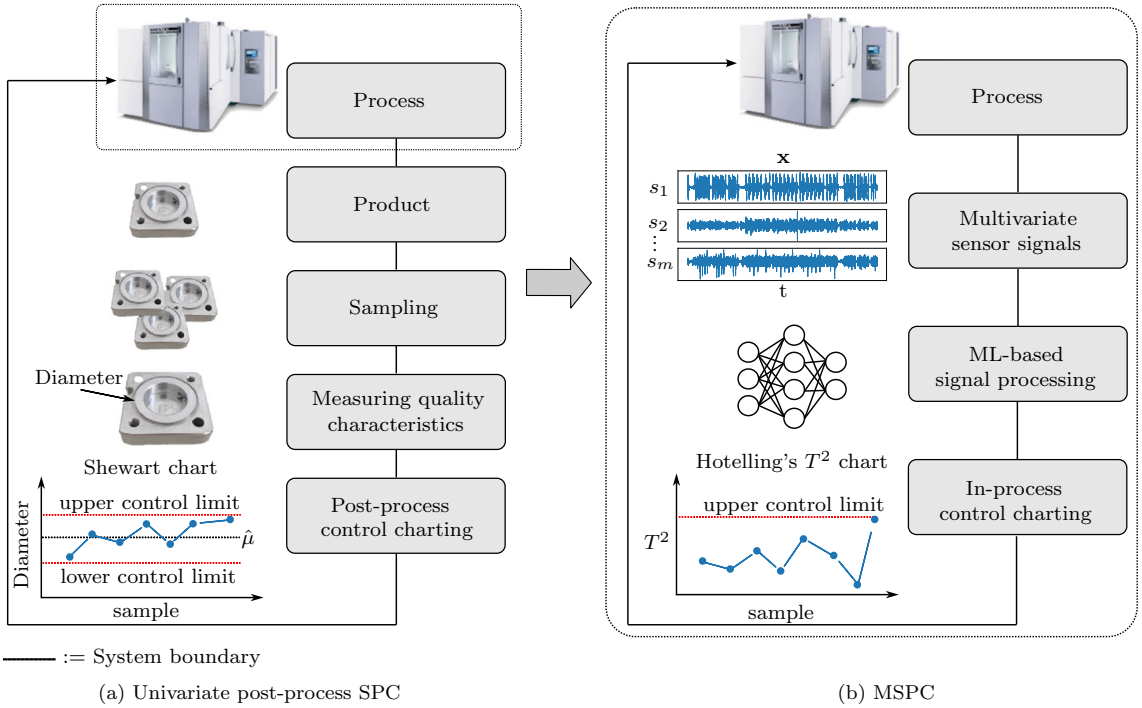
✉ Tobias Biegel
t.biegel@ptw.tu-darmstadt.de

Patrick Helm
patrick.helm@stud.tu-darmstadt.de

Nicolas Jourdan
n.jourdan@ptw.tu-darmstadt.de

Joachim Metternich
j.metternich@ptw.tu-darmstadt.de

[1] Institute of Production Management, Technology and Machine Tools, Technical University of Darmstadt, Otto-Berndt-Straße 2, 64287 Darmstadt, Hesse, Germany



(a) Univariate post-process SPC (b) MSPC

Fig. 1 Paradigm-shift in SPC. **a** Univariate post-process SPC: Univariate measurements of predefined quality characteristics, i.e., quality data of manufactured parts are sampled in equidistant time intervals from the process to evaluate the process condition. **b** MSPC: Multivariate sensor signals, i.e., process data, are sampled continuously from the process and are analyzed with ML-based methods to evaluate the process condition

control limits of the process are determined, based on historical normal process condition data. Phase II is the actual process monitoring phase, where new samples are drawn from the process and compared to the control limits to assess the process condition (Grasso et al., 2015; Qiu & Xie, 2021; Woodall, 2000; Woodall et al., 2004; Woodall & Montgomery, 1999). From an anomaly detection perspective, the SPC framework essentially describes a semi-supervised anomaly detection problem (Grasso et al., 2015; Wu et al., 2021; Xie & Peihua, 2022). Semi-supervised anomaly detection assumes that the training data incorporate only normal samples. The task is to model the normal behavior of the data and flag samples that strongly deviate from this state as anomalies (Chandola et al., 2009; Gu et al., 2019; Kumagai et al., 2019; Ruff et al., 2020; Shen et al., 2021; Tax & Duin, 2004; Ye et al., 2021). Due to this analogy, we treat MSPC in general and SSMSPC in particular as a semi-supervised anomaly detection problem.

It is worth noting that some researchers refer to the semi-supervised anomaly detection problem as unsupervised anomaly detection, see, e.g., Bergmann et al. (2019), Dehaene et al. (2020), and Zhang et al. (2019). However, unsupervised anomaly detection typically refers to a problem setting in which most but not all data in the training set are assumed to belong to the normal class (Bergman & Hoshen, 2020; Dai & Chen, 2022; Goyal et al., 2020; Pang et al., 2021).

Research in anomaly detection is extensive, with many papers being published in recent years that focus on both shallow learning and deep learning approaches, see, e.g., Chalapathy and Chawla (2019), Chandola et al. (2009), Gupta et al. (2014), and Pang et al. (2021) for excellent reviews. Ruff et al. (2021), developed a unifying view for shallow and deep anomaly detection approaches in which they identify four main categories to which these methods can be assigned: (1) one-class classification, (2) probabilistic models, (3) reconstruction models and (4) distance-based methods.

Recently, anomaly detection methods that rely on self-supervised learning have shown outstanding performance on various benchmark tasks. Self-supervised learning is a form of unsupervised learning which aims to learn effective representations for real-world downstream tasks from unlabeled data by solving a supervised pretext task with automatically generated pseudo-labels, e.g., solving jigsaw puzzles or predicting image rotations (Doersch et al., 2015; Gidaris et al., 2018; Jing & Tian, 2020; Noroozi & Favaro, 2016; Noroozi et al., 2018). Anomaly detection methods based on self-supervised learning derive their anomaly score either directly from the pretext task or by using the learned representations

in the downstream anomaly detection task (Qiu et al., 2021; Sohn et al., 2021). Thus, defining suitable pretext tasks is a vital component in self-supervised learning approaches (Li et al., 2021).

In this paper, we present SSMSPC, a novel approach for MSPC based on self-supervised learning to detect and localize anomalous process behavior in discrete manufacturing processes. We propose a pretext task that we refer to as Location + Transformation prediction. Given a time series input that has been augmented by one of k predefined augmentation functions in one of p equally sized windows, the objective in this pretext task is to classify both, the augmentation and the corresponding window in a multi-task fashion. In the downstream task, we follow the conventional one-class classification setting and compute the Hotelling's T^2 statistic as the anomaly score, based on the learned representations of the pretext task. The control limits are fitted with Kernel Density Estimation (KDE). In addition to that, we propose an extension to the traditional control chart view that combines metadata with the learned representations to (1) segment the process data into the individual process steps and (2) highlight the anomalous time steps, which supports a machine operator in the root cause analysis.

To summarize, the contribution of this paper is threefold:

- We propose SSMSPC, a novel approach for MSPC based on self-supervised learning to detect and localize anomalous process behavior in discrete manufacturing processes.
- We present a pretext task called Location + Transformation prediction for learning effective representations, where the objective is to classify both, the type and the location of the augmentation based on a given randomly augmented time series input.
- We introduce an extension to the conventional control chart view to facilitate the identification of the root cause by segmenting a raw time series signal into the individual process steps using metadata and highlighting the anomalous components.

The remainder of this paper is structured as follows: "Related work" section presents the related work with respect to recent developments in self-supervised anomaly detection and applications of in-process monitoring in continuous processes as well as discrete manufacturing processes. "Problem statement" section provides a comprehensive description of the general problem statement that we consider for the application of SSMSPC. In "SSMSPC" section, we introduce the individual components of SSMSPC. This includes a detailed explanation of the applied framework, the proposed pretext task, the subsequent downstream task and the control chart extension. "Experiments" section presents the experiments based on two real-world CNC-milling datasets. We compare SSMSPC with state-of-the-art anomaly detection baselines and conduct a comprehensive ablation study. Our contribution ends with the conclusion and an outlook for future research.

Related work

Self-supervised anomaly detection

The amount of research related to self-supervised anomaly detection has grown rapidly over recent years. Golan and El-Yaniv (2018) presented Geometric Transformations (GeoTrans) for anomaly detection in images. The authors designed a pretext task, where a multiclass neural classifier is trained to discriminate between geometric transformations that have been applied to normal images. The detection of anomalous images is accomplished by evaluating the softmax activations of the model when transformed images are used as an input. In a paper by Hendrycks et al. (2019), the authors combine rotation prediction (Gidaris et al., 2018) with geometric transformation prediction in their pretext task and are able to outperform a purely supervised approach based on outlier exposure (Hendrycks et al., 2019) to detect anomalies in images. Bergman and Hoshen (2020) present GOAD, a self-supervised anomaly detection approach for general data, i.e., images, tabular data etc. They extend the class of transformation functions in the pretext task to include random affine transformations to generalize to non-image data. Tack et al. (2020) propose contrasting shifted instances (CSI), which is based on the conventional contrastive learning scheme for learning visual representations. They introduce a new training method, in which a given sample is contrasted with distributionally-shifted augmentations of itself as well as other instances. By incorporating this into a new detection score, they achieve strong performance on state-of-the-art image anomaly detection tasks. Shen et al. (2020) propose THOC, a temporal hierarchical one-class network for time series anomaly detection. The authors use a dilated recurrent neural network with skip connections, and apply multiple hyperspheres obtained from a hierarchical clustering process to develop their one-class classification objective. They incorporate self-supervision by using a pretext task for multi-step-ahead prediction. Qiu et al. (2021) propose neural transformation learning for anomaly detection (NeuTraL AD) for data types beyond images. Their approach consists of a fixed set of learnable transformations and an encoder, that are both trained jointly on a deterministic contrastive loss (DCL) that is also used to score new samples at test time. Sohn et al. (2021) present a two-stage framework for deep one-class classification. They learn self-supervised representations from one-class data and then build a conventional one-class classifier on top of the learned representations.

The authors present a thorough analysis of different self-supervised representation learning algorithms under their proposed framework. Li et al. (2021) introduce CutPaste, a simple augmentation strategy that cuts an image patch and pastes it at a random location. Their pretext task involves detecting whether an image has been augmented with CutPaste. They follow the two-stage framework proposed by Sohn et al. (2021). Fu and Xue (2022) introduce MAD, a self-supervised learning task for time series anomaly detection. The objective of their pretext task is to predict the values of randomly masked samples of a time series input. In a paper by Shenkar et al. (2022), the authors present a novel contrastive learning approach for anomaly detection in tabular data. Given a data sample with masked features, the proposed learning approach is based on the assumption that the remaining features can be used to identify the masked ones. Wang et al. (2023) propose COCA, a negative-sample-free contrastive one-class anomaly detection method for time series data. The authors apply jittering and scaling augmentations to expand the number of training samples. They consider the representation in the latent space and the reconstructed representation of a Seq2Seq model as positive pairs.

In-process monitoring applications

Continuous processes

Process monitoring based on MSPC and ML has seen wide application in industrial processes. Most of these applications originated in the context of continuous processes, such as chemical, petrochemical or polymer processes. The most basic approaches rely on Hotelling's T^2 and Squared Prediction Error (SPE) monitoring statistics that are computed based on Principal Component Analysis (PCA) or Partial Least Squares (PLS) (Ge & Song, 2013; Qin, 2003, 2012; Wang et al., 2018). Throughout the years, many shallow learning approaches have been presented and extensively reviewed, see, e.g., Yin et al. (2014), Alauddin et al. (2018), and Qin and Chiang (2019). Recently, research with respect to deep learning applications has been very active, with numerous articles being published every year (Yu & Zhang, 2023).

Yu et al. (2021) use a convolutional long short-term memory autoencoder (CLSTM-AE) for process monitoring and compute Hotelling's T^2 and SPE monitoring statistics in the latent space and residual space, respectively. They validate their approach using two industrial benchmark processes, namely the Tenessee-Eastman process (TEP) and the continuous stirred tank reactor (CSTR) and find it to outperform conventional approaches such as PCA and LSTM-AE. In another paper by Yu and Zhang (2020), the authors propose a manifold regularized stacked AE (MRSAE) that relies on Hotelling's T^2 and SPE as monitoring statistics. Their suggested approach outperforms other deep learning approaches on the TEP and the Fed-Batch fermentation penicillin process (FBFP) benchmark. In a paper by Cheng et al. (2019), the authors present a novel monitoring approach based on a variational recurrent AE (VRAE). They use the negative variational scores as the monitoring statistic and fit the control limit with KDE. The approach is validated using the TEP. Kong and Yan (2020) introduce the inner product-based stacked AE (IPSAE), which adds the inner product between the outputs of the neurons to the loss function for regularization purposes. They compute monitoring statistics in the feature space and the residual space and fit the control limit via KDE. For evaluation purposes, the TEP process is used and compared to PCA and stacked AE. In Tang et al. (2020), the authors combine Gaussian mixture models with a variational AE to monitor nonlinear processes with multiple operating modes. They construct latent variable and reconstruction variable indices as monitoring statistics. The authors validate their approach on the TEP and a hot strip mill process. Zhang et al. (2021) introduce a hybrid deep learning model based on a 1D-CNN and a stacked denoising AE. They demonstrate the effectiveness of their approach using the TEP, FBFP and a real-world industrial process for conveyor belts. Li et al. (2022) present a slow feature analysis-aided AE (SFA-AE) which leverages the extracted high-level features by the AE to learn deep slow variation patterns. With these patterns, they compute monitoring statistics based on Hotelling's T^2 and SPE. In addition, they incorporate a self-attention mechanism to identify the anomalous process variables in a contribution plot. Liu et al. (2022) introduce a novel stacked multimanifold AE (S-MMAE) to predict and monitor key quality variables in industrial processes. The authors validate their approach using a real-world hydrocracking process. In Ai et al. (2023), the authors present KD-SCL, a novel industrial process monitoring framework based on knowledge distillation and contrastive learning. They rely on memory queue-based negative sample augmentation and hard negative sampling mechanisms to support the selection of negative samples for contrastive learning. The approach is validated using data from a lead-zinc flotation plant. Lu et al. (2023) introduce a cascaded bagging-PCA and CNN classification network. They define a self-supervised pretext task by trying to discriminate between the reconstructions of bagged and conventional PCA. The approach is validated using the TEP. Li et al. (2023) propose a self-supervised learning framework based on multisource heterogeneous contrastive learning. They employ a two-stage framework, in which the self-supervised feature learning phase is followed by a supervised fine-tuning step. The authors show the effectiveness of their approach using data collected from a heavy-plate production process.

Discrete manufacturing processes

Regarding process monitoring in the context of discrete manufacturing processes, most of the existing research focuses on shallow or deep anomaly detection approaches that rely on some kind of reconstruction-based method, such as the AE. Biegel et al. (2022b) propose a novel approach that uses text data from machine operators to efficiently label the normal process condition data retrieved from a real-world CNC-milling process. Based on these data, they fit a simple PCA-based model and monitor the process with Hotelling's T^2 and Squared Prediction Error (SPE) control charts. In another paper by Biegel et al. (2022a), the authors investigate the application of deep AE-based monitoring approaches and experiment with the reconstruction error and latent representation of the input data to compute Hotelling's T^2 and SPE monitoring statistics. They use a real-world sheet metal forming process for their evaluation. In Lindemann et al. (2019), the authors present two data-driven self-learning approaches relying on k-means and LSTM-AE that are used to detect anomalies within a massive forming process. Lindemann et al. (2020) introduce a novel approach for anomaly detection in discrete manufacturing processes based on LSTM-AE that is evaluated on a multi-step forging process. Proteau et al. (2020) use a variational AE to monitor the condition of a CNC-milling process from the aerospace industry. Hahn and Mechefske (2021) present a disentangled variational AE with a temporal CNN to monitor tool wear in a self-supervised manner. They validate their approach using data from a CNC-milling process of small ball-valves. The authors treat the general AE scheme as a self-supervised learning objective. Ahmad et al. (2020) develop a hybrid model based on deep learning and SPC to monitor manufacturing processes in the presence of image or video data. They apply a fast region-based CNN to derive statistical features that are then plotted in an exponentially weighted moving average (EWMA) control chart. However, the authors validate their approach using only a simulated video. Lorenti et al. (2022) present CUAD-Mo, an approach for anomaly detection in machine operations that is based on Isolation Forest (IF) (Liu et al., 2008). The approach is validated in a CNC-milling process. Oshida et al. (2023) propose a stacked LSTM encoder-decoder model for anomaly detection. Their approach is evaluated on a real-world turning process for Inconel 718 to detect tool wear based on acoustic emission signals. In Sun et al. (2023), the authors introduce an AE-based semi-supervised anomaly detection method for cutting tools in machining processes. They validate the proposed method on an experimental and a public cutting tool wear dataset.

The related work presented here demonstrates that the incorporation of self-supervised anomaly detection methods in process monitoring schemes for both continuous processes and discrete manufacturing processes is still rare and has not yet gained momentum. However, we expect that this line of research is going to grow rapidly in the near future.

Problem statement

In this paper, we address the semi-supervised anomaly detection problem in the context of discrete manufacturing processes. Specifically, we consider the problem of detecting and localizing anomalous process behavior using an ML model trained exclusively on data that correspond to a normal process condition. The reason for restricting ourselves to this problem setting is twofold: First, as pointed out in "Introduction" section, the assumptions made in the general SPC framework correspond to a semi-supervised anomaly detection problem. Second, especially in discrete manufacturing, anomalies typically account for rare data instances while normal data are easier to obtain and generally represent the majority of available data (Chalapathy & Chawla, 2019; Pang et al., 2021).

When we speak of a discrete manufacturing process, we assume a machining process in which, (1) high-frequency process data, e.g., vibrations, and cutting forces are recorded throughout the processing cycle of a part in the form of a multivariate time series and (2) the process data are associated with the final quality characteristics of the produced part. Here, the second condition is of crucial importance to ensure that the monitoring scheme is in alignment with the economic objective of the process, which is to produce parts that meet the quality specifications. Thus, if an anomalous process condition is signaled, it should correspond to an increased likelihood of having produced a part that is out of specification.

With the preceding explanations, we can formalize the problem statement as follows: Let $\mathcal{D} \subset \mathbb{R}^{n \times m}$ be the set of all possible process data of a machining process, where n corresponds to the number of time steps in a processing cycle and m refers to the number of features, i.e., sensors used to record the process data. Each element $\mathbf{x} \in \mathcal{D}$ thus represents a manufactured part in the form of a multivariate time series. Let further $X \subset \mathcal{D}$ represent the set of all normal process condition data.

Our overall objective is to provide a machine operator with the information whether the process data $\mathbf{x}$ for a produced part correspond to a normal process condition. To achieve this, we wish to learn a mapping $h : \mathcal{D} \to \mathbb{R}$, where larger values of $h(\mathbf{x})$ indicate an increased probability that $\mathbf{x} \notin X$, i.e., $\mathbf{x}$ is likely to represent an anomalous process condition. Based on h, we require a threshold mapping $b_{\text{ucl}} : \mathbb{R} \to \{0, 1\}$ such that

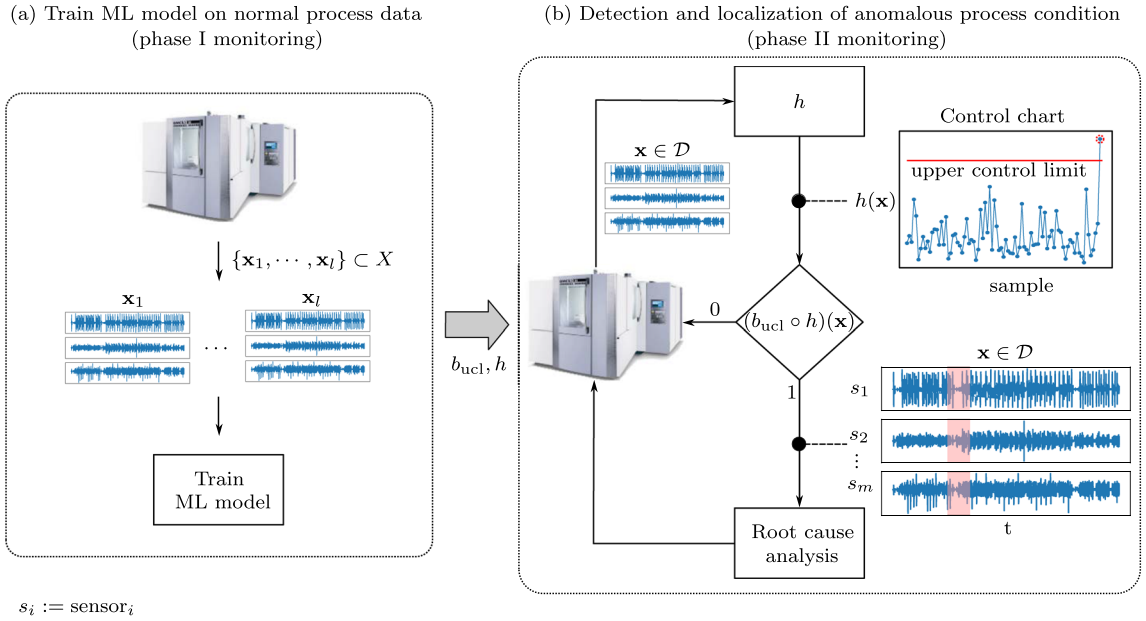


Fig. 2 Problem statement. We address the problem of anomaly detection and localization in the context of discrete manufacturing processes. **a** An ML model is trained on l normal process data $\{\mathbf{x}_1, \ldots, \mathbf{x}_l\} \subset X$, with $\mathbf{x}_i \in \mathbb{R}^{n \times m}$ to learn the mappings h and b_{ucl}. This corresponds to phase I in the SPC framework. **b** The trained model is used to monitor new samples from the process. The anomaly scores $h(\mathbf{x})$ are plotted in a control chart. If an anomalous process condition has been detected, i.e., $(b_{\text{ucl}} \circ h)(\mathbf{x}) = 1$, the model should locate the anomalous process condition in the time series to support a machine operator in the root cause analysis. This corresponds to phase II in the SPC framework

$$(b_{\text{ucl}} \circ h)(\mathbf{x}) = \left\{ \begin{array}{l} 1, \text{ if } h(\mathbf{x}) > \text{ucl} \\ 0, \text{ if } h(\mathbf{x}) \leq \text{ucl} \end{array} \right\}, \tag{1}$$

where ucl is the corresponding threshold, i.e., the upper control limit. Thus, process samples exceeding the ucl are flagged as anomalies.

Once an anomalous process condition has been detected, the ML model shall support a machine operator in the root cause analysis by locating the anomalous process condition in the time series. See Fig. 2 for a visualization of the problem statement considered in this work.

Note that SSMSPC can be applied to any discrete manufacturing process that is in accordance with this problem statement.

SSMSPC

In this section, we present SSMSPC. "Framework" section highlights the key components of the framework that constitutes our approach. "Pretext task" section dives into the details of the proposed self-supervised pretext task. "Downstream task" section explains how the learned representations of the pretext task are used to build a one-class classifier based on Hotelling's T^2 statistic (Hotelling, 1947) in the corresponding downstream task. Finally, in "Control chart extension" section, we demonstrate how the results of our approach can be visually interpreted by a machine operator to help identify the root cause for an anomalous process condition by highlighting the respective anomalous sections in the raw time series signal.

Framework

We follow the two-stage framework introduced by Sohn et al. (2021). The rationale for relying on this two-stage framework is twofold. First, self-supervised learning is inherently a two-stage process, where the first stage corresponds to solving the pretext task and the second stage embodies the downstream task (Jing & Tian, 2020; Noroozi et al., 2018). This framework thus represents a natural choice for self-supervised anomaly detection methods. Second, the selected framework has demonstrated its effectiveness over end-to-end approaches in previous works, see, e.g., Li et al. (2021). Figure 3 illustrates the aforementioned framework.

In the first stage, self-supervised learning is used to learn meaningful representations from normal process condition data with the help of a predefined pretext task. An encoder network f is used to transform the input data $\mathbf{x}$ to a latent representation $f(\mathbf{x})$, that are then fed through a prediction head g, which is usually represented by a simple multi-layer perceptron (MLP) with a softmax output for the respective

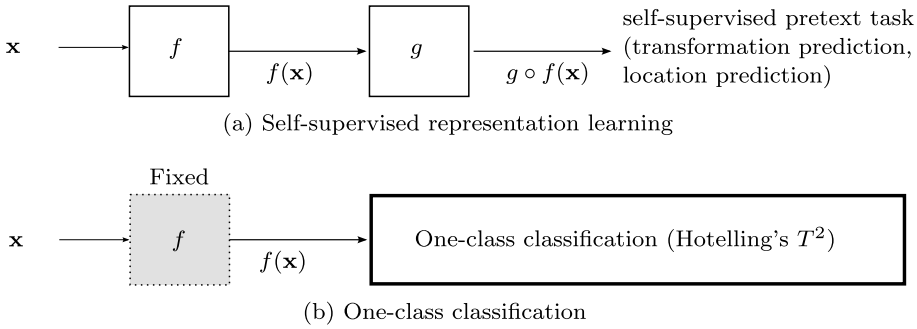


Fig. 3 Two-stage framework based on Sohn et al. (2021). **a** In the first stage, the model has to solve a predefined pretext task to learn effective representations. **b** The second stage embodies the conventional one-class classification setting based on learned representations of the pretext task

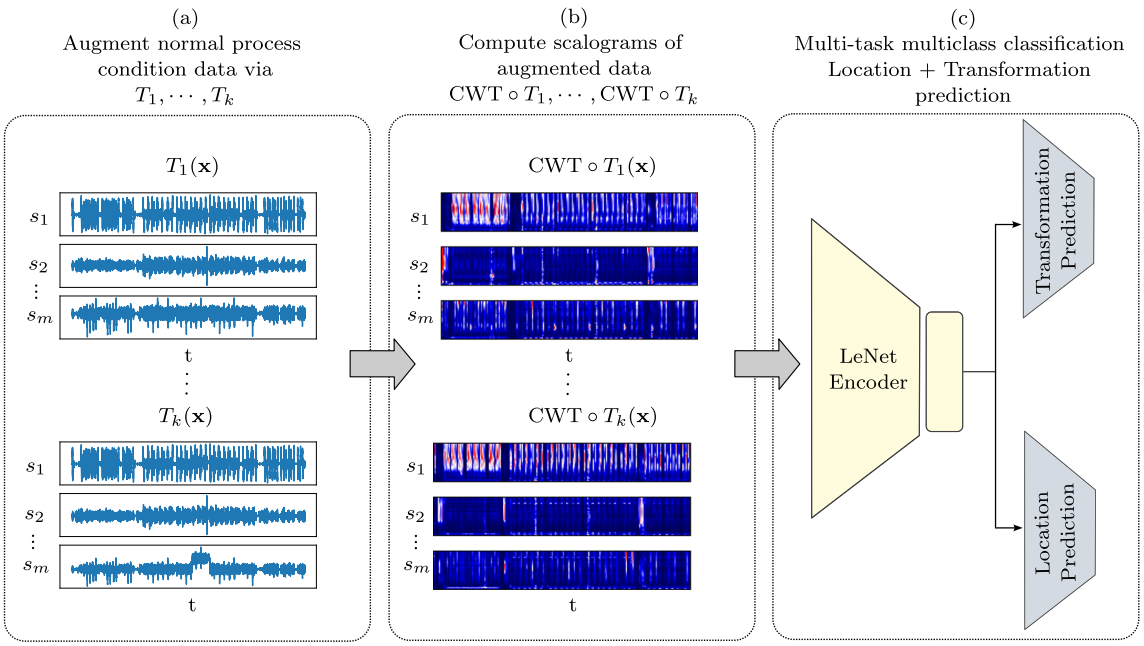


Fig. 4 Pretext task proposed in SSMSPC. **a** Normal process condition data are augmented in a randomly selected window, with a set of k predefined augmentations $T_1, \ldots, T_k$. This increases the dataset size by a factor of k. **b** CWT is applied to the augmented dataset to compute the scalogram representation. **c** The scalograms are used as an input for a LeNet-type encoder network that has two prediction heads attached to it. Given a scalogram as input, the task is to correctly predict the type and the location of the applied augmentation

classification task. The model is trained end-to-end on the respective pretext task using backpropagation.

The second stage corresponds to the downstream task where a one-class classifier is constructed on top of the learned representations of the pretext task. For this stage, it is common practice to discard the prediction head g, and only use the pretrained encoder network f as a feature extractor, since it has been shown that the learned representations of the layer right before g provide better representations (Ermolov et al., 2021; Chen et al., 2020).

It is worth mentioning that with respect to the general SPC framework both, pretext task and downstream task are incorporated in the offline monitoring phase, i.e., phase I, as they are used to fit the control limits for subsequent phase II monitoring.

Pretext task

In this work, we propose a pretext task that we refer to as Location + Transformation prediction, which is specifically designed to be applied in the setting described in "Problem statement" section.

The core intuition behind our pretext task is (1) to learn which augmentation was applied to a given time series input, and (2) to learn where the augmentation occurred in the time series by predicting the respective augmentation window. By training a model with this pretext task, we show that the learned representations are highly effective for monitoring discrete manufacturing processes. Figure 4 provides a visualization of the individual components that constitute the proposed Location + Transformation prediction pretext task.

Assuming a training set $X_{\text{train}} \subset X$ we apply a set $\mathcal{T} := \{T_1, \ldots, T_k \mid T_i : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^{n \times m}\}$ of k different augmentation functions. Each T_i augments the entire set of available normal process condition data X_{train}. For every $\mathbf{x} \in X_{\text{train}}$ the augmentation is applied in a randomly chosen window W_j for all m sensors using a set of p predefined windows $\{W_1, \ldots, W_p\}$. The augmented time series data $T_i(\mathbf{x})$ are consequently transformed using the continuous wavelet transformation (CWT) to retrieve the respective scalogram representation $\text{CWT} \circ T_i(\mathbf{x}) \in \mathbb{R}^{s \times n \times m}$, where s represents the number of different scales of the chosen wavelet. Hence, by transforming the normal process condition data in this way, we retrieve a representation that can be interpreted as an m-channel image. Following this routine, the original dataset size is increased by a factor of k. Algorithm 1 summarizes the general augmentation procedure.

With the completion of the data preparation, the augmented dataset is used as an input for the subsequent Location + Transformation prediction task. The scalograms are fed into a simple LeNet-type encoder network f (Lecun et al., 1998) on which two prediction heads g_{trans} and g_{loc} are attached for Transformation prediction and Location prediction, respectively. The prediction heads are represented by two equivalent MLPs differing only in the size of the softmax output layer. Here, g_{trans} outputs a softmax layer of size k to predict the applied augmentation, whereas g_{loc} outputs a softmax layer of size $p + 1$ to predict the window in which the augmentation occurred. Note that the output of g_{loc} is $p + 1$ in order to account for the case when no augmentation was applied, and thus no window was chosen.

Algorithm 1 General augmentation procedure

Require: $T_1, \ldots, T_k, X_{\text{train}}, p$
$X_{\text{augmented}} \leftarrow \{\}$
compute window bounds for p windows
for $i \in \{1, \ldots, k\}$ **do**
  for $\mathbf{x} \in X_{\text{train}}$ **do**
    Randomly choose a window W_j
    Augment $\mathbf{x}$ in window W_j with T_i
    Compute scalogram $\text{CWT} \circ T_i(\mathbf{x})$
    Append $\text{CWT} \circ T_i(\mathbf{x})$ to $X_{\text{augmented}}$
  end for
end for
Shuffle $X_{\text{augmented}}$
return $X_{\text{augmented}}$

In the following, we provide further details for the individual components of the proposed pretext task.

Data augmentations

We propose a total of $k = 4$ augmentations that have been proven most useful in learning good representations during our experiments.

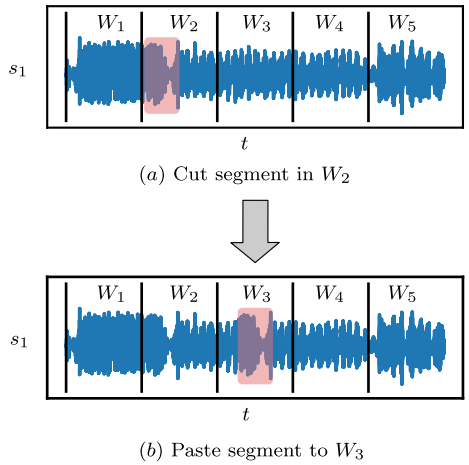


Fig. 5 Example of the CutPaste augmentation with $p = 5$ windows. **a** A random-sized segment is cut from a randomly chosen window. **b** The segment is pasted to another randomly chosen window

Identity In alignment with other recently proposed research in the field of self-supervised anomaly detection such as Tack et al. (2020) and Golan and El-Yaniv (2018) the first augmentation that we propose is simply the identity, i.e., $T_1(\mathbf{x}) = \mathbf{x}$. The reason for including the identity function is that this translates into including the normal process condition data X_{train} as one of the four classes. Since the model is exposed to the original data X_{train} during training, it needs to be able to distinguish the normal process condition from artificial anomalous data to perform well on the pretext task.

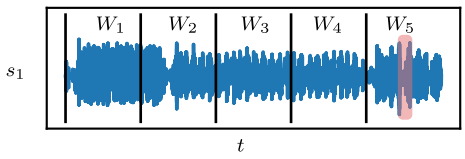
CutPaste For the second augmentation class, we draw inspiration from the work of Li et al. (2021) and as such refer to it as CutPaste. However, in contrast to the authors that proposed CutPaste to augment images, we transfer the idea of CutPaste to the time series representation and adapt it for the application in Location + Transformation prediction.

Figure 5 provides an exemplary illustration of how a time series is augmented using CutPaste. First, a random cutting window W_c is chosen from $\{W_1, \ldots, W_p\}$. Within the bounds of W_c, two points are randomly selected, marking the start and end point of the cutting segment. Next, a pasting window W_j is randomly selected from the entire set of windows. The cutting segment is then pasted to a random location within W_j. Note that the described augmentation is done for each of the m features, i.e., sensors of $\mathbf{x}$. Algorithm 2 provides further details on how the CutPaste augmentation works.

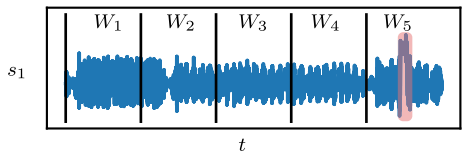
MeanShift The next augmentation that we found useful to learn effective representations is referred to as MeanShift. For each sensor in $\mathbf{x}$, this simple augmentation strategy shifts a pre-selected time series segment by the mean of the respective time series. Figure 6 shows how the proposed MeanShift augmentation works. The first step consists of selecting a random window W_j from the set of windows $\{W_1, \ldots, W_p\}$. Similar to CutPaste, two points are then randomly chosen

Algorithm 2 CutPaste

Require: X_{train}, p, window bounds b
for $\mathbf{x} \in X_{\text{train}}$ **do**
  $W_c \leftarrow$ sample cut window from $\mathcal{U}(1, p)$
  $b^{W_c}_{\text{lower}}, b^{W_c}_{\text{upper}} \leftarrow b[W_c]$
  $c_1 \leftarrow$ sample from $\mathcal{U}(b^{W_c}_{\text{lower}}, b^{W_c}_{\text{upper}})$
  $c_2 \leftarrow$ sample from $\mathcal{U}(b^{W_c}_{\text{lower}}, b^{W_c}_{\text{upper}})$
  ensure $c_2 > c_1$
  cut_snippet $\leftarrow \mathbf{x}[c_1 : c_2]$
  $\Delta \leftarrow c_2 - c_1$
  $W_j \leftarrow$ sample paste window from $\mathcal{U}(1, p)$
  $b^{W_j}_{\text{lower}}, b^{W_j}_{\text{upper}} \leftarrow b[W_j]$
  $p_1 \leftarrow$ sample from $\mathcal{U}(b^{W_j}_{\text{lower}}, b^{W_j}_{\text{upper}} - \Delta)$
  $p_2 \leftarrow p_1 + \Delta$
  $\mathbf{x}[p_1 : p_2] \leftarrow$ cut_snippet
end for
return x



(a) Select segment in W_5



(b) Shift segment mean in W_5

Fig. 6 Example of the MeanShift augmentation with $p = 5$ windows. **a** A random-sized segment is selected from a randomly chosen window. **b** The mean is computed with respect to the whole time series and added to each point in the segment

within the bounds of W_j that mark the start and end point of the segment to be augmented. For each sensor, the mean of the respective time series is then computed and added to the selected segment in W_j. Algorithm 3 summarizes the idea of the MeanShift augmentation.

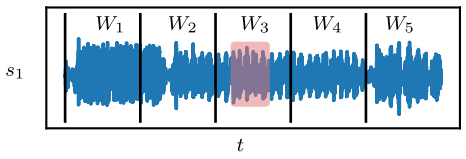
MissingSignal The last proposed augmentation is referred to as MissingSignal and resembles, as the name suggests, a missing sensor signal. Figure 7 visualizes the MissingSignal augmentation. The routine for selecting the window and the respective segment is equivalent to the presented MeanShift augmentation. However, as opposed to the other presented augmentations, the selected segment in the time series is replaced by a constant value that is equal to the first value of the original segment. Algorithm 4 provides further details on the proposed MissingSignal augmentation.

It is worth mentioning that the proposed CutPaste, MeanShift and MissingSignal augmentations were designed to approximate so-called contextual anomalies, since they represent a common type of anomalies present in time series data (Aggarwal, 2017; Chandola et al., 2009).

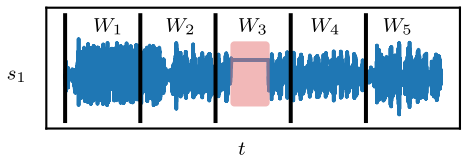
We tested more augmentations than the ones presented here, especially with respect to other anomaly classes, such as point outliers. However, as we will show in our ablation study in "Ablation study" section, these augmentations led to a deterioration in performance. Thus, we consider the augmentations presented here as the basis for SSMSPC. Nevertheless, we explicitly would like to stress the fact that, depending on the

Algorithm 3 MeanShift

Require: X_{train}, p, window bounds b
for $\mathbf{x} \in X_{\text{train}}$ **do**
  $W_j \leftarrow$ sample window from $\mathcal{U}(1, p)$
  $b^{W_j}_{\text{lower}}, b^{W_j}_{\text{upper}} \leftarrow b[W_j]$
  $c_1 \leftarrow$ sample from $\mathcal{U}(b^{W_j}_{\text{lower}}, b^{W_j}_{\text{upper}})$
  $c_2 \leftarrow$ sample from $\mathcal{U}(b^{W_j}_{\text{lower}}, b^{W_j}_{\text{upper}})$
  ensure $c_2 > c_1$
  time_series_mean $\leftarrow \mu(\mathbf{x})$
  $\mathbf{x}[c_1 : c_2] \leftarrow \mathbf{x}[c_1 : c_2] +$ time_series_mean
end for
return x



(a) Select segment in W_3



(b) Remove signal in segment in W_3

Fig. 7 Example of the MissingSignal augmentation with $p = 5$ windows. **a** A random-sized segment is selected from a randomly chosen window. **b** The signal in the segment is replaced by a constant value, i.e., the first value of the original segment

Algorithm 4 MissingSignal

Require: X_{train}, p, window bounds b
for $\mathbf{x} \in X_{\text{train}}$ **do**
  $W_j \leftarrow$ sample window from $\mathcal{U}(1, p)$
  $b^{W_j}_{\text{lower}}, b^{W_j}_{\text{upper}} \leftarrow b[W_j]$
  $c_1 \leftarrow$ sample from $\mathcal{U}(b^{W_j}_{\text{lower}}, b^{W_j}_{\text{upper}})$
  $c_2 \leftarrow$ sample from $\mathcal{U}(b^{W_j}_{\text{lower}}, b^{W_j}_{\text{upper}})$
  ensure $c_2 > c_1$
  constant $\leftarrow \mathbf{x}[c_1]$
  $\mathbf{x}[c_1 : c_2] \leftarrow$ constant
end for
return x

scenario at hand, other augmentations might be considered as a useful addition to the ones presented here.

Continuous wavelet transformation

CWT is a powerful mathematical tool to transform a 1D signal from the time domain into a 2D representation, also called scalogram, in the time-frequency domain. This 2D representation can be interpreted as a single channel image and thus allows the application of state-of-the-art CNN architectures to be used with time series data. The CWT of a signal $s(t)$ is defined as follows:

$$\mathrm{CWT}(s(t); a, \tau) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} s(t) \psi^* \left(\frac{t-\tau}{a} \right) dt, \tag{2}$$

where $a > 0$ corresponds to the scaling parameter, τ represents the translational value and ψ^* is the complex conjugate of the selected base wavelet, also called mother wavelet. The mother wavelet is scaled and shifted in time across the respective signal in order to compute the so-called wavelet coefficients that represent the similarity between the wavelet and the respective signal. Depending on the signal at hand, there are plenty of different types of wavelets that can be selected as the mother wavelet. A mother wavelet frequently used is the Morlet wavelet developed by Grossmann and Morlet (1984).

As opposed to other commonly used techniques for time-frequency analysis, such as Short-Time-Fourier-Transformation (STFT), CWT is particularly useful in the analysis of non-stationary signals like those found in manufacturing processes (Gao & Yan, 2011). The reason for this is that CWT allows variable window sizes with the help of the scaling parameter a to analyze different frequency components of a signal. In recent publications, researchers use CWT due to its advantages over conventional STFT for process monitoring applications such as chatter detection, grinding burn recognition, etc. (Hübner et al., 2020; Liao et al., 2021; Tran et al., 2020; Tran & Lundgren, 2020).

As shown in Fig. 4, we apply the CWT for each feature of the time series input separately and stack the resulting scalograms on top of each other to receive a tensor that is then interpreted as an m-channel image and fed to the LeNet-type encoder network.

Location + Transformation prediction

LeNet-type encoder network Inspired by Liznerski et al. (2021), Ruff et al. (2018), and Ruff et al. (2021), we use a LeNet-type encoder network as the default backbone architecture for SSMSPC. The main reasons for choosing this architecture are (1) its simplicity and (2) its low capacity. Since the amount of available data to monitor discrete manufacturing processes is usually very small compared to other domains, there is an increased risk of overfitting when using architectures with higher capacity.

We tested different network architectures in the development phase of SSMSPC, and found that the LeNet-type encoder represents a reasonable choice in terms of performance, training and inference time for the considered problem statement. It is worth noting that SSMSPC generally allows different design choices depending on the problem at hand. Thus, if the capacity of the LeNet-type encoder is too low, it can be replaced with a superior CNN-based architecture.

Transformation prediction The objective in transformation prediction is to predict the augmentation $T_i \in \{T_1, \ldots, T_k\}$ that has been applied to the time series input. To do so, we attach an MLP prediction head g_{trans} to the LeNet-type encoder f with a softmax output of size k. The loss function used in transformation prediction is simply the cross-entropy loss based on the softmax output of the prediction head.

$$\mathcal{L}_{\text{trans}} = - \sum_{i \in \{1,\ldots,k\}} y_i \log((g_{\text{trans}} \circ f \circ \mathrm{CWT})(\mathbf{x})) \tag{3}$$

Note that the idea of using a classifier to predict the respective augmentation is not new, but rather common practice in self-supervised approaches for anomaly detection, see, e.g., the works of Li et al. (2021), Tack et al. (2020), and Hendrycks et al. (2019).

Location prediction Location prediction aims at predicting the correct location, i.e., the window $W_j \in \{W_1, \ldots, W_p\}$ in which an augmentation occurred. Similar to transformation prediction, we attach a simple MLP prediction head g_{loc} with a softmax output of size $p + 1$ to the encoder network. The objective function for this task is again a simple cross-entropy loss.

$$\mathcal{L}_{\text{loc}} = - \sum_{j \in \{1,\ldots,p+1\}} y_j \log((g_{\text{loc}} \circ f \circ \mathrm{CWT})(\mathbf{x})) \tag{4}$$

Combining the idea of transformation prediction and location prediction leads to the proposed Location + Transformation prediction setting. The novelty here is to attach both prediction heads $g_{\text{trans}}, g_{\text{loc}}$ to the same encoder network f and train the network via backpropagation in a multi-task setting. The loss function for Location + Transformation prediction is just the linear combination of $\mathcal{L}_{\text{trans}}$ and $\mathcal{L}_{\text{loc}}$.

$$\mathcal{L}_{\text{loc+trans}} = \lambda_1 \mathcal{L}_{\text{trans}} + \lambda_2 \mathcal{L}_{\text{loc}} \tag{5}$$

where λ_1, λ_2 are scaling factors. For the sake of simplicity, we use $\lambda_1, \lambda_2 = 1$ for the remainder of the paper.

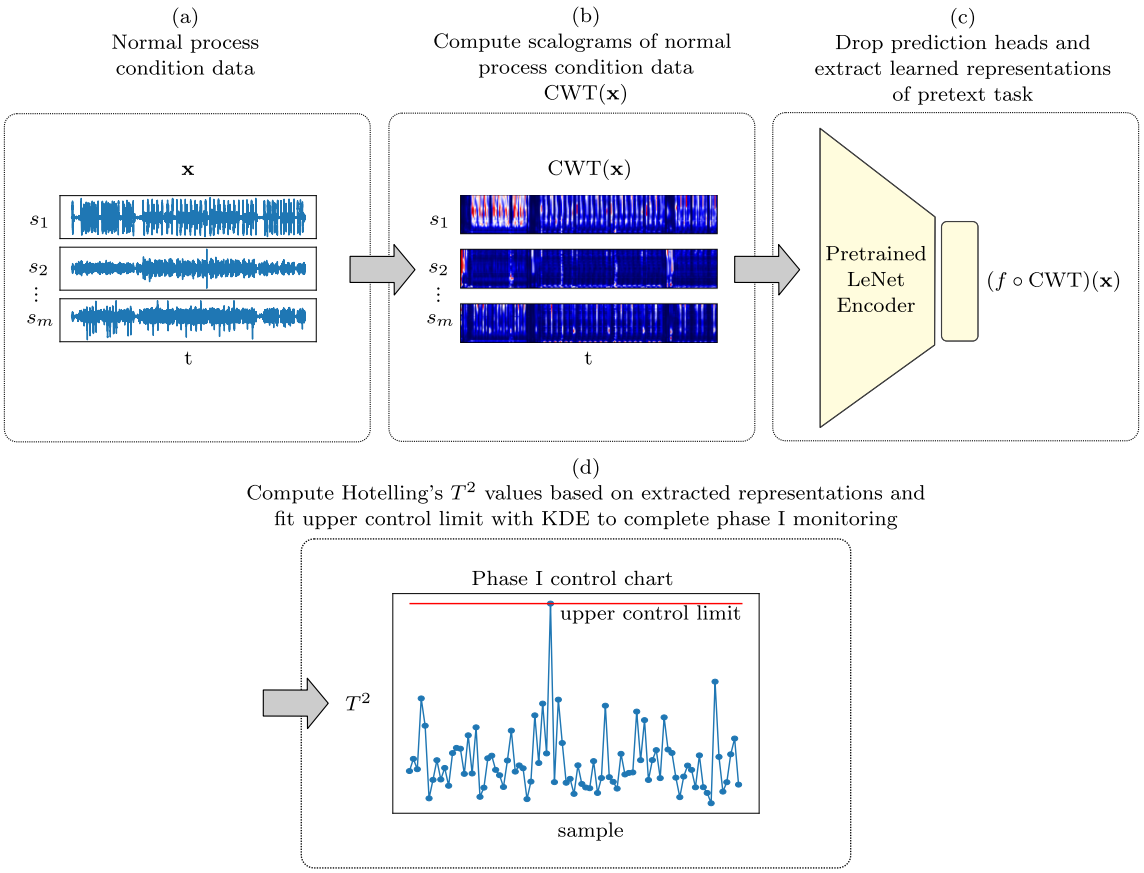


Fig. 8 Downstream task proposed in SSMSPC. **a** Normal process condition data are used as an input for **b** CWT to compute the scalograms. **c** The scalograms are fed to the pretrained LeNet-type encoder network from the pretext task to extract the learned representations. **d** Based on the encoded normal process condition data we perform conventional one-class classification using Hotelling's T^2. The control limits are fitted with KDE based on the resulting Hotelling's T^2 values

Downstream task

With the completion of the pretext task, the next step in the two-stage framework, as depicted in Fig. 3, consists in fitting a one-class classifier on top of the learned representations of the pretext task. In SSMSPC, we propose to use Hotelling's T^2 for this purpose. Figure 8 illustrates the individual components of the downstream task.

As opposed to the pretext task, in which the normal process condition data are augmented by a set of k different augmentation functions, the downstream task uses only the normal process condition data, as this is typical for the one-class classification setting.

Given a process sample $\mathbf{x} \in X_{\text{train}} \subset X$, the multivariate time series is transformed to the scalogram representation $\text{CWT}(\mathbf{x}) =: \widetilde{\mathbf{x}}$, and then fed through the pretrained LeNet-type encoder network to retrieve the learned representations $f(\widetilde{\mathbf{x}})$. Recall that the prediction heads g_{trans} and g_{loc} are discarded in this step, since they were only used for the pretext task. Based on the extracted features, we apply Hotelling's T^2, which is defined as

$$T^2 := (f(\widetilde{\mathbf{x}}) - \hat{\boldsymbol{\mu}})^\intercal \hat{\Sigma}^{-1} (f(\widetilde{\mathbf{x}}) - \hat{\boldsymbol{\mu}}) \in \mathbb{R}, \tag{6}$$

where $\hat{\boldsymbol{\mu}} \in \mathbb{R}^{r\times 1}$ and $\hat{\Sigma}^{-1} \in \mathbb{R}^{r\times r}$ represent the estimated mean and the inverse of the estimated covariance matrix of the learned representations $f(\widetilde{X}_{\text{train}}) \in \mathbb{R}^{l\times r}$, respectively. Consequently, for each $\mathbf{x} \in X_{\text{train}}$, a single T^2 value is computed that represents the processing cycle of the corresponding part. In order to compute the upper control limit, we follow the approach taken by Biegel et al. (2022a, 2022b). First, KDE is used to estimate the probability density function $\hat{\phi}_{T^2}$ based on the l computed T^2 values

$$\hat{\phi}_{T^2}(T^2) = \frac{1}{lh} \sum_i K\left(\frac{T^2 - T_i^2}{h}\right), \tag{7}$$

where h is the bandwidth and $K(\cdot)$ corresponds to the selected kernel function, in our case a Gaussian kernel. Next, we select a significance value α and compute the quantile function $\hat{\Phi}^{-1}_{T^2}$

of the corresponding cumulative distribution function $\hat{\Phi}_{T^2}$ of $\hat{\phi}_{T^2}$, to retrieve the upper control limit.

$$\hat{\Phi}_{T^2}^{-1}(1-\alpha) = \text{ucl}. \tag{8}$$

Note that α corresponds to the type I error or false alarm rate. Varying α translates into an increase or decrease of the type II error, i.e., the false negative rate. Thus, the specific choice of α is problem dependent. If there are no specific restrictions regarding, e.g., the false alarm rate, a possible way is to select the value for α that maximizes the F1-score on the validation set. Combining all the components presented so far, we now have a mapping to provide the anomaly score for a given time series input $\mathbf{x}$, and a corresponding threshold based on which the decision will be made whether a sample is flagged as normal or anomalous. Recall that this corresponds exactly to the desired mappings $h : \mathcal{D} \to \mathbb{R}$ and $b_{\text{ucl}} : \mathbb{R} \to \{0, 1\}$ as stated in "Problem statement" section.

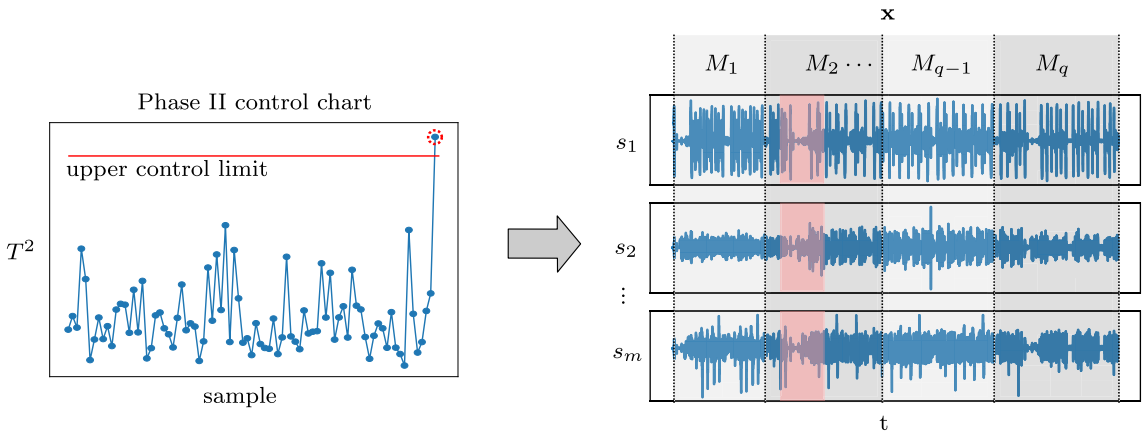
Control chart extension

Once a control chart signals an anomalous process condition, a machine operator needs to determine the root cause of the anomaly and decide which steps to take next (Qiu & Xie, 2021). However, by using the information provided in a control chart alone, it is not possible to infer which part of the process is deemed anomalous and therefore requires a time-intensive investigation of the process to find the potential root cause. According to Jackson (1991), a good monitoring approach needs to provide an answer to the question of what the problem of the process is if the control chart signals an anomalous process condition.

In order to accomplish this, it is necessary that the underlying model decisions are interpretable, i.e., a mechanism is required to understand the reasoning behind the decision-making process of the model (Ruff et al., 2021). When using time series data, this translates into highlighting the components of the multivariate time series that the model considers anomalous (Abdulaal et al., 2021; Li et al., 2021).
However, with respect to the problem statement considered in this paper, visualizing anomalies in the raw time series signal without any additional domain context makes it very hard for a machine operator to interpret the model decisions correctly. A machining process usually consists of a set of q different process modes $\{M_1, \ldots, M_q\}$, where each mode corresponds to a different process step in the processing cycle of the part. By collecting additional metadata during the processing cycle such as the Numerical Control (NC) lines of the machine program, it is possible to extract the start point for each process mode and thus segment the time series accordingly.
In SSMSPC, we suggest augmenting the conventional control chart perspective by an additional view in which (1) the anomalous time steps are highlighted and (2) the time series is segmented into the individual process modes. See Fig. 9 for an exemplary illustration of the extended control chart view.

Highlighting anomalies

For the purpose of highlighting the anomalous time steps, we use Gradient-weighted Class Activation Mapping (Grad-CAM) (Selvaraju et al., 2020), which is a simple and widely used technique to provide visual explanations for the deci-



(a) Anomalous process condition

(b) Interpretability of model decision

Fig. 9 Visualization of the proposed control chart extension in SSMSPC. **a** After processing a part, the control chart signals an anomalous process condition. **b** An additional view is displayed in which the multivariate time series is segmented into the individual process modes $\{M_1, \ldots, M_q\}$ by incorporating metadata provided by the NC lines of the processing cycle and highlighting the anomalous time steps. Thus, a machine operator is pointed directly to the process step in which the model found an anomalous process condition

sions of CNN-based models. The first step in Grad-CAM consists of computing the gradients of $h(\mathbf{x})$ with respect to the feature map activations $A^{\psi} \in \mathbb{R}^{u \times v}$ of the last convolutional layer of the corresponding LeNet-type encoder and use global average pooling on these gradients to obtain the neuron importance weights α_{ψ}

$$\alpha_{\psi} = \frac{1}{uv} \sum_{i} \sum_{j} \frac{\partial h(\mathbf{x})}{\partial A^{\psi}_{i,j}}, \forall \psi \in \{1, \ldots, \Psi\}. \tag{9}$$

To retrieve the Grad-CAM heatmap $\mathcal{H} \in \mathbb{R}^{u \times v}$ that contains the visual explanations, the next step involves the application of a ReLU on top of the linear combination of the neuron importance weights α_{ψ} and the feature map activations A^{ψ}

$$\mathcal{H} = \text{ReLU}\left(\sum_{\psi} \alpha_{\psi} A^{\psi} \right). \tag{10}$$

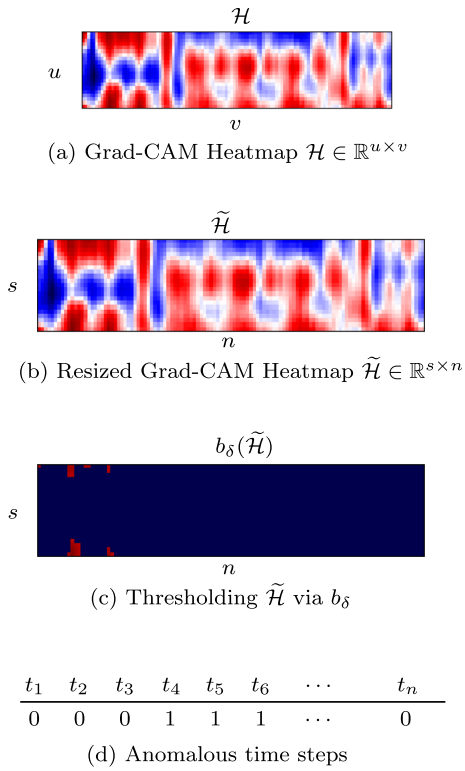
In order to obtain the anomalous time steps, we first resize the Grad-CAM heatmap $\mathcal{H}$ to $\widetilde{\mathcal{H}} \in \mathbb{R}^{s \times n}$. Recall that s represents the number of scales used when applying CWT and n stands for the number of time steps. Next, we look for a threshold function $b_{\delta} : \mathbb{R}^{s \times n} \rightarrow \mathbb{R}^{s \times n}$ such that

$$b_{\delta}(\widetilde{\mathcal{H}}) = \begin{Bmatrix} \widetilde{\mathcal{H}}_{i,j}, & \text{if } \widetilde{\mathcal{H}}_{i,j} > \delta \\ 0, & \text{if } \widetilde{\mathcal{H}}_{i,j} \leq \delta \end{Bmatrix} \forall i, j. \tag{11}$$

To find the threshold δ, we follow the exact same idea as demonstrated in the previous section. Thus, we first compute the probability density function $\hat{\phi}_{\widetilde{\mathcal{H}}}$ using KDE. Selecting a significance value $\alpha_{\widetilde{\mathcal{H}}}$ and evaluating the quantile function at the respective position $\hat{\Phi}^{-1}_{\widetilde{\mathcal{H}}}(1 - \alpha_{\widetilde{\mathcal{H}}})$, we find the desired threshold δ. Lastly, we simply sum up the columns of $b_{\delta}(\widetilde{\mathcal{H}})$, set all nonzero values to 1 and thus retrieve a binary value for each time step indicating whether it is to be considered anomalous

$$\mathbb{1}_{b_{\delta}(\widetilde{\mathcal{H}})_{i,j} > 0} \left(\sum_{i} b_{\delta}(\widetilde{\mathcal{H}})_{i,j} \right) \forall j \in \{1, \ldots, n\}. \tag{12}$$

Figure 10 visualizes the aforementioned steps.



(a) Grad-CAM Heatmap $\mathcal{H} \in \mathbb{R}^{u \times v}$

(b) Resized Grad-CAM Heatmap $\widetilde{\mathcal{H}} \in \mathbb{R}^{s \times n}$

(c) Thresholding $\widetilde{\mathcal{H}}$ via b_{δ}

(d) Anomalous time steps

Fig. 10 Scheme for localizing anomalies via Grad-CAM. **a** First, we compute the Grad-CAM heatmap $\mathcal{H}$ and **b** resize it to the initial input dimension $\widetilde{\mathcal{H}}$. **c** Then we apply a threshold function b_{δ} to each value in $\widetilde{\mathcal{H}}$. (d) The columns of $b_{\delta}(\widetilde{\mathcal{H}})$ with nonzero values mark the anomalous time steps

Experiments

In this section, we compare the performance of SSMSPC with state-of-the-art anomaly detection baselines using two real-world CNC-milling datasets. The experiments have been conducted on an AMD Ryzen 9 3900X processor with 3.8 GHz, 12 cores and 24 threads using a GeForce RTX 2080 Ti GPU. We use TensorFlow (Abadi et al., 2015) and scikit-learn (Pedregosa et al., 2011) for implementation purposes.

Bosch CNC-milling dataset

The dataset used for the first experiment is the recently published Bosch CNC-milling dataset (Tnani et al., 2022). This dataset contains vibration data from 3 different CNC machines over a three-year period collected with a triaxial accelerometer using a sampling rate of 2 kHz. The machines produce different aluminum parts with a total of 15 different tool operations OP00 to OP14 that are used on all 3 machines. Here, each OP corresponds to a different tool with unique process parameters. For each machine, the respective OP data are labeled good or bad, indicating whether the data correspond to a normal or an anomalous process condition. Table 1 provides an overview of the dataset structure.

Most of the process steps correspond to drilling operations of varying length. In addition, it is observable that there is a strong imbalance in the dataset with normal process condition data corresponding to $\approx 96\%$ of the available data, which is typical for discrete manufacturing processes. The authors of the dataset note that the process steps have been shuffled and some are not included in the dataset for confidentiality purposes. Thus, the order of the process steps as displayed

Table 1 Overview of the dataset structure of the Bosch CNC-milling dataset, adapted from Tnani et al. (2022)

Process	Description	Duration (s)	Machine 01		Machine 02		Machine 03		$\sum$ Machines	
			Good	Bad	Good	Bad	Good	Bad	Good	Bad
OP00	Step Drill	≈ 132	29	0	26	1	28	0	83	1
OP01	Step Drill	≈ 29	38	2	51	3	47	2	136	7
OP02	Drill	≈ 42	45	1	52	2	51	1	148	4
OP03	Step Drill	≈ 77	27	1	28	1	13	0	68	2
OP04	Step Drill	≈ 64	34	2	29	3	42	2	105	7
OP05	Step Drill	≈ 18	34	4	42	2	38	0	114	6
OP06	Step Drill	≈ 91	19	4	30	0	35	0	84	4
OP07	Step Drill	≈ 24	43	4	52	3	53	3	148	10
OP08	Step Drill	≈ 37	31	3	42	4	39	0	112	7
OP09	Straight Flute	≈ 102	35	1	43	0	35	0	113	1
OP10	Step Drill	≈ 45	29	4	44	2	39	1	112	7
OP11	Step Drill	≈ 59	17	4	31	2	20	0	68	6
OP12	Step Drill	≈ 46	34	3	42	2	42	0	118	5
OP13	T-Slot Cutter	≈ 32	43	0	51	0	48	0	142	0
OP14	Step Drill	≈ 34	27	1	54	2	0	0	81	3
		Total	485	34	617	27	530	9	1632	70

in Table 1 does not reflect the actual process flow of the produced parts (Tnani et al., 2022).

We investigate a setting that is in alignment with the problem statement formulated in "Problem statement" section. As a first step, we merge the data of each individual machine OP-wise and thus act as if the data originate from a single imaginary machine, see the last column of Table 1. The rationale for doing this is to enlarge the amount of available data. In the next step, we crop the data within each OP to have exactly the same length. This is done by cutting off all data points that are exceeding the minimum OP length within the respective OP class.

The scenario that we address centers around a multimodal process, i.e., a process that contains more than one OP. Our objective with this scenario is to demonstrate the practical applicability of SSMSPC in a real-world setting that is representative of a typical discrete manufacturing process. We concatenate OP01, OP02, OP07 horizontally and cut off all samples that exceed the minimum number of samples of these three OPs. We decided to select these OPs since concatenating them results in the largest possible dataset for a process with three modes. After following the aforementioned steps we obtain the dataset that forms the basis for this scenario $\mathcal{D}_S \subset \mathbb{R}^{136084\times 3}$, with $|\mathcal{D}_S| = 143$. We then split the data using a train, validation, test split of 60%, 10% and 30%, respectively. Table 2 displays the number of samples for each split.

Table 2 Overview of the data split size for $\mathcal{D}_S$ including the number of normal and anomalous samples

Split	Normal	Anomalous	Total
Train	85	–	85
Validation	10	5	15
Test	30	13	43

Pretext task: Bosch CNC-milling dataset

As stated in "Pretext task" section, we use the normal process condition data for the pretext task and augment them with the presented *Identity*, *CutPaste*, *MeanShift* and *MissingSignal* augmentations using a total of 5 windows. Thus, the initial train dataset increases by a factor of 4 to a total of 340 samples. Following this, we apply CWT on the augmented dataset using Morlet wavelets with $s = 128$ scales. Then we resize the resulting scalograms down to a size of 128×512 and apply min-max scaling. Doing this we obtain the dataset for the pretext task $\mathcal{D}_S^{\text{Pretext}} \subset \mathbb{R}^{128\times 512\times 3}$, where $|\mathcal{D}_S^{\text{Pretext}}| = 340$.

Model architecture Regarding the model architecture, we use a LeNet-type encoder network f with two prediction heads g_{loc} and g_{trans} attached to it, as shown in Fig. 4c. Given the small dataset size in this scenario we use a very compact architecture to prevent overfitting.

The convolutional modules of the LeNet-type encoder network consist of a simple 2D convolutional layer followed by a batch normalization (Ioffe & Szegedy, 2015) layer, leaky ReLU activation and a subsequent 2D max-pooling layer. We

apply a total of three convolutional modules using $8 \times (7 \times 7)$ filters for the first module, $16 \times (7 \times 7)$ filters for the second module and $32 \times (7 \times 7)$ for the third. The convolutional modules are followed by two dense modules consisting of a dense layer, a batch normalization layer and leaky ReLU activation, respectively. Here, the first dense layer uses 32 neurons whereas the second uses 16.

The prediction heads are mostly equivalent in their structure, i.e., each head consists of two dense layers with 16 and 8 neurons, respectively, followed by a batch normalization layer and leaky ReLU activation. The heads differ only in the last dense and softmax layer, where g_{loc} uses 6 neurons and g_{trans} uses 4 neurons. Table 3 provides further insights into the different layers used.

Model training We train the model on the proposed Location + Transformation pretext task for 20 epochs with a batch size of 8 and a cosine learning rate schedule (Loshchilov & Hutter, 2017) with an initial learning rate of 10^{-4}. We use a leakiness of $\beta = 10^{-1}$ for the leaky ReLU activations. With respect to optimization, we apply the Adam optimizer (Kingma & Ba, 2015) and use weight decay with $\lambda = 10^{-3}$ for regularization.

Table 3 Model architecture used for the pretext task

	Layer type	Output shape	Parameters
f	Input Layer	(None, 128, 512, 3)	0
	Conv2D	(None, 122, 506, 8)	1184
	BatchNorm	(None, 122, 506, 8)	32
	LeakyReLU	(None, 122, 506, 8)	0
	MaxPool2D	(None, 61, 253, 8)	0
	Conv2D	(None, 55, 247, 16)	6288
	BatchNorm	(None, 55, 247, 16)	64
	LeakyReLU	(None, 55, 247, 16)	0
	MaxPool2D	(None, 27, 123, 16)	0
	Conv2D	(None, 21, 117, 32)	25120
	BatchNorm	(None, 21, 117, 32)	128
	LeakyReLU	(None, 21, 117, 32)	0
	MaxPool2D	(None, 10, 58, 32)	0
	Flatten	(None, 18560)	0
	Dense	(None, 32)	593952
	BatchNorm	(None, 32)	128
	LeakyReLU	(None, 32)	0
	Dense	(None, 16)	528
	BatchNorm	(None, 16)	64
	LeakyReLU	(None, 16)	0
g_{loc}	Dense	(None, 16)	272
	BatchNorm	(None, 16)	64
	LeakyReLU	(None, 16)	0
	Dense	(None, 8)	136
	BatchNorm	(None, 8)	32
	LeakyReLU	(None, 8)	0
	Dense	(None, 6)	54
	Softmax	(None, 6)	0
g_{trans}	Dense	(None, 16)	272
	BatchNorm	(None, 16)	64
	LeakyReLU	(None, 16)	0
	Dense	(None, 8)	136
	BatchNorm	(None, 8)	32
	LeakyReLU	(None, 8)	0
	Dense	(None, 4)	45
	Softmax	(None, 4)	0

Downstream task: Bosch CNC-milling dataset

With respect to the subsequent downstream task, we first apply CWT on each split of $\mathcal{D}_{\text{S}}$ using the same configuration as in the pretext task. We then resize the scalograms to 128×512 and apply min-max scaling based on the train split. Doing this, we obtain the required dataset for the downstream task $\mathcal{D}_{\text{S}}^{\text{Downstream}} \subset \mathbb{R}^{128 \times 512 \times 3}$, where $|\mathcal{D}_{\text{S}}^{\text{Downstream}}| = 143$. Recall from "Downstream task" section, that we use the train split containing only normal process condition data for fitting the Hotelling's T^2 one-class classifier.

Model architecture As shown in Fig. 8, we discard the prediction heads g_{loc} and g_{trans} and keep only the pretrained encoder f to extract the learned representations using the output of the last dense layer. We then attach a Hotelling's T^2 layer to it. Table 4 shows the model architecture for the downstream task.

Model training For the one-class classification training phase, we pass the train data of $\mathcal{D}_{\text{S}}^{\text{Downstream}}$ through the frozen encoder network f and fit the required parameters for Hotelling's T^2, i.e., $\hat{\boldsymbol{\mu}} \in \mathbb{R}^{16 \times 1}$ and $\hat{\Sigma}^{-1} \in \mathbb{R}^{16 \times 16}$ based on the learned representations. We then use these parameters to compute the Hotelling's T^2 values for the train data. Following this, we fit the upper control limit as described in "Downstream task" section using a significance value of $\alpha = 0.01$, to obtain a low false alarm rate. Thus, we take the 0.99 quantile of $\hat{\phi}_{T^2}$ as the upper control limit. Recall that this step completes the offline monitoring phase, i.e., phase I with respect to the conventional SPC framework. All hyperparameters, including those for pretext training, are selected using grid search based on the performance on the validation set in the downstream task. The final performance is reported based on the hold out test set.

Baselines: Bosch CNC-milling dataset

We compare the performance of SSMSPC with state-of-the-art baselines from the realms of shallow, deep and self-supervised anomaly detection. We evaluate the baselines on (1) $\mathcal{D}_{S}^{\text{Downstream}}$, i.e., CWT features and (2) statistical

Table 4 Model architecture used for the downstream task

	Layer type	Output shape	Parameters
f	Input Layer	(None, 128, 512, 3)	0
	Conv2D	(None, 122, 506, 8)	1184
	BatchNorm	(None, 122, 506, 8)	32
	LeakyReLU	(None, 122, 506, 8)	0
	MaxPool2D	(None, 61, 253, 8)	0
	Conv2D	(None, 55, 247, 16)	6288
	BatchNorm	(None, 55, 247, 16)	64
	LeakyReLU	(None, 55, 247, 16)	0
	MaxPool2D	(None, 27, 123, 16)	0
	Conv2D	(None, 21, 117, 32)	25120
	BatchNorm	(None, 21, 117, 32)	128
	LeakyReLU	(None, 21, 117, 32)	0
	MaxPool2D	(None, 10, 58, 32)	0
	Flatten	(None, 18560)	0
	Dense	(None, 32)	593952
	BatchNorm	(None, 32)	128
	LeakyReLU	(None, 32)	0
	Dense	(None, 16)	528
T^2	Hotelling's T^2	(None, 1,1)	0

features. For the statistical features, we compute *root-mean square*, *peak-to-peak*, *interquartile range*, *mean*, *standard deviation*, *kurtosis*, *skewness* and *median absolute deviation* for each sensor of the raw time series data in $\mathcal{D}_S$. Following this, we scale the data to have zero mean and unit variance based on the train set. Thus, for the evaluation with statistical features we obtain the following dataset $\mathcal{D}_S^{\text{stat}} \subset R^{24}$, where $|\mathcal{D}_S^{\text{stat}}| = 143$.

Shallow baselines For the shallow baselines, we choose One-Class Support Vector Machine (OC-SVM) (Schölkopf et al., 1999), IF, Principal Component Analysis (PCA) and Kernel-PCA, as they are regularly used for the comparison of anomaly detection methods.

Since the samples in $\mathcal{D}_S^{\text{Downstream}}$ represent image data, we first train a LeNet-type AE (LeNet-AE) on the train data and then use the encoder part to transform the scalograms into the latent representation. The shallow baselines are then applied on top of the extracted features. This scheme is commonly used in literature, see, e.g., Golan and El-Yaniv (2018). The encoder of the LeNet-AE has the same architecture as the LeNet-type encoder network that is used in SSMSPC. The decoder is constructed symmetrically to the encoder. We just replace convolutions with deconvolutions and max-pooling with upsampling.

Regarding the evaluation on $\mathcal{D}_S^{\text{stat}}$, we apply the shallow baselines directly on top of the statistical features.

We use the scikit-learn implementation for the presented baselines. Hyperparameters are selected using grid search based on the performance on the validation set. Table 5 provides an overview of the selected hyperparameters.

Table 5 Hyperparameters for shallow baselines

	Hyperparameter	$\mathcal{D}_S^{\text{Downstream}}$	$\mathcal{D}_S^{\text{stat}}$
OC-SVM	kernel	RBF	RBF
	γ	0.25	0.25
	ν	0.005	0.005
IF	n_estimators	10	10
	max_samples	32	80
PCA	n_components	12	23
Kernel-PCA	n_components	15	13
	kernel	cosine	cosine

Deep baselines As representatives for deep anomaly detection baselines, we choose DAGMM (Zong et al., 2018), DeepSVDD (Ruff et al., 2018), PatchCore (Roth et al., 2022), and the LeNet-AE presented in "Baselines: Bosch CNC-milling dataset" section.

DAGMM utilizes a deep AE and combines both, the latent representation and the reconstruction error of the input which is then fed to a Gaussian mixture model. The mixture model and the AE are trained in an end-to-end fashion. DAGMM is designed to work on tabular data and as such we evaluate it only on $\mathcal{D}_S^{\text{stat}}$.

DeepSVDD uses the encoder of a pretrained AE as weight initialization to extract features from image data. The encoder is then trained end-to-end on the one-class classification objective to map the data into a minimum-volume hypersphere (Ruff et al., 2018).

PatchCore is a recently published method that has achieved state-of-the-art results on the MVTec AD (Bergmann et al., 2019) industrial image anomaly detection benchmark. The intriguing property of PatchCore is that it does not comprise an actual training phase in the sense of gradient-based optimization, but it leverages the learned representations of mid-level feature representations of a ResNet (He et al., 2015) architecture trained on ImageNet (Krizhevsky et al., 2012). Specifically, given an image dataset for training that contains only normal data, PatchCore extracts locally aware patch features and stores them in a memory bank, by passing the training data through the network. With this memory bank in place, PatchCore computes the anomaly score of a given test image based on the maximum distance score between test patch-features in its patch collection and each corresponding nearest neighbor in the memory bank (Roth et al., 2022). For PatchCore, DAGMM and DeepSVDD, we use the official implementation that has been made publicly available by the authors with their respective publication. Table 6 presents the selected hyperparameters.

Table 6 Hyperparameters for deep baselines

	Hyperparameter	$\mathcal{D}_S^{\text{Downstream}}$	$\mathcal{D}_S^{\text{stat}}$
DAGMM	batch_size	–	8
	learning_rate	–	10^{-4}
	epochs	–	1000
DeepSVDD	batch_size	8	–
	learning_rate	10^{-3}	–
	epochs	50	–
	weight_decay	10^{-6}	–
	ν	0.005	–
PatchCore	batch_size	2	–
	patch_size	3	–
LeNet-AE	batch_size	8	–
	learning_rate	10^{-4}	–
	epochs	100	–

Table 7 Hyperparameters for self-supervised baselines

	Hyperparameter	$\mathcal{D}_S^{\text{Downstream}}$
RotNet	batch_size	8
	learning_rate	10^{-1}
	epochs	100
	weight_decay	$5 \cdot 10^{-4}$
	input_size	128×128
+ PCA	n_components	5
+Kernel-PCA	n_components	3
	kernel	cosine
+OC-SVM	kernel	RBF
	γ	0.5
	ν	0.005
+IF	n_estimators	10
	max_samples	64
NeuTraL AD	batch_size	8
	learning_rate	10^{-3}
	epochs	20
	transformations	15
GeoTrans	batch_size	8
	learning_rate	10^{-3}
	epochs	3
	input_size	128×128

Self-supervised baselines Considering the self-supervised anomaly detection baselines, we choose RotNet (Gidaris et al., 2018), NeuTraL AD and GeoTrans. Regarding RotNet, we follow two different approaches. First, we treat it as a feature extractor and use the learned representations from the rotation prediction pretext task as an input for PCA, Kernel-PCA, OC-SVM and IF. Second, we follow the scheme presented by Hendrycks et al. (2019) and use the softmax probabilities of RotNet to determine if a sample is normal or anomalous. As with the deep baselines, we use the publicly available implementation of the respective self-supervised baseline to conduct the experiments. Table 7 displays the selected hyperparameters.

Results: Bosch CNC-milling dataset

In Table 8, we present the results of our experiments. We report mean and standard deviation of the Area under the Receiver Operating Characteristic (AUROC) curve over 10 different random seeds. The Receiver Operating Characteristic (ROC) curve of a classifier is a two-dimensional graph in which the recall is plotted over the false alarm rate for every possible decision threshold in the test set (Fawcett, 2006). The area under this curve represents the AUROC which is a threshold-independent evaluation measure that (1) provides an overall assessment of the capabilities of a classifier and (2) allows to compare the peformance of different classifiers (Ding et al., 2014). The AUROC value ranges from 0 to 1 (or 0% to 100%) and can be defined as the following mean computed across all anomalous and normal data pairs in the test set (Aggarwal, 2017; Campos et al., 2016):

$$\text{AUROC} := \underset{\mathbf{a}\in A, \mathbf{n}\in N}{\text{mean}} \begin{cases} 1, & \text{if } h(\mathbf{a}) > h(\mathbf{n}) \\ \frac{1}{2}, & \text{if } h(\mathbf{a}) = h(\mathbf{n}) \\ 0, & \text{if } h(\mathbf{a}) < h(\mathbf{n}) \end{cases} \tag{13}$$

where N and A represent the respective sets of normal and anomalous data in the test set and $h(\cdot)$ corresponds to the anomaly score produced by the classifier. Intuitively, the AUROC can be interpreted as the probability that a classifier ranks a randomly chosen anomalous sample higher than a randomly chosen normal sample (Bradley, 1997; Fawcett, 2006). Due to the aforementioned properties, the AUROC has established itself as the standard measure for comparing the performance of anomaly detection models (Abati et al., 2019; Carrara et al., 2020; Cheng & Vasconcelos, 2021; Hu et al., 2020; Ruff et al., 2021; Shen et al., 2021; Wu et al., 2021).

Focusing on the results of our experiment, we see that SSM-SPC achieves a perfect score on the given task outperforming all baselines except LeNet-AE and PatchCore which have on-par performance. We can see that the shallow baselines perform well on the statistical features, especially OC-SVM and PCA with 91.5% and 89.7% AUROC, respectively. Perhaps surprisingly, the performance of the shallow baselines

Table 8 Results on the Bosch CNC-milling dataset

Category	Feature	Method	AUROC
Shallow	Statistical	OC-SVM	91.5 ± 4.4
		PCA	89.7 ± 4.7
		Kernel-PCA	88.0 ± 4.8
		IF	79.1 ± 8.7
	CWT	LeNet + PCA	75.9 ± 10.3
		LeNet + Kernel-PCA	78.3 ± 8.7
		LeNet + OC-SVM	81.3 ± 9.8
		LeNet + IF	54.3 ± 10.8
Deep	Statistical	DAGMM	92.8 ± 5.5
	CWT	DeepSVDD	99.9 ± 0.2
		LeNet-AE	$\mathbf{100.0 \pm 0.0}$
		PatchCore	$\mathbf{100.0 \pm 0.0}$
Self-supervised	CWT	NeuTraL AD	97.9 ± 3.6
		GeoTrans	94.5 ± 6.7
		RotNet	56.0 ± 9.9
		RotNet + PCA	94.6 ± 5.6
		RotNet + Kernel-PCA	84.7 ± 12.0
		RotNet + OC-SVM	98.4 ± 1.6
		RotNet + IF	97.6 ± 3.1
		SSMSPC (ours)	$\mathbf{100.0 \pm 0.0}$

We report the mean and standard deviation for AUROC over 10 different random seeds. State-of-the-art baselines are investigated using both, statistical features and CWT features. The best results are given in bold

on the CWT features is much lower, with IF being only barely able to exceed the random guessing AUROC baseline of 50%. In terms of the deep baselines, we see an overall strong performance. Deep SVDD, LeNet-AE and PatchCore are very competitive and perform similar to SSMSPC with 99.9%, 100.0% and 100.0% AUROC, respectively. DAGMM shows the best performance among the baselines that are evaluated on statistical features with 92.8% AUROC.

Regarding the self-supervised anomaly detection methods, we can see that RotNet, when evaluated with the scheme of Hendrycks et al. (2019), is performing close to the random guessing AUROC baseline. However, when used as a feature extractor we see that the performance strongly increases with RotNet + OC-SVM achieving 98.4% AUROC. GeoTrans and NeuTraL AD show a performance in the range of RotNet when used as a feature extractor, with 94.5% and 97.9% AUROC, respectively.

Even though the general evaluation in this paper is based on AUROC, it is nonetheless necessary for practical applications to select a fixed threshold. Thus, we include threshold-dependent performance metrics to provide further insights into the quality of the threshold selection scheme in SSMSPC. Table 9 shows the performance of SSMSPC with respect to mean and standard deviation of Precision, Recall and F1-Score using KDE to fit the control limits of the process as described in "Downstream task" section. We present the performance for varying choices of α. With $\alpha = 0.01$, SSMSPC achieves an average F1-Score of 91.7%, while identifying all the anomalies present in the hold out test set across all random seeds.

Table 9 Results for Precision, Recall and F1-score of SSMSPC for varying levels of α on the Bosch CNC-milling dataset

α	Precision	Recall	F1-score
0.050	65.6 ± 6.6	100.0 ± 0.0	79.0 ± 5.1
0.025	72.3 ± 6.2	100.0 ± 0.0	83.8 ± 4.0
0.010	84.9 ± 4.0	100.0 ± 0.0	91.7 ± 2.2

We report mean and standard deviation over 10 different random seeds

Localizing anomalies

Figure 11 presents qualitative results of the proposed control chart extension in SSMSPC based on anomalous samples taken from the validation set. We highlight the anomalous process modes identified by our model and compare it to the true labels as given in the dataset. As is shown, SSMSPC is able to identify the anomalous process modes and thus provides an effective support for a machine operator to find the root cause of the anomalous process condition.

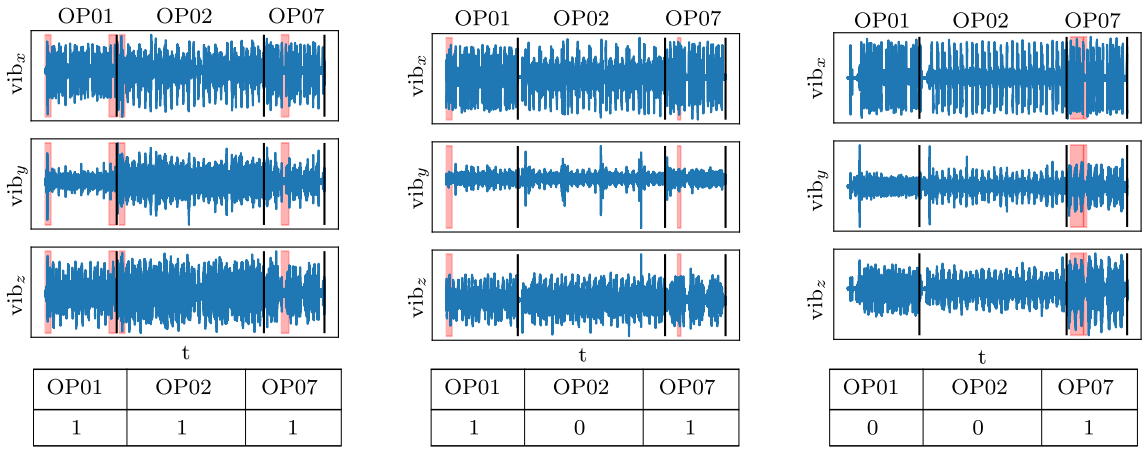


Fig. 11 Qualitative visualization of the control chart extension proposed in SSMSPC. We present samples from the validation set and show in the tables below the true labels for each mode of the process (anomaly = 1, normal = 0). The highlighted regions are the anomalous components of the respective process steps that have been identified by the model

Table 10 Effects of adding augmentations on the performance of the downstream task

Method	Augmentation	AUROC
SSMSPC	base	$\mathbf{100.0 \pm 0.0}$
	+ *PointOutlier*	98.7 ± 1.6
	+ *Trend*	98.3 ± 2.7
	+ *Gaussian*	96.0 ± 3.9

We report mean and standard deviation for AUROC over 10 different random seeds. The best results are given in bold

Table 11 Effects of removing augmentations on the performance in the downstream task

Method	Augmentation	AUROC
SSMSPC	base	$\mathbf{100.0 \pm 0.0}$
	w/o *MeanShift*	99.5 ± 0.7
	w/o *Identity*	99.3 ± 1.4
	w/o *CutPaste*	99.1 ± 1.4
	w/o *MissingSignal*	98.7 ± 1.2

We report mean and standard deviation for AUROC over 10 different random seeds. The best results are given in bold

Ablation study

We perform an ablation study to provide further insights into the individual components of SSMSPC. First, we analyze the effects of adding and removing augmentations for the pretext task. Second, we compare the performance of the proposed Location + Transformation prediction pretext task with Location prediction and Transformation prediction as individual pretext tasks. Third, we vary the number of windows that is used for the augmentations in the pretext task to see how this affects performance.

Augmentations We define three additional augmentations, which we refer to as *PointOutlier*, *Trend* and *Gaussian* in order to assess whether additional augmentations have a positive effect on the performance in the downstream task. For the *PointOutlier* augmentation, we select two random points in a window and add the maximum value of the whole time series to these two points. Regarding the *Trend* augmentation, we emulate a growing shift in the standard deviation within a selected window. *Gaussian* selects a time series segment within a window and adds random numbers sampled from a Gaussian distribution with specified mean and standard deviation. Table 10 displays the results. We observe that the additional augmentations lead to a decrease in performance. The strongest decline can be seen with the *Gaussian* augmentation. In addition to adding new augmentations we also investigate the effects of removing the proposed augmentations one after another, to understand which of the augmentations is most important for the performance. The results are reported in Table 11. We observe that the *MissingSignal* augmentation is the most important component, leading to the strongest decline in performance upon omission, followed by *CutPaste*. The decline observed by removing *MeanShift* or *Identity* is less strong but nonetheless existing and thus justifies their inclusion into SSMSPC.

Different pretext tasks To demonstrate the meaningfulness of the proposed Location + Transformation prediction pretext task and the corresponding loss function, we investigate the performance of Location prediction and Transformation prediction as individual pretext tasks. This is done by simply removing one of the prediction heads and then training SSMSPC on the remaining pretext task. All other components of SSMSPC are left unchanged. Table 12 shows the results. It

Table 12 Effects of using Transformation prediction and Location prediction as pretext tasks

Method	Pretext task	AUROC
SSMSPC	Loc. + Trans. prediction	**100.0 ± 0.0**
	Transformation prediction	99.2 ± 1.6
	Location prediction	97.9 ± 1.7

We report mean and standard deviation for AUROC over 10 different random seeds. The best results are given in bold

Table 13 Effects of varying the number of windows used for generating the dataset for the pretext task

Method	#Windows	AUROC
SSMSPC	3	99.5 ± 0.7
	5	**100.0 ± 0.0**
	7	98.9 ± 1.5

We report mean and standard deviation for AUROC over 10 different random seeds. The best results are given in bold

can be seen that the performance of SSMSPC decreases when trained solely on either Location prediction or Transformation prediction as pretext task. Hence, the representations learned by Location + Transformation prediction are more effective.

Number of windows We investigate the effects of varying the number of windows on the performance of SSMSPC. Table 13 shows the results. We see that SSMSPC is fairly robust with respect to varying the number of windows, which is an important feature for practical purposes.

CiP-DMD dataset

The second experiment in this paper uses parts of the Center for industrial Productivity - Discrete Manufacturing Dataset (CiP-DMD), which is a novel benchmark dataset for discrete manufacturing processes that we are going to publish soon. The specific dataset that we consider originates from a CNC-milling process in which steel cylinder bottoms are produced for pneumatic cylinders. Since this dataset is not yet published, we provide a short introduction of the process and give some insights into the dataset structure.

Process description

The test bed used for the development of the dataset is a Deckel-Maho DMC-50H horizontal CNC-milling machine, see Fig. 12.

The machining space of the DMC-50H is equipped with a rotating tower on which special fixtures are mounted that can hold several parts at once. Figure 13 provides a visualization for both sides of the rotating tower.

Fig. 12 DMC-50H CNC-milling machine used as a test bed within the CiP-DMD dataset

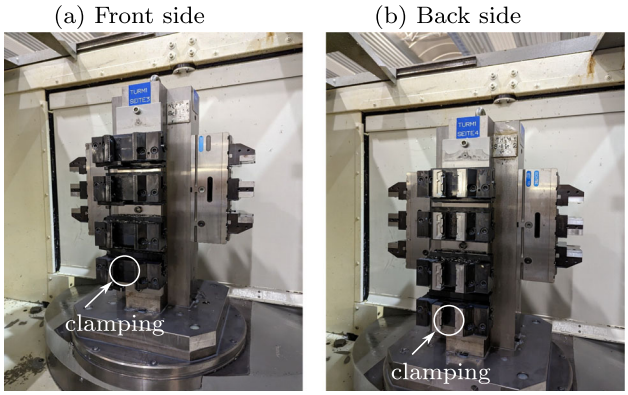


Fig. 13 **a** Front side and **b** back side of the rotating tower in the machining space of the DMC-50H test bed with special fixtures to hold raw material and semi-finished parts, respectively. The white circles indicate the positions where the parts are clamped

At the beginning of a machining cycle, two distinct parts, i.e., a piece of raw material and a semi-finished part are clamped to the front and back side of the rotating tower, respectively. Upon completion of a machining cycle, the process outputs a semi-finished part and a finished part. Note that for the sake of simplicity regarding the creation of the dataset, we used only one part per tower side. Since the machine processes two distinct parts per cycle, we refer to the part at the front side of the rotating tower as part 1 (P1) and to the part at the back side as part 2 (P2). Figure 14 provides an illustration of the respective production stages for P1 and P2.

Regarding the machining process itself, it is worth mentioning that the process for P1 consists of 9 distinct process modes whereas the process for P2 consists of 4. The total duration of one machining cycle amounts to roughly 5 minutes. More precisely, the process takes 199 seconds to complete P1 and another 113 seconds to complete P2.

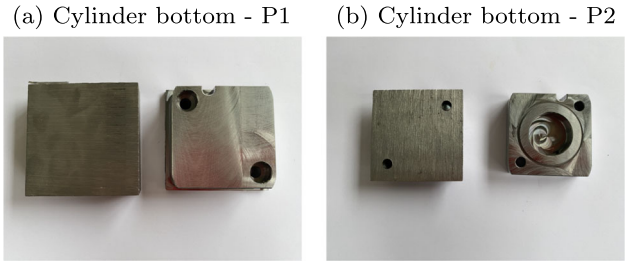


Fig. 14 **a** P1 before and after processing. This corresponds to the conversion from raw material to a semi-finished part. **b** P2 before and after processing. This corresponds to the conversion from a semi-finished part to a finished part

Fig. 15 Accelerometer used for data acquisition

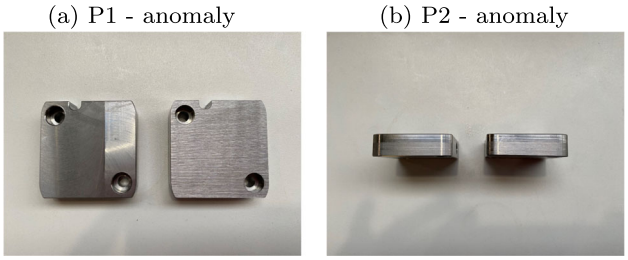


Fig. 16 Anomalies for P1 and P2. **a** Comparison of normal P1 (left) and anomalous P1 (right). **b** Comparison of normal P2 (left) and anomalous P2 (right)

Table 14 Overview of the data split size for $\mathcal{D}_{\text{P1}}$ and $\mathcal{D}_{\text{P2}}$ including the number of available normal and anomalous samples

	Split	Normal	Anomalous	Total
$\mathcal{D}_{\text{P1}}$	Train	465	–	465
	Validation	67	10	77
	Test	204	30	234
$\mathcal{D}_{\text{P2}}$	Train	465	–	465
	Validation	68	9	77
	Test	204	29	233

Dataset structure

The DMC-50H test bed is equipped with a triaxial accelerometer having a sampling rate of 2.5 kHz that is mounted directly on the spindle. The sensor is connected to a data acquisition card that is plugged into an industrial PC which is located on top of the machine, see Fig. 15 for an illustration of the sensor positioning. In addition to the process data, we also recorded metadata from the machine control such as the NC lines of the machine program with a sampling rate of 5Hz. For the purpose of this experiment, we recorded a total of 776 machining cycles between November 2022 and March 2023. Specifically, we have 776 samples for P1 and 775 samples for P2. Note that the discrepancy in the total number of samples between P1 and P2 stems from the fact that we have one air cut for P2 which we did not include in the dataset.

For P1, 736 samples correspond to normal processing cycles, whereas for P2 we have a total of 737 normal processing cycles. The remaining cycles correspond to anomalous process behavior.

We incorporated two realistic process anomalies which need to be detected and located during the process. The first anomaly targets P1 and results from a piece of raw material that has been sawed off too short. As a consequence, the cutting tool does not remove any chips during the face milling process step. The result of this anomalous process is a semi-finished part with an unprocessed surface. The second anomaly targets P2 and corresponds to a falsely clamped part. Due to this clamping error, the face milling step removes more chips from one side of the semi-finished part than from another. The finished cylinder bottom is thus crooked. Figure 16 illustrates the two anomalies.

We refer to the dataset of P1 as $\mathcal{D}_{\text{P1}}$ and to the dataset of P2 as $\mathcal{D}_{\text{P2}}$. Given the processing time of 199 seconds for P1 and a sampling rate of 2.5 kHz, we have a total of 497500 samples for every axis of the sensor. Thus, $\mathcal{D}_{\text{P1}} \subset \mathbb{R}^{497500\times 3}$, with $|\mathcal{D}_{\text{P1}}| = 776$. Conversely, for P2 with a processing time of 113 seconds, we have a total of 282500 samples per axis of the sensor, i.e., $\mathcal{D}_{\text{P2}} \subset \mathbb{R}^{282500\times 3}$, where $|\mathcal{D}_{\text{P2}}| = 775$. We split the data using the same train, val and test split as we did in the first experiment, i.e., 60%, 10%, and 30%. Note that, as before, the training data comprise only normal process data. Table 14 shows the respective number of samples for $\mathcal{D}_{\text{P1}}$ and $\mathcal{D}_{\text{P2}}$.

As in the previous experiment, we compare the performance of the selected baselines with statistical features and CWT features. The statistical features are computed according to the description in "Baselines: Bosch CNC-milling dataset" section with respect to each sensor in the raw time series data for $\mathcal{D}_{P1}$ and $\mathcal{D}_{P2}$. Following this, we scale the data to have zero mean and unit variance based on the corresponding train split. Hence, we obtain $\mathcal{D}_{P1}^{\text{stat}}, \mathcal{D}_{P2}^{\text{stat}} \subset \mathbb{R}^{24}$.

For the baselines that rely on CWT features, we follow the procedure presented in "Downstream task: Bosch CNC-milling dataset" section. Thus, for $\mathcal{D}_{P1}$ and $\mathcal{D}_{P2}$, we compute the CWT using Morlet wavelets with $s = 128$ scales on the raw time series data, resize the resulting scalograms

Table 15 Hyperparameters for shallow baselines

	Hyperparameter	$\mathcal{D}_{P1}^{\text{Downstream}}$	$\mathcal{D}_{P1}^{\text{stat}}$
OC-SVM	kernel	RBF	RBF
	γ	2^{-30}	2^{-30}
	ν	0.01	0.005
IF	n_estimators	300	200
	max_samples	4	128
PCA	n_components	3	6
Kernel-PCA	n_components	9	23
	kernel	cosine	cosine
	Hyperparameter	$\mathcal{D}_{P2}^{\text{Downstream}}$	$\mathcal{D}_{P2}^{\text{stat}}$
OC-SVM	kernel	RBF	RBF
	γ	4	0.5
	ν	0.005	0.005
IF	n_estimators	10	10
	max_samples	256	64
PCA	n_components	10	17
Kernel-PCA	n_components	10	18
	kernel	cosine	cosine

Table 16 Hyperparameters for self-supervised baselines

	Hyperparameter	$\mathcal{D}_{P1}^{\text{Downstream}}$
RotNet		
+PCA	n_components	5
+Kernel-PCA	n_components	9
	kernel	cosine
+OC-SVM	kernel	RBF
	γ	2^{-30}
	ν	0.005
+IF	n_estimators	50
	max_samples	64
	Hyperparameter	$\mathcal{D}_{P2}^{\text{Downstream}}$
RotNet		
+PCA	n_components	64
+Kernel-PCA	n_components	10
	kernel	cosine
+OC-SVM	kernel	RBF
	γ	0.125
	ν	0.005
+IF	n_estimators	50
	max_samples	16

to 128×512 and apply min-max scaling based on the train split. This leads to $\mathcal{D}_{P1}^{\text{Downstream}}, \mathcal{D}_{P2}^{\text{Downstream}} \subset \mathbb{R}^{128\times512\times3}$. Note that each of the preceding datasets contains a total of 776 samples for P1 and 775 samples for P2.

Regarding the pretext datasets for SSMSPC, we use the setting as presented in "Pretext task: Bosch CNC-milling dataset" section, i.e., we augment the data using *Identity*, *CutPaste*, *MeanShift* and *MissingSignal* augmentations with a total of 5 windows. This is followed by CWT using Morlet wavelets with $s = 128$ scales, resizing to 128×512 and subsequent min-max scaling based on the training data. Thus, we have $\mathcal{D}_{P1}^{\text{Pretext}}, \mathcal{D}_{P2}^{\text{Pretext}} \subset \mathbb{R}^{128\times512\times3}$, where $|\mathcal{D}_{P1}^{\text{Pretext}}| = 1860$ and $|\mathcal{D}_{P2}^{\text{Pretext}}| = 1860$. Recall that we use the respective train split of $\mathcal{D}_{P1}$ and $\mathcal{D}_{P2}$ as the basis for the pretext dataset generation.

Baselines: CiP-DMD dataset

We use the baselines as presented in "Baselines: Bosch CNC-milling dataset" section. For the deep and some of the self-supervised baselines, we keep the hyperparameters from the first experiment, as we found them to provide the best performance in terms of the corresponding validation sets. The hyperparameters of the shallow and the RotNet self-supervised baselines are listed in Tables 15 and 16. Regarding the architecture of SSMSPC, we use the settings from Tables 3 and 4, respectively. For the pretext tasks and the corresponding downstream tasks, we follow the procedure presented in "Pretext task" and "Downstream task" sections. In terms of the hyperparameters for SSMSPC, we increase the batch size to 32 and reduce the epochs to 10, leaving the remaining hyperparameters unchanged.

Results: CiP-DMD dataset

Table 17 shows the results for the second experiment. We follow the evaluation scheme presented in "Results: Bosch CNC-milling dataset" section, i.e., we evaluate the performance of each model with respect to AUROC. We included a "Total" column that contains the average performance of the corresponding baselines with respect to P1 and P2. Note that we use the average performance as the final evaluation measure, since it provides the best estimate for the overall monitoring performance.

As we can see from the results, SSMSPC yields the highest AUROC score with respect to the overall performance and beats all other baselines on P1. In contrast to the first experiment, we make the surprising observation that the shallow baselines relying on statistical features demonstrate a very strong performance. Specifically, PCA outperforms state-of-the-art deep and self-supervised anomaly detection models such as, e.g., DeppSVDD, PatchCore and NeuTraL AD. When evaluated solely on P2, PCA even outperforms SSMSPC by a small margin. The shallow models relying on CWT features are worse, showing a drop in performance of at least 14.8% when compared to SSMSPC.

Table 17 Results on the CiP-DMD dataset

Category	Feature	Method	AUROC P1	P2	Total
Shallow	Statistical	OC-SVM	98.3 ± 1.2	98.8 ± 0.7	98.5
		PCA	99.3 ± 0.3	**99.5 ± 0.4**	99.4
		Kernel-PCA	98.7 ± 0.7	98.4 ± 1.2	98.5
		IF	98.7 ± 0.5	86.7 ± 3.9	92.7
	CWT	LeNet + PCA	67.4 ± 20.5	93.3 ± 3.4	80.4
		LeNet + Kernel-PCA	75.7 ± 18.0	92.7 ± 4.5	84.2
		LeNet + OC-SVM	71.8 ± 14.3	97.8 ± 0.8	84.8
		LeNet + IF	68.0 ± 16.0	85.2 ± 5.3	76.6
Deep	Statistical	DAGMM	99.4 ± 0.3	95.3 ± 3.0	97.4
	CWT	DeepSVDD	86.3 ± 17.0	75.5 ± 15.2	80.9
		LeNet-AE	61.8 ± 22.8	73.1 ± 22.9	67.4
		PatchCore	99.4 ± 0.4	99.2 ± 0.4	99.3
Self-supervised	CWT	NeuTraL AD	99.7 ± 0.4	98.2 ± 0.7	99.0
		GeoTrans	97.6 ± 1.6	59.5 ± 18.4	78.6
		RotNet	79.8 ± 4.2	71.2 ± 5.9	75.5
		RotNet + PCA	99.5 ± 0.4	97.7 ± 1.4	98.6
		RotNet + Kernel-PCA	97.9 ± 1.3	93.4 ± 2.9	95.7
		RotNet + OC-SVM	97.9 ± 1.6	97.8 ± 0.8	97.9
		RotNet + IF	98.2 ± 1.1	81.6 ± 7.5	89.9
		SSMSPC (ours)	**99.8 ± 0.25**	99.4 ± 0.3	**99.6**

We report the mean and standard deviation for AUROC over 5 different random seeds for P1 and P2. State-of-the-art baselines are investigated using both, statistical features and CWT features. The best results are given in bold

In comparison to the first experiment, we see a strong deterioration for some of the deep baselines. Specifically, DeepSVDD and LeNet-AE show a weak performance. In fact, LeNet-AE yields the lowest overall score, being 32.3% worse than SSMSPC. Conversely, PatchCore and DAGMM show an overall strong performance.

Regarding the self-supervised baselines, we see that NeuTraL AD shows the best results. The learned representations by RotNet when used as a feature extractor yield a strong performance similar to the first experiment. In Table 18, we display the Precision, Recall and F1-score of SSMSPC for varying values of α to demonstrate its effects on the monitoring performance.

Summarizing, the results of the presented experiments provide some interesting insights. First, we can conclude that the learned representations of SSMSPC are indeed useful for solving tasks that are in alignment with the problem statement described in "Problem statement" section. Second, we can confirm that self-supervised learning is a very promising concept that helps to learn effective representations in situations where the amount of available data is scarce. Third, as we have seen in prior works such as Biegel et al. (2022a), it seems that in certain situations in discrete manufacturing the performance of conventional deep learning methods varies strongly and is often outperformed by basic shallow learning approaches. Fourth, methods like PatchCore that do not encompass an actual training phase but leverage the learned representations from a pretraining phase with data from another domain show very promising performance for future applications in discrete manufacturing.

Table 18 Results for Precision, Recall and F1-score of SSMSPC for varying levels of α on the CiP-DMD dataset

	α	Precision	Recall	F1-score
$\mathcal{D}_{P1}$	0.050	75.0 ± 0.8	98.3 ± 0.2	85.1 ± 0.4
	0.025	87.3 ± 0.3	94.8 ± 2.4	91.0 ± 1.3
	0.010	92.7 ± 0.6	88.0 ± 7.3	90.2 ± 4.1
$\mathcal{D}_{P2}$	0.050	74.5 ± 4.9	100.0 ± 0.0	85.3 ± 3.3
	0.025	89.7 ± 3.2	90.7 ± 7.2	90.2 ± 4.6
	0.010	88.1 ± 4.6	93.1 ± 3.4	90.5 ± 3.8

We report mean and standard deviation over 5 different random seeds

Towards in-process control

As a final demonstration of the practical applicability for real-world discrete manufacturing processes, we deployed SSMSPC at the DMC-50H machine that we used for the creation of the CiP-DMD dataset. Specifically, we incorporated

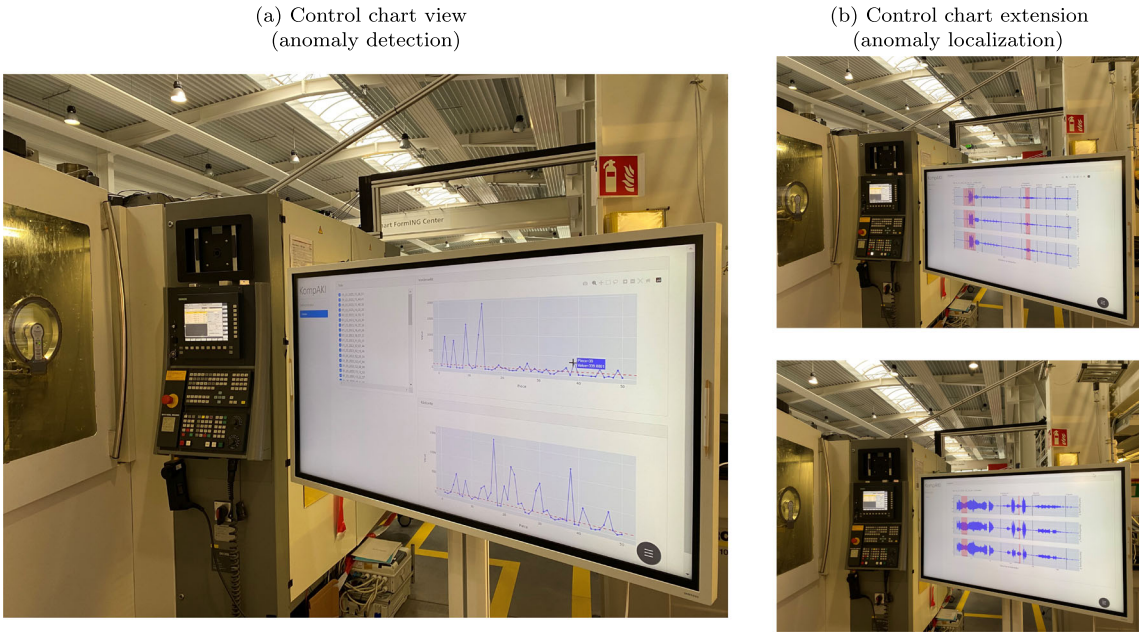


Fig. 17 Web interface of the deployed SSMSPC system at the DMC-50H test bed. **a** Control chart view which is used for anomaly detection. This view shows the control charts for P1 and P2 on a single page and is used by a machine operator to assess the process condition during the process. **b** Extended control chart view for P1 (top) and P2 (bottom). Once an anomalous process condition has been detected, the control chart extension can be used for anomaly localization purposes, to support a machine operator in the root-cause analysis

SSMSPC into a web-based Graphical User Interface (GUI) that allows a machine operator to interact with the system, see Fig. 17. The web interface consists of two separate views. In the first view, two control charts are displayed for P1 and P2, respectively. These control charts are updated automatically, i.e., during the processing cycle of the machine. If the system detects an anomaly, the machine operator can access the control chart extension simply by clicking on the corresponding sample in the control chart. This constitutes the second view of the web interface. The control chart extension presents the process data of the current part. The time series is segmented into the corresponding processing steps using the NC lines that are recorded from the machine control and the anomalous time steps are highlighted. Note that the presented functionality is in alignment with the visualization of the problem statement in Fig. 2.

The deployed system monitors the process data of every part, which is equivalent to a 100% check of the running process. As opposed to a conventional SPC scheme, in which manufactured parts are sampled in equidistant time intervals from the process, the presented system can be used as a trigger for measuring produced parts. Specifically, when the system detects an anomalous process condition, the machine operator can decide, whether it is necessary to measure the corresponding part, based on the extended control chart view.

Conclusion

In this paper we introduce SSMSPC, a novel approach for MSPC based on self-supervised learning to detect and localize anomalous process behavior in discrete manufacturing processes.

We present a pretext task that we refer to as Location + Transformation prediction. Given a randomly augmented time series input, the objective in this pretext task is to train a model to classify both, the type and the location of the augmentation. In the subsequent downstream task, we apply Hotelling's T^2 on top of the learned representations from the pretext task, following the one-class classification setting. We fit the control limits using KDE based on the computed T^2 values in the downstream task.

In addition to the conventional control chart view, we propose a control chart extension that facilitates the identification of the root cause by segmenting a raw time series signal into the individual process steps using the NC lines of the machining program and highlighting the anomalous process components.

We evaluate the performance of SSMSPC using two real-world CNC-milling datasets and compare it to state-of-the-art baselines from the realms of shallow, deep and self-supervised anomaly detection. Our experiments show that SSMSPC learns effective representations achieving the highest score on the given tasks. In addition to that, we

provide qualitative results regarding the proposed control chart extension and show that SSMSPC correctly identifies anomalous process sections. We perform an ablation study to demonstrate how varying different components, such as adding or removing augmentations, affects the performance of SSMSPC in the respective downstream task.

Despite the strong performance of our presented approach, it is necessary to mention that the results presented within this paper are not without limitations. The datasets that we used as a basis for our evaluation stem exclusively from CNC-milling processes, which limits the expressiveness of the conducted experiments in terms of generalizability to other discrete manufacturing processes. However, to the best of our knowledge, there is currently no publicly available dataset existing that would provide a more realistic setting for the problem statement considered in this work. This highlights the urgency to develop more domain-specific datasets, especially with respect to monitoring discrete manufacturing processes for benchmarking purposes, to advance the research in this field.

The presented paper opens up several research directions for future work. First, it would be interesting to investigate SSMSPC in a contrastive learning setting by employing a contrastive loss function that pulls together the augmented versions of a given time series sample and pushes away all other time series samples. Second, instead of highlighting the anomalous process data across all sensors per default, it would be beneficial to highlight only the process data of the individual sensors that actually recorded an anomaly. Lastly, instead of relying on Hotelling's T^2 as the anomaly score, it would be worthwhile to explore other shallow learning approaches, e.g., PCA or OC-SVM in terms of detection performance.

Acknowledgements The research leading to these results has received funding from the German Federal Ministry of Education and Research (BMBF) under grant agreement number 02L19C150 (KompAKI). We would like to express our gratitude to the anonymous reviewers for their thoughtful feedback and suggestions that greatly improved the quality of this paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

References

Abadi, M., Ashish, A., Paul, B., Eugene, B., Zhifeng, C., Craig, C. & Xiaoqiang, Z. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems.

Abati, D., Porrello, A., Calderara, S. & Cucchiara, R. (2019). Latent space autoregression for novelty detection. In *Conference on computer vision and pattern recognition* (pp. 481–490). https://doi.org/10.1109/CVPR.2019.00057

Abdulaal, A., Liu, Z. & Lancewicki, T. (2021). Practical approach to asynchronous multivariate time series anomaly detection and localization. In *KDD* (pp. 2485–2494). https://doi.org/10.1145/3447548.3467174

Aggarwal, C. C. (2017). Outlier analysis. *Springer, New York.* https://doi.org/10.1007/978-3-319-47578-3

Ahmad, S., Enshaei, N., Naderkhani, F. & Awasthi, A. (2020). Integrated deep learning and statistical process control for online monitoring of manufacturing processes. In *International conference on prognostics and health management* (pp. 1–6). https://doi.org/10.1109/ICPHM49022.2020.9187046

Ai, M., Xie, Y., Ding, S. X., Tang, Z., & Gui, W. (2023). Domain knowledge distillation and supervised contrastive learning for industrial process monitoring. *IEEE Transactions on Industrial Electronics, 70*(9), 9452–9462. https://doi.org/10.1109/TIE.2022.3206696

Alauddin, M., Khan, F., Imtiaz, S., & Ahmed, S. (2018). A bibliometric review and analysis of data-driven fault detection and diagnosis methods for process systems. *Industrial & Engineering Chemistry Research, 57*(32), 10719–10735. https://doi.org/10.1021/acs.iecr.8b00936

Bergman, L., & Hoshen, Y. (2020). Classification-based anomaly detection for general data. In *International conference on learning representations*. arXiv:2005.02359

Bergmann, P., Fauser, M., Sattlegger, D., & Steger, C. (2019). MVTEC ad—A comprehensive real-world dataset for unsupervised anomaly detection. *Conference on computer vision and pattern recognition* (pp. 9592–9600). https://doi.org/10.1007/s11263-020-01400-4

Biegel, T., Jourdan, N., Hernandez, C., Cviko, A., & Metternich, J. (2022). Deep learning for multivariate statistical in-process control in discrete manufacturing: A case study in a sheet-metal forming process. *Procedia CIRP, 107*, 422–427. https://doi.org/10.1016/j.procir.2022.05.002

Biegel, T., Jourdan, N., Madreiter, T., Kohl, L., Fahle, S., Ansari, F., & Metternich, J. (2022). Combining process monitoring with text mining for anomaly detection in discrete manufacturing. *SSRN*. https://doi.org/10.2139/ssrn.4073942

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition, 30*, 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

Campos, G. O., Zimek, A., Sander, J., Campello, R. J. G. B., Micenková, B., Schubert, E., & Houle, M. E. (2016). On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery, 30*(4), 891–927. https://doi.org/10.1007/s10618-015-0444-8

Carrara, F., Amato, G., Brombin, L., Falchi, F. & Gennaro, C. (2020). Combining gans and autoencoders for efficient anomaly detection. In *International conference on pattern recognition* (pp. 3939–3946). arXiv:2011.08102

Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint: arXiv:1901.03407.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), 1–58. https://doi.org/10.1145/1541880.1541882

Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607). arXiv:2002.05709

Cheng, F., He, Q. P., & Zhao, J. (2019). A novel process monitoring approach based on variational recurrent autoencoder. *Computers & Chemical Engineering, 129*, 106515. https://doi.org/10.1016/j.compchemeng.2019.106515

Cheng, J., & Vasconcelos, N. (2021). Learning deep classifiers consistent with fine-grained novelty detection. In *Conference on computer vision and pattern recognition* (pp. 1664–1673).

Dai, E., & Chen, J. (2022). Graph-augmented normalizing flows for anomaly detection of multiple time series. In *International conference on learning representations*. arXiv:2202.07857

Dehaene, D., Frigo, O., Combrexelle, S. & Eline, P. (2020). Iterative energy-based projection on a normal data manifold for anomaly localization. In *International conference on learning representations*. arXiv:2002.03734

Ding, X., Li, Y., Belatreche, A., & Maguire, L. P. (2014). An experimental evaluation of novelty detection methods. *Neurocomputing, 135*, 313–327. https://doi.org/10.1016/j.neucom.2013.12.002

Doersch, C., Gupta, A., & Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. *International conference on computer vision* (pp. 1422–1430). https://doi.org/10.1109/ICCV.2015.167

Ermolov, A., Siarohin, A., Sangineto, E. & Sebe, N. (2021). Whitening for self-supervised representation learning. In *International conference on machine learning* (pp. 3015–3024). arXiv:2007.06346

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters, 27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Ferrer, A. (2007). Multivariate statistical process control based on principal component analysis (MSPC-PCA): Some reflections and a case study in an autobody assembly process. *Quality Engineering, 19*(4), 311–325. https://doi.org/10.1080/08982110701621304

Ferrer, A. (2014). Latent structures-based multivariate statistical process control: A paradigm shift. *Quality Engineering, 26*(1), 72–91. https://doi.org/10.1080/08982112.2013.846093

Fu, Y., & Xue, F. (2022). Mad: Self-supervised masked anomaly detection task for multivariate time series. *International Joint Conference on Neural Networks*. https://doi.org/10.1109/IJCNN55064.2022.9892218

Gao, R. X., & Yan, R. (2011). *Continuous wavelet transform*. In Wavelets: Theory and applications for manufacturing. Springer. https://doi.org/10.1007/978-1-4419-1545-0_3

Ge, Z., Song, Z., & Gao, F. (2013). Review of recent research on data-based process monitoring. *Industrial & Engineering Chemistry Research, 52*(10), 3543–3562. https://doi.org/10.1021/ie302069q

Gidaris, S., Singh, P. & Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *International conference on learning representations*. arXiv:1803.07728

Golan, I., & El-Yaniv, R. (2018). Deep anomaly detection using geometric transformations. *Conference on neural information processing systems*. arXiv:1805.10917

Goyal, S., Raghunathan, A., Jain, M., Simhadri, H.V. & Jain, P. (2020). Drocc: Deep robust one-class classification. In *International conference on machine learning*. arXiv:2002.12718

Grasso, M., Colosimo, B. M., Semeraro, Q., & Pacella, M. (2015). A comparison study of distribution-free multivariate SPC methods for multimode data. *Quality and Reliability Engineering International, 31*(1), 75–96. https://doi.org/10.1002/qre.1708

Grossmann, A., & Morlet, J. (1984). Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM Journal on Mathematical Analysis, 15*(4), 723–736. https://doi.org/10.1137/0515056

Gu, X., Akoglu, L. & Rinaldo, A. (2019). Statistical analysis of nearest neighbor methods for anomaly detection. In *Conference on neural information processing systems*. arXiv:1907.03813

Gupta, M., Gao, J., Aggarwal, C. C., & Han, J. (2014). Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*. https://doi.org/10.1109/TKDE.2013.184

Hahn, T., & Mechefske, C. K. (2021). Self-supervised learning for tool wear monitoring with a disentangled-variational-autoencoder. *International Journal of Hydromechatronics*. https://doi.org/10.1504/IJHM.2021.10035377

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep residual learning for image recognition. In *Conference on computer vision and pattern recognition* (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

Hendrycks, D., Mazeika, M. & Dietterich, T. (2019). Deep anomaly detection with outlier exposure. In *International conference on learning representations*. arXiv:1812.04606

Hendrycks, D., Mazeika, M., Kadavath, S. & Song, D. (2019). Using self-supervised learning can improve model robustness and uncertainty. In *Conference on neural information processing systems*. arXiv:1906.12340

Hotelling, H. (1947). Multivariate quality control, illustrated by the air testing of sample bombsights. Techniques of statistical analysis (pp. 111–184).

Hu, W., Wang, M., Qin, Q., Ma, J. & Liu, B. (2020). Hrn: A holistic approach to one class learning. In *Conference on neural information processing systems* (pp. 19111–19124).

Hübner, H. B., Duarte, M. A. V., & Da Silva, R. B. (2020). Automatic grinding burn recognition based on time-frequency analysis and convolutional neural networks. *The International Journal of Advanced Manufacturing Technology, 110*(7–8), 1833–1849. https://doi.org/10.1007/s00170-020-05902-w

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning,37,448-456. arXiv:1502.03167

Jackson, J. E. (1991). A user's guide to principal components. *New YorkJohn Wiley*. https://doi.org/10.1002/0471725331

Jing, L., & Tian, Y. (2020). Self-supervised visual feature learning with deep neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence,4037-4058. arXiv:1902.06162

Kingma, D.P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations*. arXiv:1412.6980

Kong, D., & Yan, X. (2020). Industrial process deep feature representation by regularization strategy autoencoders for process monitoring. *Measurement Science and Technology, 31*(2), 025104. https://doi.org/10.1088/1361-6501/ab48c7

Kourti, T., & MacGregor, J. F. (1996). Multivariate SPC methods for process and product monitoring. *Journal of Quality Technology, 28*(4), 409–428. https://doi.org/10.1080/00224065.1996.11979699

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Conference on neural information processing systems* (Vol. 25, pp. 1097–1105).

Kumagai, A., Iwata, T. & Fujiwara, Y. (2019). Transfer anomaly detection by inferring latent domain representations. In *Conference on neural information processing systems*.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324. https://doi.org/10.1109/5.726791

Li, C.-L., Sohn, K., Yoon, J. & Pfister, T. (2021). Cutpaste: Self-supervised learning for anomaly detection and localization. In

Conference on computer vision and pattern recognition (pp. 9664–9674).

Li, D., Lu, J., Zhang, T., & Ding, J. (2023). Self-supervised learning and multisource heterogeneous information fusion based quality anomaly detection for heavy-plate shape. *In IEEE transactions on automation science and engineering*. https://doi.org/10.1109/TASE.2023.3265649

Li, S., Luo, J., & Hu, Y. (2022). Toward interpretable process monitoring: Slow feature analysis-aided autoencoder for spatiotemporal process feature learning. *IEEE Transactions on Instrumentation and Measurement, 71*, 1–11. https://doi.org/10.1109/TIM.2021.3127284

Li, W., Zhang, C., Tsung, F., & Mei, Y. (2020). Nonparametric monitoring of multivariate data via KNN learning. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2020.1812750

Li, Z., Zhao, Y., Han, J., Su, Y., Jiao, R., Wen, X. & Pei, D. (2021). Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding. In *KDD* (pp. 3220–3230). https://doi.org/10.1145/3447548.3467075

Liao, Y., Ragai, I., Huang, Z., & Kerner, S. (2021). Manufacturing process monitoring using time-frequency representation and transfer learning of deep neural networks. *Journal of Manufacturing Processes, 68*, 231–248. https://doi.org/10.1016/j.jmapro.2021.05.046

Lindemann, B., Fesenmayr, F., Jazdi, N., & Weyrich, M. (2019). Anomaly detection in discrete manufacturing using self-learning approaches. *Procedia CIRP, 79*, 313–318. https://doi.org/10.1016/j.procir.2019.02.073

Lindemann, B., Jazdi, N., & Weyrich, M. (2020). Anomaly detection and prediction in discrete manufacturing based on cooperative LSTM networks. In *IEEE international conference on automation science and engineering* (pp. 1003–1010). https://doi.org/10.1109/CASE48305.2020.9216855

Liu, C., Wang, K., Wang, Y., & Yuan, X. (2022). Learning deep multimanifold structure feature representation for quality prediction with an industrial application. *IEEE Transactions on Industrial Informatics, 18*(9), 5849–5858. https://doi.org/10.1109/TII.2021.3130411

Liu, F.T., Ting, K.M. & Zhou, Z,-H. (2008). Isolation forest. In *IEEE international conference on data mining* (pp. 413–422). https://doi.org/10.1109/ICDM.2008.17

Liznerski, P., Ruff, L., Vandermeulen, R.A., Franks, B.J., Kloft, M. & Müller, K.-R. (2021). Explainable deep one-class classification. In *International conference on learning representations*. arXiv:2007.01760

Lorenti, L., de Rossi, G., Annoni, A., Rigutto, S., & Susto, G. A. (2022). Cuad-mo: Continuos unsupervised anomaly detection on machining operations. *In IEEE conference on control technology and applications*. https://doi.org/10.1109/CCTA49430.2022.9966138

Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *International conference on learning representations*. https://doi.org/10.48550/ARXIV.1608.03983

Lu, S., Dong, H., & Yu, H. (2023). Abnormal condition detection method of industrial processes based on cascaded bagging-PCA and CNN classification network. *In IEEE transactions on industrial informatics*. https://doi.org/10.1109/TII.2023.3242811

MacGregor, J. F. (1997). Using on-line process data to improve quality: Challenges for statisticians. *International Statistical Review, 65*(3), 309–323. https://doi.org/10.1111/j.1751-5823.1997.tb00311.x

Montgomery, D.C. (2009). Introduction to Statistical Quality Control Introduction to statistical quality control (6th ed.). Wiley.

Noroozi, M., & Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. *European conference on computer vision* (pp. 69–84). arXiv:1603.09246

Noroozi, M., Vinjimoor, A., Favaro, P. & Pirsiavash, H. (2018). Boosting self-supervised learning via knowledge transfer. In *Conference on computer vision and pattern recognition* (pp. 9359–9367). arXiv:1805.00385

Oshida, T., Murakoshi, T., Zhou, L., Ojima, H., Kaneko, K., Onuki, T., & Shimizu, J. (2023). Development and implementation of real-time anomaly detection on tool wear based on stacked LSTM encoder-decoder model. *International Journal of Advanced Manufacturing Technology*. https://doi.org/10.1007/s00170-023-11497-9

Pang, G., Shen, C., Cao, L., & van den Hengel, A. (2021). Deep learning for anomaly detection: A review. *ACM Computing Surveys, 54*(2), 1–38. https://doi.org/10.1145/3439950

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. arXiv:1201.0490.

Proteau, A., Zemouri, R., Tahan, A., & Thomas, M. (2020). Dimension reduction and 2d-visualization for early change of state detection in a machining process with a variational autoencoder approach. *The International Journal of Advanced Manufacturing Technology, 111*(11–12), 3597–3611. https://doi.org/10.1007/s00170-020-06338-y

Qin, S. J. (2003). Statistical process monitoring: Basics and beyond. *Journal of Chemometrics, 17*(8–9), 480–502. https://doi.org/10.1002/cem.800

Qin, S. J. (2012). Survey on data-driven industrial process monitoring and diagnosis. *Annual Reviews in Control, 36*(2), 220–234. https://doi.org/10.1016/j.arcontrol.2012.09.004

Qin, S. J., & Chiang, L. H. (2019). Advances and opportunities in machine learning for process data analytics. *Computers & Chemical Engineering, 126*, 465–473. https://doi.org/10.1016/j.compchemeng.2019.04.003

Qiu, C., Pfrommer, T., Kloft, M., Mandt, S. & Rudolph, M. (2021). Neural transformation learning for deep anomaly detection beyond images. In *International conference on machine learning* (pp. 8703–8714). arXiv:2103.16440

Qiu, P., & Xie, X. (2021). Transparent sequential learning for statistical process control of serially correlated data. *Technometrics*. https://doi.org/10.1080/00401706.2021.1929493

Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T. & Gehler, P. (2022). Towards total recall in industrial anomaly detection. In *Conference on computer vision and pattern recognition*. arXiv:2106.08265

Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., & Müller, K.-R. (2021). A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE, 109*(5), 756–795. https://doi.org/10.1109/JPROC.2021.3052449

Ruff, L., Vandermeulen, R.A., Görnitz, N., Binder, A., Müller, E., Müller, K.-R. & Kloft, M. (2020). Deep semi-supervised anomaly detection. In *International conference on learning representations*. arXiv:1906.02694

Ruff, L., Vandermeulen, R.A., Görnitz, N., Deecke, L., Siddiqui, S.A., Binder, A. & Kloft, M. (2018). Deep one-class classification. In *International conference on machine learning* (pp. 4398–4402).

Schölkopf, B., Williamson, R.C., Smola, A.J., Shawe-Taylor, J. & Platt, J.C. (1999). Support vector method for novelty detection. In *Conference on neural information processing systems* (pp. 582–588).

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2020). Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision, 128*(2), 336–359. https://doi.org/10.1007/s11263-019-01228-7

Shen, L., Li, Z., & Kwok, J. T. (2020). Timeseries anomaly detection using temporal hierarchichal one-class network. *Conference on Neural Information Processing Systems, 33*, 13016–13026.

Shen, L., Yu, Z., Ma, Q., & Kwok, J. T. (2021). Time series anomaly detection with multiresolution ensemble decoding. *AAAI Conference on Artificial Intelligence, 35*(11), 9567–9575. https://doi.org/10.1609/aaai.v35i11.17152

Shenkar, T., & Wolf, L. (2022). Anomaly detection for tabular data with internal contrastive learning. In *International conference on learning representations*.

Sohn, K., Li, C-.L., Yoon, J., Jin, M. & Pfister, T. (2021). Learning and evaluating representations for deep one-class classification. In *International conference on learning representations*.

Sun, S., Liu, Y., Hu, X., & Zhang, W. (2023). A semisupervised autoencoder-based method for anomaly detection in cutting tools. *Journal of Manufacturing Processes, 93*, 315–327. https://doi.org/10.1016/j.jmapro.2023.03.043

Tack, J., Mo, S., Jeong, J. & Shin, J. (2020). Csi: Novelty detection via contrastive learning on distributionally shifted instances. In *Conference on neural information processing systems* (Vol. 33, pp. 11839–11852). arXiv:2007.08176

Tang, P., Peng, K., Dong, J., Zhang, K., & Zhao, S. (2020). Monitoring of nonlinear processes with multiple operating modes through a novel Gaussian mixture variational autoencoder model. *IEEE Access, 8*, 114487–114500. https://doi.org/10.1109/ACCESS.2020.3003095

Tax, D. M., & Duin, R. P. (2004). Support vector data description. *Machine Learning, 54*(1), 45–66. https://doi.org/10.1023/B:MACH.0000008084.60811.49

Tnani, M.-A., Feil, M., & Diepold, K. (2022). Smart data collection system for brownfield CNC milling machines: A new benchmark dataset for data-driven machine monitoring. *Procedia CIRP, 107*, 131–136. https://doi.org/10.1016/j.procir.2022.04.022

Tran, M.-Q., Liu, M.-K., & Tran, Q.-V. (2020). Milling chatter detection using scalogram and deep convolutional neural network. *The International Journal of Advanced Manufacturing Technology, 107*(3–4), 1505–1516. https://doi.org/10.1007/s00170-019-04807-7

Tran, T., & Lundgren, J. (2020). Drill fault diagnosis based on the scalogram and mel spectrogram of sound signals using artificial intelligence. *IEEE Access*. https://doi.org/10.1109/ACCESS.2020.3036769

Wang, R., Liu, C., Mou, X., Gao, K., Guo, X., Liu, P. & Liu, X. (2023). Deep contrastive one-class time series anomaly detection. In *SIAM international conference on data mining* (pp. 694–702). arXiv:2207.01472

Wang, Y., Si, Y., Huang, B., & Lou, Z. (2018). Survey on the theoretical research and engineering applications of multivariate statistics process monitoring algorithms: 2008–2017. *Canadian Journal of Chemical Engineering, 96*(10), 2073–2085. https://doi.org/10.1002/cjce.23249

Woodall, W. H. (2000). Controversies and contradictions in statistical process control. *Journal of Quality Technology, 32*(4), 341–350. https://doi.org/10.1080/00224065.2000.11980013

Woodall, W. H. (2017). Bridging the gap between theory and practice in basic statistical process monitoring. *Quality Engineering, 29*(1), 2–15. https://doi.org/10.1080/08982112.2016.1210449

Woodall, W. H., & Montgomery, D. C. (1999). Research issues and ideas in statistical process control. *Journal of Quality Technology, 31*(4), 376–386. https://doi.org/10.1080/00224065.1999.11979944

Woodall, W. H., Spitzner, D. J., Montgomery, D. C., & Gupta, S. (2004). Using control charts to monitor process and product quality profiles. *Journal of Quality Technology, 36*(3), 309–320. https://doi.org/10.1080/00224065.2004.11980276

Wu, J-.C., Chen, D-.J., Fuh, C-.S. & Liu, T-.L. (2021). Learning unsupervised metaformer for anomaly detection. In *International conference on computer vision* (pp. 4369–4378).

Wu, Z., Li, Y., Tsung, F., & Pan, E. (2021). Real-time monitoring and diagnosis scheme for IOT-enabled devices using multivariate SPC techniques. *IISE Transactions, 55*(4), 348–362. https://doi.org/10.1080/24725854.2021.2000681

Xie, X., & Peihua, Q. (2022). Machine learning control charts for monitoring serially correlated data. In *Control charts and machine learning for anomaly detection in manufacturing*. Springer.

Ye, Z., Chen, Y. & Zheng, H. (2021). Understanding the effect of bias in deep anomaly detection. *International joint conference on artificial intelligence* (Vol. 3, pp. 3314–3320). https://doi.org/10.24963/ijcai.2021/456

Yin, S., Ding, S. X., Xie, X., & Luo, H. (2014). A review on basic data-driven approaches for industrial process monitoring. *IEEE Transactions on Industrial Electronics, 61*(11), 6418–6428. https://doi.org/10.1109/TIE.2014.2301773

Yu, J., Liu, X., & Ye, L. (2021). Convolutional long short-term memory autoencoder-based feature learning for fault detection in industrial processes. *IEEE Transactions on Instrumentation and Measurement, 70*, 1–15. https://doi.org/10.1109/TIM.2020.3039614

Yu, J., & Zhang, C. (2020). Manifold regularized stacked autoencoders-based feature learning for fault detection in industrial processes. *Journal of Process Control, 92*, 119–136. https://doi.org/10.1016/j.jprocont.2020.06.001

Yu, J., & Zhang, Y. (2023). Challenges and opportunities of deep learning-based process fault detection and diagnosis: A review. *Neural Computing and Applications, 35*(1), 211–252. https://doi.org/10.1007/s00521-022-08017-3

Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W. & Chawla, N.V. (2019). A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *AAAI conference on artifical intelligence* (pp. 1409–1416).

Zhang, C., Tsung, F., & Zou, C. (2015). A general framework for monitoring complex processes with both in-control and out-of-control information. *Computers & Industrial Engineering, 85*, 157–168. https://doi.org/10.1016/j.cie.2015.03.007

Zhang, C., Yu, J., & Wang, S. (2021). Fault detection and recognition of multivariate process based on feature learning of one-dimensional convolutional neural network and stacked denoised autoencoder. *International Journal of Production Research, 59*(8), 2426–2449. https://doi.org/10.1080/00207543.2020.1733701

Zong, B., Song, Q., Renqiang Min, M., Cheng, W., Lumezanu, C., Cho, D. & Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*.