A self-supervised learning approach for multivariate statistical in-process control in discrete manufacturing processes

Am Fachbereich Maschinenbau
an der Technischen Universität Darmstadt
zur
Erlangung des Grades eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

DISSERTATION

vorgelegt von

Tobias Biegel, M. Sc.

aus Lebach

Berichterstatter:	Prof. Dr.-Ing. Joachim Metternich
Mitberichterstatter:	Prof. Dr.-Ing. Uwe Klingauf
Tag der Einreichung:	21.10.2023
Tag der mündlichen Prüfung:	13.12.2023

Darmstadt 2023
D17

Preface of the Author

The underlying thesis was written during my time as a research associate at the Institute of Production Management, Technology and Machine Tools (PTW) in the research group Center for industrial Productivity (CiP) at the Technical University of Darmstadt. Over the past few years, I have met many individuals who offered me support and guidance in my journey to complete this work. I want to use the opportunity and express my gratitude to these individuals in the following paragraphs.

First, I would like to thank Prof. Dr.-Ing. Joachim Metternich for the supervision of this thesis, the trust and the constructive cooperation during my time at PTW. Being a research associate at PTW has had a profound impact on my personal and professional development. The opportunity to work on a variety of different tasks and topics, such as developing new graduate courses, writing research proposals, conducting research and consulting projects, was challenging, but at the same time extremely rewarding and instructive. Maneuvering through all of these tasks in addition to writing the PhD thesis has been an excellent preparation for the challenges that lie ahead. Next, I want to thank Prof. Dr.-Ing. Uwe Klingauf for his genuine interest in my work and for accepting to be my co-referee. The opportunity to present my thesis topic in front of your research group was a great experience.

There are many people at PTW in general and in the CiP research group in particular to whom I owe my gratitude for their continued motivation and support. Special thanks go to the colleagues in my focus area "Data Science and Artificial Intelligence in Manufacturing". Here, I want to mention Nicolas Jourdan, Florian Mitschke, and Jan Hämmelmann. Thank you guys for proofreading this thesis, for the professional collaboration, and for the fruitful discussions we had even in the most challenging times. I wish you all the best for your own PhD. Next, I would like to thank our technician Christoph Schwarz, and the research assistants Frederic Fries, Linus Schütte, Louis Jöbstl, Frederik Zahradnik, An Ngo, and Hannes Mittwollen. Thank you for your support and the many hours we spent together at the DMC 50H machine during the development of the CiP-DMD. I would also like to thank Patrick Helm, my former KompAKI research assistant, for his outstanding proofreading skills and contributions to this thesis. Your help has certainly improved the quality of my work.

Aside from PTW, there are other individuals who deserve to be mentioned in this context. I would like to thank you (Sir) Benedikt Kelm for proofreading this thesis and for your continuous support and friendship since we embarked on our journey together in 2014. Furthermore, I want to thank my brother Amin Ranem for the time he invested in proofreading this thesis and for his feedback. Special thanks go to my parents, Marion Ranem and Herbert Biegel, for providing the necessary foundation to complete this journey.

The last paragraph of this preface is dedicated to my wife Nadiia and our two dogs Byasha and Esha. Dear Nadiia, thank you for your love, your never-ending support, and for keeping me motivated these past years. You have always put your own interests aside and I have to admit that you have sacrificed a lot to help me focus on writing this thesis. It is no exaggeration to say that it would not have been possible to complete this thesis in both time and quality without your support. The same goes for Byasha and Esha. Thank you for your love and for protecting us from all kinds of danger, such as birds, flies, cats, and squirrels.

Abstract

Die Überwachung diskreter Fertigungsprozesse zur zuverlässigen Detektion von Anomalien ist von zentraler Bedeutung, um ungeplante Ausfallzeiten zu reduzieren und letztendlich einen Wettbewerbsvorteil zu erzielen. Multivariate statistische In-Prozess-Kontrolle (MSPC) ist eines der Hauptforschungsgebiete, das sich mit der Erkennung von Anomalien in Prozessdaten befasst und als solches stark von den Entwicklungen in der allgemeinen Anomaly Detection (AD) Forschung beeinflusst wird. Jüngste Erkenntnisse aus verschiedenen Bereichen zeigen, dass der Einsatz von Self-Supervised Learning (SSL) in AD-Methoden zu einem ungeahnten Leistungsniveau führt, was als wichtiger Meilenstein in der AD-Literatur angesehen wird. Obwohl diese Ergebnisse darauf hindeuten, dass der der Einsatz von SSL für die MSPC in der diskreten Fertigung von Vorteil sein könnte, gibt es derzeit keine Arbeiten, die SSL in die MSPC für diskrete Fertigungsprozesse einbeziehen.

Die vorliegende Dissertation hat das Ziel, diese Lücke zu schließen und umfasst die Entwicklung eines SSL-Ansatzes für MSPC in diskreten Fertigungsprozessen. Dieser Ansatz wird als Self-Supervised Multivariate Statistical In-Process Control (SSMSPC) bezeichnet. SSMSPC besteht aus drei Komponenten: (1) Location + Transformation Prediction Pretext Task, (2) AD Downstream Task, und (3) Regelkartenerweiterung. Das Ziel der Location + Transformation Prediction Pretext Task ist die Klassifizierung der Augmentation $\mathcal{T}_i$ und des Fensters $\mathcal{W}_j$ basierend auf einer Zeitreihenprobe, die durch eine von k Augmentationsfunktionen in einem von p Fenstern augmentiert wird. In der AD Downstream Task werden die gelernten Repräsentationen aus der Pretext Task verwendet, um den Anomalie-Score zu berechnen, der der Hotelling's T^2 Statistik entspricht. Die Kontrollgrenzen der Regelkarte werden mit Hilfe der Kerndichteschätzung bestimmt. Die Regelkartenerweiterung bietet eine zusätzliche Ansicht zur Regelkarte, in der die anomalen Bereiche in den Prozessdaten hervorgehoben werden.

Der entwickelte Ansatz wird aus Performance und Usability Perspektive validiert. In Bezug auf die Performance wird SSMSPC mit dem Bosch CNC-Fräsdatensatz und dem Center for Industrial Productivity Discrete Manufacturing Dataset (CiP-DMD) mit State-of-the-Art shallow, deep und self-supervised AD-Methoden verglichen. Beide Datensätze repräsentieren reale CNC-Fräsprozesse, bei denen hochfrequente Prozessdaten gesammelt werden. SSMSPC übertrifft in beiden Datensätzen die führenden State-of-the-Art-Baselines und erreicht den höchsten Score in Bezug auf die Fläche unter der Receiver Operating Characteristic Kurve. Die Usability von SSMSPC wird im Rahmen einer Usability-Studie mit Maschinenbedienern bewertet. In dieser Studie wird der CNC-Fräsprozess des CiP-DMD mit einer deployten Instanz von SSMSPC und der univariaten Post-Prozess statistischen Prozesskontrolle (SPC) verglichen, die den Industriestandard in der diskreten Fertigung repräsentiert. Die Usability wird mit Hilfe von Überwachungsprotokollen als objektives Bewertungsmaß und der System Usability Scale als subjektives Bewertungsmaß evaluiert. Es zeigt sich, dass die hervorragenden Detektions- und Lokalisierungsmöglichkeiten von SSMSPC noch nicht ausreichen, um die Ursachenanalyse effektiv zu unterstützen. Stattdessen legen die Ergebnisse der Usability-Studie nahe, die Regelkartenerweiterung von SSMSPC mit den Erkenntnissen aus einem Qualitätskontrollschritt des gefertigten Teils zu kombinieren, um die Usability der univariaten Post-Prozess-SPC sowohl aus objektiver als auch aus subjektiver Sicht zu übertreffen.

Abstract

Monitoring discrete manufacturing processes to reliably detect anomalies, is of fundamental relevance to reduce unplanned downtimes and ultimately gain a competitive advantage. Multivariate Statistical In-Process Control (MSPC) represents one of the main areas of research that center around the detection of anomalies in process data and, as such, is heavily influenced by the developments of general Anomaly Detection (AD) research. Recent findings from various domains demonstrate that the incorporation of Self-Supervised Learning (SSL) into AD methods leads to unprecedented performance levels and is regarded as a major milestone in the AD literature. However, while these findings suggest that the incorporation of SSL might be beneficial for MSPC in discrete manufacturing, there is currently no research that incorporates SSL into MSPC for discrete manufacturing processes.

The objective of this thesis is to close this gap and encompasses the development of an SSL approach for MSPC in discrete manufacturing processes. This approach is referred to as Self-Supervised Multivariate Statistical In-Process Control (SSMSPC). SSMSPC consists of three components: (1) Location + Transformation prediction pretext task, (2) AD downstream task, and (3) control chart extension. The objective of the Location + Transformation prediction pretext task is to classify the augmentation $\mathcal{T}_i$ and the window $\mathcal{W}_j$ based on a time series sample that is artificially augmented by one of k augmentation functions in one of p windows. In the AD downstream task, the learned representations from the pretext task are used to compute the anomaly score, which corresponds to the Hotelling's T^2 statistic. The control limits of the control chart are determined with the help of a Kernel Density Estimation-based threshold selection scheme. The control chart extension provides an additional view to the conventional control chart, in which the anomalous regions in the process data are highlighted.

The developed approach is validated both from a performance and a usability perspective. In terms of performance, SSMSPC is benchmarked against state-of-the-art shallow, deep, and self-supervised AD methods using the Bosch CNC milling dataset and the Center for Industrial Productivity Discrete Manufacturing Dataset (CiP-DMD). Both datasets represent real-world CNC milling processes in which high-frequency process data are collected. SSMSPC is shown to outperform leading state-of-the-art baselines on both datasets, achieving the highest overall score in terms of area under the receiver operating characteristic curve. The usability of SSMSPC is evaluated in the context of a usability study with human machine operators. In this study, the CNC milling process of the CiP-DMD is monitored with a deployed instance of SSMSPC and univariate post-process Statistical Process Control (SPC), which represents the accepted industry standard in discrete manufacturing. The usability is evaluated with the help of monitoring protocols as an objective evaluation measure and the System Usability Scale as a subjective evaluation measure. It is shown that the outstanding detection and localization capabilities provided by SSMSPC are not yet sufficient to effectively support the root-cause analysis. Instead, the findings of the usability study suggest to combine the control chart extension of SSMSPC with the insights obtained from a quality control step of the manufactured part to exceed the usability of univarate post-process SPC, both from an objective and subjective point of view.

Contents

List of Figures

List of Tables

List of Symbols

Sets and intervals

$\{1, \cdots, n\}$	A set of n elements
$[a; b] = \{x \mid a \leq x \leq b\}$	A closed interval
$(a; b) = \{x \mid a < x < b\}$	An open interval
$(a; b] = \{x \mid a < x \leq b\}$	A half-open interval (left open and right closed)
$[a; b) = \{x \mid a \leq x < b\}$	A half-open interval (left closed and right open)
$\mathbb{S}$	An arbitrary set
$\|\mathbb{S}\|$	Cardinality of set $\mathbb{S}$
$\mathbb{R}$	The set of real valued numbers
$\mathbb{R}^n$	The set of real valued n-dimensional vectors
$\mathbb{R}^{n \times m}$	The set of real valued matrices with n rows and m columns
$\mathbb{R}^{d_1 \times d_2 \times \cdots \times d_r}$	The set of real valued tensors with r dimensions
$\mathbb{X}_{\text{train}}$	The set of training data
$\mathbb{X}_{\text{val}}$	The set of validation data
$\mathbb{X}_{\text{test}}$	The set of test data
$\mathbb{Y}_{\text{train}}$	The set of training labels
$\mathbb{Y}_{\text{val}}$	The set of validation labels
$\mathbb{Y}_{\text{test}}$	The set of test labels

Scalars, vectors, matrices and tensors

$\mathbf{A}$	Tensors are denoted by bold uppercase letters
A	Matrices are denoted by italic uppercase letters
$\mathbf{a}$	Vectors are denoted by bold lowercase letters
$\mathbf{a} = [a_1, \cdots, a_n]^T$	A column vector with n elements
$\mathbf{a}^T = [a_1, \cdots, a_n]$	A row vector with n elements
$\mathbf{a}_{i,:}$	The i-th row vector of Matrix A
$\mathbf{a}_{:,j}$	The j-th column vector of Matrix A
a	Scalars are denoted by lowercase letters
a_i	The element of vector $\mathbf{a}$ at index i
$a_{i,j}$	The element of matrix A in row i and column j
$\mathbf{X}$	A data tensor
X	A data matrix
$\mathbf{x}$	A data vector
$\mathbf{y}$	A label vector
$\hat{\mathbf{y}}$	The predicted label vector
y	A label scalar
$\hat{y}$	The predicted label scalar

W	A weight matrix
$\mathbf{w}$	A weight vector
w	A weight scalar
$\hat{\mu}$	The estimated mean vector
$\hat{\Sigma}$	The estimated covariance matrix
$\mathbf{0}$	A vector containing only 0's
$\nabla_{\mathbf{w}} f$	The gradient of a function f with respect to $\mathbf{w}$

Functions and mappings

$f : \mathbb{A} \rightarrow \mathbb{B}$	A mapping f between two sets $\mathbb{A}$ and $\mathbb{B}$
$f(\mathbb{A}) = \{f(a) \| a \in \mathbb{A}\}$	The image of a set $\mathbb{A}$ under f
$f(\mathbf{x}; W)$	A function f of $\mathbf{x}$ that is parameterized by W
$f \circ g$	The composition of two functions f and g
$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$	A loss function taking as input a label $\mathbf{y}$ and a corresponding prediction $\hat{\mathbf{y}}$
$\dim(\cdot)$	The dimensionality function that yields the dimensionality of the input
$\|\| \cdot \|\|_2$	The euclidean norm
$\lfloor \cdot \rfloor$	The floor function
$\log(\cdot)$	The natural logarithm
$\mathbb{1}(\cdot)$	The indicator function
$\det(\cdot)$	The determinant

Distributions

$\mathcal{U}(a, b)$	Discrete uniform distribution with lower bound a and upper bound b

List of Abbreviations

AD	Anomaly Detection
AdaGrad	Adaptive Gradient
ADAM	Adaptive Moment Estimation
AE	Autoencoder
AI	Artificial Intelligence
AUROC	Area Under Receiver Operating Characteristic
BGD	Batch Gradient Descent
BN	Batch Normalization
CAM	Class Activation Mapping
CBPCA	Cascaded Bagging-PCA
CDF	Cumulative Distribution Function
CE	Cross-Entropy
CiP-DMD	Center for Industrial Productivity Discrete Manufacturing Dataset
CLSTM-AE	Convolutional Long Short-Term Memory Autoencoder
CNC	Computerized Numerical Control
CNN	Convolutional Neural Network
COCA	Contrastive One-Class Anomaly Detection
CPS	Cyber-Physical System
CSI	Contrasting Shifted Instances
CSTR	Continuous Stirred Tank Reactor
CUSUM	Cumulated Sum
CWT	Continuous Wavelet Transformation
DAGMM	Deep Autoencoding Gaussian Mixture Model
DL	Deep Learning
Deep-SVDD	Deep Support Vector Data Description
EWMA	Exponentially Weighted Moving Average
FBFP	Fed-Batch Fermentation Penicillin Process
FCN	Fully Connected Network
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Network
GDP	Gross Domestic Product
GeoTrans	Geometric Transformations
GMVAE	Gaussian Mixture Variational Autoencoder
Grad-CAM	Gradient-weighted Class Activation Mapping
GUI	Graphical User Interface
HAI	Hardware In-The-Loop Augmented Industrial Control System
IF	Isolation Forest

IPSAE	Inner Product-Based Stacked Autoencoder
ISO	International Organization for Standardization
KDE	Kernel Density Estimation
KNN	k-Nearest Neighbor
KPCA	Kernel-PCA
LSTM-AE	Long Short-Term Memory Autoencoder
LCL	Lower Control Limit
MAD	Masked Anomaly Detection
MCUSUM	Multivariate CUSUM
MEWMA	Multivariate EWMA
MGD	Mini-Batch Gradient Descent
ML	Machine Learning
MLP	Multi-Layer Perceptron
MNIST	Modified National Institute of Standards and Technology
MRSAE	Manifold Regularized Stacked Autoencoder
MSE	Mean Squared Error
MSPC	Multivariate Statistical In-Process Control
NC	Numerical Control
NeuTraL AD	Neural Transformation Learning for Anomaly Detection
NN	Neural Network
NT-Xent	Normalized Temperature-Scaled Cross Entropy Loss
OC-SVM	One-Class Support Vector Machine
PDF	Probability Density Function
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
ResNet	Residual Network
RGB	Red-Green-Blue
RL	Reinforcement Learning
RMSProp	Root Mean Square Propagation
ROC	Receiver Operating Characteristic
RotNet	Rotation Network
SFA-AE	Slow Feature Analysis-Aided Autoencoder
SGD	Stochastic Gradient Descent
SimCLR	Simple Framework for Contrastive Learning of Visual Representations
SL	Supervised Learning
SPC	Statistical Process Control
SPE	Squared Prediction Error
SSL	Self-Supervised Learning
SSMSPC	Self-Supervised Multivariate Statistical In-Process Control
STFT	Short-Time Fourier Transformation
SUS	System Usability Scale
SVDD	Support Vector Data Description
SVM	Support Vector Machine
TEP	Tennessee Eastman Process
TFT-LCD	Thin-Film Transistor Liquid-Crystal Display
THOC	Temporal Hierarchical One-Class Network

TN	True Negative
TP	True Positive
UCL	Upper Control Limit
UL	Unsupervised Learning
VAE	Variational Autoencoder

1. Introduction

Artificial Intelligence (AI) represents one of the most influential technological trends of the 21st century [Chui23a]. Over the past half decade, the global adoption of AI in businesses has increased by more than 150% [Chui22]. In 2021, the global corporate investments in AI amounted to $\$176.47$ billion which corresponds to an increase of $> 3000\%$ compared to the levels of 2013 [Zhan22]. By 2030, AI is expected to contribute up to $\$15.7$ trillion to the global Gross Domestic Product (GDP) [Rao17].

These impressive figures are substantiated by the recent breakthroughs in the field of generative AI, which demonstrate the social and economic impact of this technology on our society. Generative AI alone is expected to add as much as $\$4.4$ trillion in economic value by unlocking novel and enhancing existing use cases [Chui23a]. Whether looking at OpenAI's flagships ChatGPT [Open22a] and DALL·E 2 [Open22b] or Meta AI's Llama 2 [Touv23], the plethora of potential applications resulting from these systems in our daily and professional lives is enormous. According to McKinsey's 2023 state of AI report, less than one year after many of these tools emerged, one-third of the surveyed respondents state that they employ generative AI tools in at least one business function [Chui23b]. In addition, more than 40% of the respondents are determined to increase their organization's investments in AI as a direct consequence of the advances in generative AI [Chui23b].

One of the driving forces behind these recent breakthroughs in AI is a concept known as Self-Supervised Learning (SSL) [Ibra23]. Within a short amount of time, SSL has pushed the boundaries of Deep Learning (DL) across various domains by providing new means that enable learning from excessive amounts of unlabelled data [Ibra23]. By removing the dependency on carefully annotated datasets, SSL eliminates the existing bottleneck represented by conventional approaches, paving the way for more intelligent and generalist models [Lecu21; Ibra23]. In fact, leading AI researchers believe that SSL may unlock the next frontier of AI, as it represents arguably the most promising way to approximate a form of common sense in AI systems, which is considered to be the "dark matter of intelligence" [Lecu21].

Capital-intensive economic sectors, such as manufacturing, are projected to be among the main beneficiaries of the advent of AI [Rao17]. Companies operating in this sector are confronted with unprecedented opportunities to improve manufacturing processes along the three dimensions of time, quality, and cost [Jour21a]. The key enabling factor for harnessing the potential of AI in manufacturing lies in the simplified accessibility and increasing availability of manufacturing-related data in the wake of ongoing digitization efforts and Industry 4.0, which are transforming manufacturing systems into data-generating Cyber-Physical Systems (CPSs) [Jour21b]. As a consequence of the increasing availability of manufacturing-related data, researchers develop novel AI-based algorithms for use cases such as predictive maintenance, predictive quality, or process monitoring [Jour21a].

Monitoring discrete manufacturing processes to reliably detect anomalies and failures, is of fundamental relevance to reduce unplanned downtimes and ultimately gain a competitive advantage [Bieg22b]. The ramifications of unplanned downtimes in discrete manufacturing can be seen, e.g., in the automotive industry, where it is reported that one minute of downtime incurs costs of $\$20,000$ [Spie00; Zhan19a]. Multivariate Statistical In-Process Control (MSPC) represents one of the main areas of applied AI research that center around the detection of anomalies in process data and, as such, is heavily influenced by the developments of general Anomaly Detection (AD) research [Bieg22a; Bieg22b; Bieg23]. Recent findings from various domains demonstrate that the incorporation of SSL into AD methods leads to unprecedented performance levels and is regarded as a major milestone in the AD literature [Lizn21; Ruff21a]. However, while these findings suggest that the incorporation of SSL might be beneficial for MSPC in discrete manufacturing, research efforts in MSPC concentrate on established shallow or deep reconstruction-based methods. Consequently, there is currently no research that incorporates SSL into MSPC for discrete manufacturing processes.

1.1. Research objective

The objective of the underlying thesis is to address the aforementioned gap and as such encompasses:

Research objective

The development of an SSL approach for MSPC in discrete manufacturing processes.

This approach will henceforth be referred to as Self-Supervised Multivariate Statistical In-Process Control (SSMSPC). Based on this objective, the following research questions shall be answered.

Research question 1

How can SSL be incorporated into an MSPC system for discrete manufacturing processes?

Research question 2

What are the benefits of SSL compared to existing shallow and DL approaches for MSPC in discrete manufacturing processes?

Research question 3

How can SSMSPC be applied in practice to replace univariate post-process SPC, which is accepted in industry and embodied in today's norms?

1.2. Thesis structure

The underlying thesis is subdivided into seven chapters. Chapter 1 outlines the motivation, defines the research objective, and formulates the corresponding research questions that are to be answered throughout the course of this thesis. Following this, Chapter 2 establishes the theoretical foundation along the three pillars: (1) Machine Learning (ML), (2) AD, and (3) Statistical Process Control (SPC), upon which the subsequent chapters are built. Chapter 3 presents the state of the art by analyzing the results from two separately conducted systematic literature reviews in order to derive the research gap. Based on the identified research gap, Chapter 4 concretizes the research objective and the research questions. In addition, the internal and external requirements are derived and the problem statement for SSMSPC is formulated. In Chapter 5, the three components of SSMSPC are introduced, namely: (1) Location + Transformation prediction pretext task, (2) AD downstream task, and (3) control chart extension. This chapter represents the main scientific contribution of the thesis and provides the means to answer the first research question. Chapter 6 presents the validation of SSMSPC from two perspectives. First, the performance is evaluated and compared to state-of-the-art AD baselines using two real-world discrete manufacturing datasets. Second, the usability is evaluated and compared to the conventional SPC scheme used in discrete manufacturing practice by conducting a usability study with human machine operators in which a real-world discrete manufacturing process is monitored with both approaches. The findings from this chapter provide the necessary insights to answer the second and third research questions. Finally, Chapter 7 concludes this thesis by answering the research questions, summarizing the scientific contributions, discussing the limitations, and providing an outlook for future research directions. Figure 1.1 depicts an overview of the thesis structure.

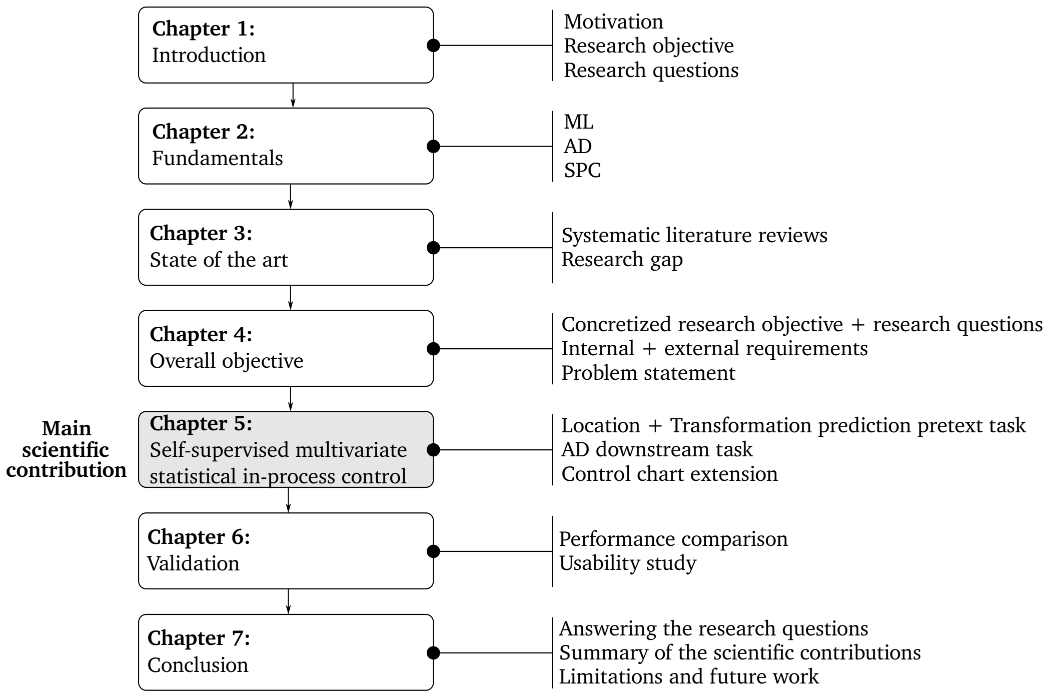

Chapter 1:
Introduction
Motivation
Research objective
Research questions
Chapter 2:
Fundamentals
ML
AD
SPC
Chapter 3:
State of the art
Systematic literature reviews
Research gap
Chapter 4:
Overall objective
Concretized research objective + research questions
Internal + external requirements
Problem statement
Main scientific contribution
Chapter 5:
Self-supervised multivariate statistical in-process control
Location + Transformation prediction pretext task
AD downstream task
Control chart extension
Chapter 6:
Validation
Performance comparison
Usability study
Chapter 7:
Conclusion
Answering the research questions
Summary of the scientific contributions
Limitations and future work

Figure 1.1.: Thesis structure, own illustration.

2. Fundamentals

This chapter develops the theoretical foundation upon which the subsequent parts of the thesis are built. Section 2.1 addresses the relevant aspects of ML. Section 2.2 presents the foundations of AD. Section 2.3 focuses on SPC in the context of discrete manufacturing processes. The chapter concludes with a short summary of the most important findings.

2.1. Machine learning

In this section, the required background knowledge with respect to ML is established. First, an overview of the main concepts of ML is provided. Following this, the emphasis will be placed on the principles of Neural Networks (NNs). Based on this foundation, Convolutional Neural Networks (CNNs) are presented as they embody the backbone of SSMSPC. In this context, the Continuous Wavelet Transformation (CWT) will be introduced, given its capability to transform a one-dimensional time series into a two-dimensional time-scale representation that can be processed by modern CNN architectures. Lastly, the fundamentals of SSL are elaborated.

2.1.1. Basic concepts and terminologies

As a subset of AI, ML is by its nature a multidisciplinary subject that is situated within the field of computer science and broadly relies on different branches of mathematics such as linear algebra, calculus, probability theory, information theory, etc., to learn models from data [Mitc97; Good16]. Specifically, the objective of ML is to learn models that automatically detect patterns in data in order to apply this knowledge, i.e., generalize to new, previously unseen data [Bish06; Murp12]. Here, the term model refers to a mathematical model that is implemented in a computer program [Zhan20]. Instead of explicitly designing a program to follow a certain behavior, in ML, the behavior of a program is defined by a set of parameters, also called weights, that are adapted based on the input data throughout a so-called learning phase [Zhan20]. The learning phase is also known as the training phase of the model, and the corresponding input data are referred to as training data [Bish06]. Summarizing the preceding statements, the concept of ML can be defined as follows:

Definition 2.1.1 (Machine learning). *"A computer program is said to learn from experience* E *with respect to some class of tasks* T *and performance measure* P*, if its performance at tasks in* T*, as measured by* P*, improves with experience* E*" [Mitc97].*

The task T describes the problem that ML shall solve, such as AD [Good16]. The performance measure P quantifies how good the ML model is in terms of solving T, e.g., the number of correctly identified anomalies [Good16]. The experience E, or rather the source through which experience is obtained, can be seen as the actual dataset that is used in the training phase, e.g., a dataset that contains both normal and anomalous parts of a machining process [Mitc97].
Note that the rationale for relying on Definition 2.1.1 instead of any other existing definition of ML is

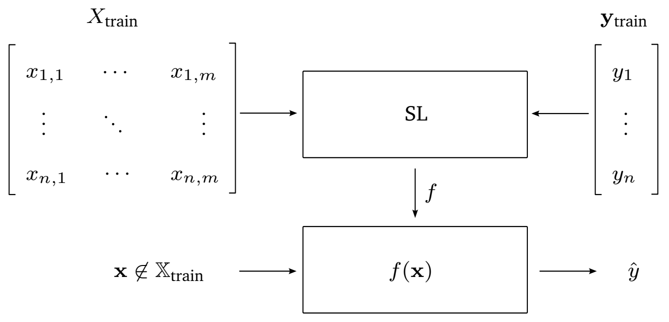
its formal yet intuitive character regarding the individual components that are relevant within the learning process.

Learning paradigms

ML problems can be broadly categorized into three so-called learning paradigms: (1) Supervised Learning (SL), (2) Unsupervised Learning (UL) and (3) Reinforcement Learning (RL) [Russ16]. Within the context of this thesis, the first two paradigms, i.e., SL and UL, are discussed in more detail. RL will be omitted due to its lack of relevance to the underlying work. The interested reader is encouraged to investigate [Sutt18] for an in-depth overview of RL.
SL is arguably the most prominent form of ML [Lecu15]. The main property of an SL problem is that the ML model has access to a set of so-called labels that provide the ground-truth for each example during training in order to learn the underlying relationship between inputs and outputs [Oliv18].

Definition 2.1.2 (Supervised learning). *In SL, the task of an ML model is to learn a mapping $f : \mathbb{X} \rightarrow \mathbb{Y}$ based on a set of training data $\mathbb{X}_{train} \subset \mathbb{X}$ and a set of labels $\mathbb{Y}_{train} \subset \mathbb{Y}$ [Zhou18].*

In general, a training example $\mathbf{x} \in \mathbb{X}_{\text{train}} \subset \mathbb{R}^m$ is an m-dimensional vector with each entry corresponding to a so-called feature, whereas the label that guides the learning process is typically a scalar $y \in \mathbb{Y}_{\text{train}} \subset \mathbb{R}$ [Murp12]. Assuming all n training data points $\mathbf{x}$ are of the same dimensionality, a common way to describe a dataset is to define a design or data matrix $X_{\text{train}} \in \mathbb{R}^{n \times m}$ that contains the feature observations for every data point and a corresponding label vector $\mathbf{y}_{\text{train}} \in \mathbb{R}^n$ that holds the labels [Good16]. Figure 2.1 provides an illustration of the general SL paradigm.



X_{train}
$x_{1,1}$ $\cdots$ $x_{1,m}$
$x_{n,1}$ $\cdots$ $x_{n,m}$
SL
$\mathbf{y}_{\text{train}}$
y_1
y_n
f
$\mathbf{x} \notin \mathbb{X}_{\text{train}}$
$f(\mathbf{x})$
$\hat{y}$

Figure 2.1.: SL paradigm, own illustration adapted from [Zhan20].

Depending on the characteristics of $\mathbb{Y}$, a given SL task can be described as a (1) classification or (2) regression problem [Hast08]. In a classification problem, the ML model is tasked with assigning a given input $\mathbf{x}$ to one of c discrete outputs [Murp12], e.g., classifying whether a produced part is OK or NOK based on an image of that part.

Definition 2.1.3 (Classification). *In a classification problem, the ML model learns a mapping $f : \mathbb{X} \rightarrow \mathbb{Y}$, where $\mathbb{Y} = \{1, \cdots, c\}$ [Good16].*

In a regression problem, the ML model is tasked with assigning a given input $\mathbf{x}$ to a continuous output [Murp12], e.g., predicting the remaining useful lifetime of a cutting tool based on the vibrations recorded by an accelerometer.

Definition 2.1.4 (Regression). *In a regression problem, the ML model learns a mapping $f : \mathbb{X} \rightarrow \mathbb{Y}$, where $\mathbb{Y} = \mathbb{R}$ [Good16].*

As opposed to SL, in UL, the ML model does not have access to ground-truth labels during training [Russ16]. Typical tasks in UL involve AD, clustering groups of similar data, estimating the distribution of the data in the input space, also called density estimation, or projecting the data from a high-dimensional input space to a lower-dimensional output space, which is referred to as dimensionality reduction [Bish06]. The learning outcomes from these tasks are then often used to extract relevant information for subsequent SL problems [Beng13c]. Due to its vital role in contemporary DL approaches in learning effective representations, see Section 2.1.4, UL has experienced a surge in relevance in recent years [Lecu15; Caro18].

Definition 2.1.5 (Unsupervised learning). *In UL, the task of an ML model is to learn a mapping $f : \mathbb{X} \to \mathbb{S}$, where $\mathbb{S}$ is an arbitrary set, to find and extract interesting associations, patterns, or representations from the input data $\mathbb{X}$ [Hast08].*

It is worth mentioning that UL is much less well-defined than SL, since it is often not clear what kind of associations, patterns, or representations the model shall look for or to what extent they actually exist in the data [Murp12]. Figure 2.2 provides an illustration of the concept of UL.

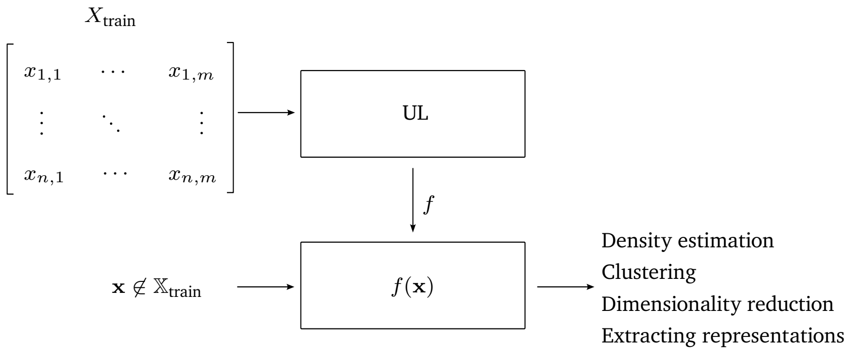

X_{train}
$x_{1,1}$ $\cdots$ $x_{1,m}$
$x_{n,1}$ $\cdots$ $x_{n,m}$
UL
f
$\mathbf{x} \notin \mathbb{X}_{\text{train}}$
$f(\mathbf{x})$
Density estimation
Clustering
Dimensionality reduction
Extracting representations

Figure 2.2.: UL paradigm, own illustration adapted from [Zhan20].

Loss functions

The training phase in ML usually involves the minimization of a so-called loss function $\mathcal{L} : \mathbb{R}^2 \to \mathbb{R}$ that rewards or penalizes the model for the decisions made with respect to the given task [Barb12]. The loss function is typically specified as a per-example loss function, which is averaged over each training example to retrieve the so-called training error [Hast08; Good16].
In SL problems, the loss function relies on the label information y and the output of the model $\hat{y} = f(\mathbf{x})$ to determine the quality of the prediction [Russ16]. One of the most commonly used loss functions for regression problems is the Mean Squared Error (MSE) [Hast08; Good16; Zhan20]. The MSE simply computes the average of the squared differences between the ground-truth label and the model output across all training examples, following Equation 2.1

$$\mathcal{L}_{\text{MSE}} := \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2. \tag{2.1}$$

Conversely, in terms of classification problems, a popular loss function is the Cross-Entropy (CE) loss, which is given by Equation 2.2 [Hast08; Zhan20]

$$\mathcal{L}_{\text{CE}} := -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} y_{i,j}\log(f(\mathbf{x}_i)). \tag{2.2}$$

CE is a loss function that has its origins in the field of information theory and is often used in conjunction with NNs [Bish06; Stra19]. In fact, CE has replaced the MSE loss function as the standard for NNs as it overcomes the problems of saturation and slow learning in gradient-based optimization [Good16], see Section 2.1.2 for more details on gradient-based optimization. Intuitively, CE tries to maximize the likelihood of the observed data and minimize the expected surprise between the observed true labels $\mathbf{y}$ and the model output $\hat{\mathbf{y}}$ [Zhan20]. Equation 2.2 assumes that the labels are one-hot-encoded, i.e., each label is represented as a c-dimensional vector where the index corresponding to the true class is one and all other entries are zero [Jame13].
Considering UL problems, the learning objectives are often reformulated in such a way that a label alternative is retrieved based on the specific task at hand to create a suitable loss function, e.g., reconstructing the input [Good16].

Evaluation metrics

Evaluation metrics represent human-interpretable and task-specific performance measures that are used to assess a model after training. These evaluation metrics typically differ from the loss function that is employed to train a model [Good16]. Reasons for this include, e.g., issues in optimizing the desired evaluation metric [Deis20] or interpreting the employed loss function with respect to the problem at hand. As an example for this, consider the error rate, which is easy to interpret as an evaluation metric but intractable to optimize as a loss function. Conversely, the CE loss can be conveniently optimized, however, it is challenging to interpret from a human perspective [Good16]. This paragraph restricts itself to the presentation of evaluation metrics for classification, specifically binary classification, i.e., where $c = 2$, as this represents the setting for which SSMSPC is designed. For a more thorough analysis of evaluation metrics, the interested reader is encouraged to investigate [Kuhn19].
A prominent tool to assess the performance of a binary classifier is the confusion matrix [Kuhn13], see Table 2.1. The confusion matrix is a $c \times c$ matrix where the columns correspond to the ground-truth labels and the rows correspond to the predicted labels of the model [Zhan20].

Table 2.1.: Confusion matrix, own illustration adapted from [Ande03; Murp12].

		Ground-truth	
		$y = 1$	$y = 0$
Predicted	$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
	$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Here, the negative class corresponds to $y = 0$, whereas the positive class corresponds to $y = 1$. When a model's decision is in alignment with the ground-truth label, i.e., $\hat{y} = y$, the example is called a TP in the case of a positive example and a TN otherwise. Thus, the entries on the main diagonal of a confusion matrix represent the number of instances that have been correctly classified. If a model's decision deviates from the ground-truth label, i.e., $\hat{y} \neq y$, the result is either an FP also called false alarm ($\hat{y} = 1$), or an FN ($\hat{y} = 0$). FP and FN are also referred to as type I error and type II error, respectively [Grin97]. These error types play a vital role in determining the decision threshold for AD methods that are going to be discussed in Section 2.2.1.
The confusion matrix allows to derive a wide range of useful evaluation metrics. For the purpose of this thesis, the focus will be placed on precision, recall, and f1-score. The rationale for this is that these metrics represent the first choice in situations of imbalanced datasets, such as in the case of AD

tasks, when there are much more normal ($y = 0$) than anomalous instances ($y = 1$) [Agga17]. The precision quantifies the proportion of correctly predicted positive examples among all examples that the model predicted as being positive, according to Equation 2.3

$$\text{precision} := \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.3}$$

Conversely, the recall quantifies the number of correctly predicted positive examples in relation to all positive examples that are present in the dataset and is given by Equation 2.4

$$\text{recall} := \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.4}$$

The f1-score combines both precision and recall into a single metric that is defined by Equation 2.5

$$\text{f1-score} := \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{2.5}$$

Note that precision and recall need to be assessed in conjunction since it is straightforward to create, e.g., a naive classifier that achieves perfect recall by simply flagging every example as positive.

Model selection and model assessment

Recall that the overarching objective in ML is to learn models that are able to generalize and perform well on previously unseen data [Barb12]. The so-called "no free lunch theorem" states that there is no ML model that universally performs best on every possible problem class [Wolp97]. Thus, meeting the aforementioned objective involves two consecutive steps that need to be taken into account: (1) model selection and (2) model assessment [Hast08].

Definition 2.1.6 (Model selection). *In the model selection step, a proportion of the training data called the validation set is used to fine-tune specific model settings, also known as hyperparameters, to select the best-performing model f_i^* from the set of v potentially suitable models $\{f_1, \cdots, f_v\}$ for the given task [Hast08; Jame13; Niel15].*

Definition 2.1.7 (Hyperparameter). *Hyperparameters collectively represent model settings that are not learned during training but have to be chosen manually before training [Good16].*

Definition 2.1.8 (Model assessment). *In the model assessment step, the best-performing model f_i^* from the model selection step is evaluated on a holdout test dataset to estimate the generalization error [Hast08; Good16; Kuhn19].*

The test dataset represents a fixed proportion of the available data for the task that is reserved prior to training [Deis20]. Note that the test set performance of the selected model is measured only once and that the test data are not used for model selection purposes, as this would result in an unrealistically optimistic estimate of the generalization error [Murp12; Zhan20]. This is the reason why the validation set is constructed from the training set and not from the test set [Good16]. Summarizing, the model selection and model assessment steps require that the available data are split into three separate sets: (1) a training set, (2) a validation set, and (3) a test set. Both training and validation sets are used for model selection, whereas the test set is used for model assessment. Figure 2.3 visualizes the preceding statements. It is worth mentioning that there is no general rule to determine the split ratio between train, validation, and test set, as this depends on various factors, such as the number of available data, the task, etc. [Hast08; Murp12]. Note also that an alternative way of handling data in the model selection step is to use cross-validation, which is especially useful in situations where data are scarce, see, e.g., [Bish06].

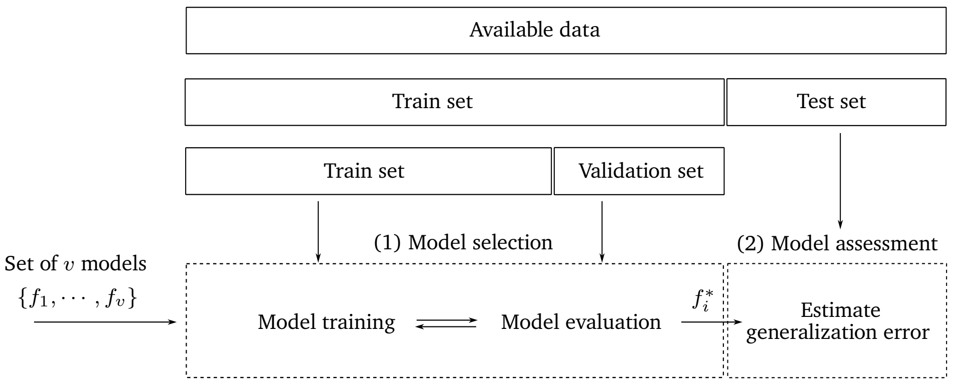


Figure 2.3.: Dataset splits utilized in the model selection and model assessment steps, own illustration.

Overfitting, underfitting and bias-variance trade-off

The phenomenon of overfitting is one of the major challenges that researchers and practitioners alike are confronted with in ML [Good16]. Intuitively, the overfitting problem describes a situation in which a model performs well on the training data but is not able to generalize to new data, i.e., shows poor performance on the test data [Shal14].

Definition 2.1.9 (Overfitting). *A model f suffers from overfitting if there exists another model f', such that f achieves a lower training error than f' but f' achieves a lower generalization error than f on the given task [Mitc97].*

An explanation for this phenomenon can be found in the capacity or complexity of a model, which can be thought of as the ability to adapt to the characteristics of the training data [Good16]. Specifically, the higher the complexity of a model, the higher its ability to model every minor detail in the training data, i.e., the model adapts to specific patterns of the training data that are not part of the general underlying dependency [Murp12; Kuhn19]. In the extreme case, this leads to the situation where a model starts to memorize the training data, i.e., the training error converges to zero [Jame13]. Note that overfitting is also the reason why the performance on the test set in the model assessment step is evaluated only once, as the model could otherwise adapt its parameters to the characteristics of the test data, which would result in seriously underestimating the generalization error [Zhan20]. Aside from increasing the number of training examples, which is in many situations unfeasible, another effective way to encounter overfitting is the idea of regularization, which will be presented in more detail in Section 2.1.2.
The opposite extreme of overfitting is referred to as underfitting.

Definition 2.1.10 (Underfitting). *A model f suffers from underfitting when the model complexity is too low, i.e., the model is not able to achieve a sufficiently low training error [Murp12; Good16].*

From a statistical point of view, choosing the right model complexity for a given task is also known as the bias-variance trade-off [Bish06; Shal14]. A model that underfits the data has high bias and low variance, whereas in the case of overfitting, the model has low bias and high variance [Stra19]. Figure 2.4 illustrates the bias-variance trade-off in ML. Naturally, the objective is to search for the optimum proportion between bias and variance in order to find the "right" model complexity.

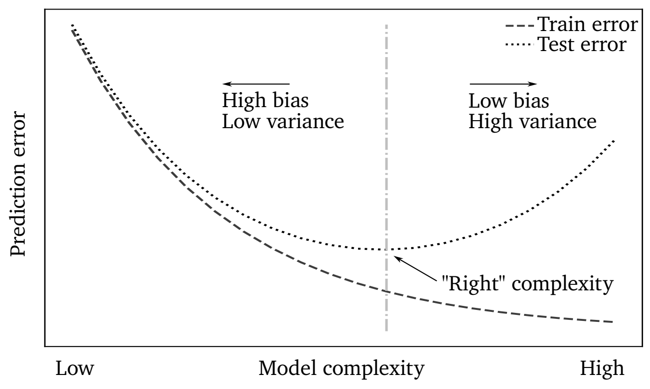


Figure 2.4.: Bias-variance trade-off when choosing model complexity, own illustration adapted from [Hast08].

Data preprocessing

Preprocessing the input data prior to training is an important component for practical applications that can strongly affect the generalization performance of an ML model [Bish96]. One of the most essential preprocessing steps is feature scaling, i.e., transforming the features of the raw data to lie on the same scale. The reason for this is that the performance of many ML models is negatively impacted by features that have widely varying scales [Gero19]. Specifically, models that involve some form of distance computation or rely on gradient-based optimization schemes for training, such as NNs, experience low convergence speed and bad performance if the input features are not on the same scale [Lecu98b; Wies11].
For this purpose, z-score scaling and min-max scaling represent two popular feature scaling approaches [Zhen18; Kuhn19]. Given a data matrix $X_{\text{train}} \in \mathbb{R}^{n \times m}$ and an input vector $\mathbf{x} \in \mathbb{R}^m$, z-score scaling transforms the data to have a zero mean and unit variance according to Equation 2.6

$$\tilde{\mathbf{x}} = \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}}, \tag{2.6}$$

where $\tilde{\mathbf{x}}$ represents the scaled vector and $\hat{\mu} \in \mathbb{R}^m, \hat{\sigma} \in \mathbb{R}^m$ hold the estimated means and standard deviations, respectively, for each feature of X_{train}.
Conversely, min-max scaling is typically used for images and transforms the data to lay in the interval [0;1], and is defined according to Equation 2.7 [Good16]

$$\tilde{\mathbf{x}} = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}. \tag{2.7}$$

Here, $\max(\mathbf{x}) \in \mathbb{R}^m$ and $\min(\mathbf{x}) \in \mathbb{R}^m$ represent the corresponding max and min values of the features in X_{train}. Appendix A.1.1 provides a comprehensive example that demonstrates the necessity of feature scaling.

Deep learning

The performance of conventional, so-called shallow ML models such as linear regression, decision trees, etc. strongly relies on manually engineered features, i.e., data representations designed by human experts. The reason for this is that these models lack the capability of extracting useful representations directly from raw data, such as image pixels [Beng13c]. Since the design of effective handcrafted features is very resource-intensive, researchers in ML increasingly focus on the development of models that possess the ability to automatically learn useful features from raw data [Zhan20]. This area of ML research is known as representation learning [Beng13a].

Definition 2.1.11 (Representation learning). *"Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification" [Lecu15].*

As a particular form of representation learning, DL is a research field that emerged in 2006 following the breakthroughs by [Hint06] to successfully train deep model architectures and automatically learn multiple levels of abstract representations from raw data [Beng07b; Glor11; Beng13a]. Since then, research in DL has seen a tremendous increase in popularity, with countless papers being published in leading ML conferences exceeding the state-of-the-art performance by a large margin in areas such as object detection and speech recognition [Hint12; Kriz12; He15].

Definition 2.1.12 (Deep learning). *"[...][DL] is a particular kind of [...][ML] that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones" [Good16].*

Research results from the field of neuroscience indicate that the primal visual system processes information in a sequential manner via multiple levels of abstraction [Serr05; Serr07]. DL attempts to incorporate this way of information processing into a computer [Murp12].
The term "deep" in DL refers to the number of layers of subsequently concatenated non-linear computation modules in the learned mapping [Beng09]. Theoretical findings suggest that deep architectures are more efficient than shallow architectures with respect to the required number of computational components and parameters in learning mappings that are able to represent high-level abstractions [Beng07a; Beng11; Beng13b]. In particular, there are two major advantages that result from applying deep architectures: (1) reuse of features and (2) learning abstract features that are built on less abstract features with increasing depth [Beng13c]. Intuitively, reuse of features refers to the ability to construct various high-level features by reusing low-level features, which grows exponentially in depth [Beng13b]. Note that there is no formal definition of how "deep" an architecture needs to be in order to qualify as a deep architecture [Good16]. Some researchers provide a rather informal rule in which models that are composed of one to three layers are considered shallow, and models with more than three layers are considered deep [Beng07b; Beng09; Chol17].

2.1.2. Neural networks

NNs represent a class of ML models that form the basis of modern DL approaches [Good16; Jing20]. The so-called perceptron represents the main building block of an NN. Its development was inspired by the work of [McCu43] and dates back to [Rose58]. A perceptron computes the linear combination between an input $\mathbf{x} \in \mathbb{R}^m$ and a weight vector $\mathbf{w} \in \mathbb{R}^m$ to which an additional bias parameter w_0 is added, according to Equation 2.8

$$w_0 + w_1 x_1 + \cdots + w_m x_m = w_0 + \sum_{i=1}^{m} w_i x_i = w_0 + \mathbf{w}^T \mathbf{x}. \tag{2.8}$$

Both weights and bias represent the learnable parameters. The name bias refers to the fact that the output of this weighted sum is biased with respect to w_0 in the case of $\mathbf{x} = \mathbf{0}$ or $\mathbf{w} = \mathbf{0}$ [Good16]. Note that for notational convenience, it is common to incorporate the bias term into $\mathbf{w}$ and expand $\mathbf{x}$ with an additional constant $x_0 = 1$, so that $\mathbf{x} \in \mathbb{R}^{m+1}$ and $\mathbf{w} \in \mathbb{R}^{m+1}$ [Mitc97]. The result of Equation 2.8 is subsequently fed into a so-called activation function ρ, which in the case of the original perceptron, as proposed by [Rose58] is a simple threshold function given by Equation 2.9

$$\rho(\mathbf{w}^T\mathbf{x}) = \begin{cases} 1, & \mathbf{w}^T\mathbf{x} > \delta \\ -1, & \text{otherwise} \end{cases}, \tag{2.9}$$

where δ is the selected threshold. Thus, a perceptron is a typical example of a linear model that learns a mapping $\phi : \mathbb{R}^m \to \mathbb{R}$ and has historically been used for binary classification [Bish96].

Multi-layer perceptron

A Multi-Layer Perceptron (MLP) is a composition of multiple perceptrons that are chained together in a fully connected layered structure, which is why they are also known as Fully Connected Networks (FCNs) [Bish96]. As such, they represent a composition of multiple mappings $\phi_1, \cdots, \phi_{l+1}$ [Good16]. Compared to the original perceptron, MLPs differ in three aspects: (1) the output of ϕ_i is a vector $\mathbf{a}_i$ and is referred to as the i-th layer in the network, (2) the activation functions $\rho_1, \cdots, \rho_{l+1}$ are no longer simple threshold functions but specific nonlinear and differentiable functions [Bish94], (3) instead of a single weight vector $\mathbf{w}$, the layers are connected with weight matrices $W_1, \cdots, W_{l+1}$. Figure 2.5 shows an illustration of a typical MLP that takes as input a vector $\mathbf{x} \in \mathbb{R}^m$ and outputs a vector $\mathbf{y} \in \mathbb{R}^c$.

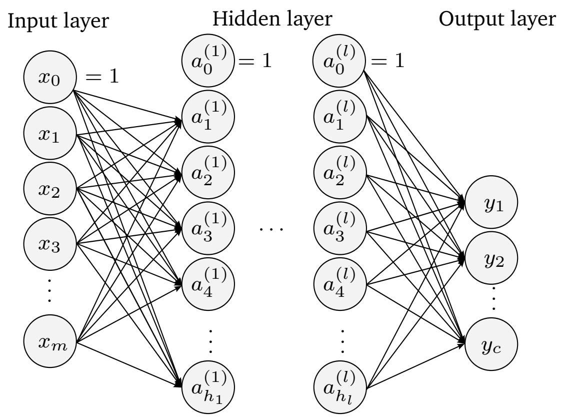


Figure 2.5.: Schematic overview of an MLP, own illustration adapted from [Zhan20].

The first and last layers of an MLP are referred to as the input layer and output layer, respectively, whereas the l intermediate layers are called hidden layers [Zhan20]. In alignment with the description of deep architectures in previous paragraphs, MLPs are called shallow when $l \leq 1$ and deep when $l > 1$ [Beng07b; Beng09; Chol17].
Several researchers, e.g., [Cybe89; Funa89; Horn89] have demonstrated that a shallow MLP is a so-called universal function approximator [Chen94]. However, as stated before, the inefficiency of shallow architectures regarding the required number of computational components and parameters to represent any arbitrary function is usually prohibitively large, which motivates the application of deep architectures [Good16].

Activation functions

Activation functions are crucial for the success of contemporary NNs, as they embody the key ingredient that enables a network to learn arbitrarily complex mappings [Zhan20]. An important requirement to leverage the potential of activation functions is that they satisfy the condition of being both nonlinear and differentiable in order to enable gradient-based optimization [Chen94].
Early works on MLPs focused on sigmoid or tanh as the activation function, as they represent both a smooth and differentiable approximation to the threshold activation that was used in the original perceptron [Bish96; Bish06; Shal14]. However, researchers discovered that the so-called Rectified Linear Unit (ReLU) activation function (Equation 2.10) and its relatives, such as leaky ReLU (Equation 2.11), led to dramatic improvements in both training and performance due to their convenient mathematical properties [Jarr09; Nair10; Glor11; Xu15]. In fact, replacing sigmoid and tanh activation functions with ReLU has been one of the major milestones in successfully training deep networks [Beng13a]. As of today, ReLU-based activation functions have established themselves as the standard for contemporary networks [Good16; Zhan20]

$$\mathrm{ReLU}(x) = \max(0, x) \qquad \frac{d}{dx}\mathrm{ReLU}(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2.10}$$

$$\text{Leaky ReLU}(x) = \begin{cases} x, & x > 0 \\ \beta x, & x \leq 0, \beta \in [0; 1] \end{cases} \qquad \frac{d}{dx}\text{Leaky ReLU}(x) = \begin{cases} 1, & x > 0 \\ \beta, & x \leq 0 \end{cases}. \tag{2.11}$$

The presented activation functions are typically employed in the context of hidden layers. Regarding the output layer, the activation function ρ_{l+1} usually differs from the l activation functions in the hidden layers [Bish94; Beng09]. The rationale for this is that the last activation determines the final output of the network and thus depends on the problem at hand. Arguably the most popular choice for the last activation function, in terms of classification problems, is the so-called softmax function, which is displayed in Equation 2.12 [Hint12; Liu16]

$$\mathrm{softmax}(a_i^{(l+1)}) = \hat{y}_i = \frac{e^{a_i^{(l+1)}}}{\sum_{k=1}^{c} e^{a_k^{(l+1)}}} \; \forall i \in \{1, \cdots, c\}. \tag{2.12}$$

The intriguing property of the softmax function is that its output can be interpreted as a probability mass function over the c available classes, i.e., $\sum_{i=1}^{c} \hat{y}_i = 1$. Thus, the maximum element of the softmax output represents the class to which the network assigns the highest probability.

Gradient-based optimization

Recall from Section 2.1.1 that the training phase of ML models involves the minimization of a problem-specific loss function. In terms of NNs, analytical closed-form solutions to these optimization problems do typically not exist [Bish96; Zhan20]. Thus, training NNs strongly relies on optimization techniques, specifically gradient-based optimization techniques that search iteratively for a good local minimum of the loss function [Mitc97; Good16; Stra19].
The most basic approach in the context of gradient-based optimization is the Batch Gradient Descent (BGD) algorithm [Niel15; Rude16]. This algorithm applies the update rule given by Equation 2.13 for t iterations to each weight in the network, starting from an initial guess until a certain stopping criterion is met [Bish94; Boyd04; Nest04; Shal14]

$$\mathbf{w}_{j+1} \leftarrow \mathbf{w}_j - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_{\mathbf{w}} \mathcal{L}(\phi(\mathbf{x}_i; \mathbf{w}_j), \mathbf{y}_i) \; \forall j \in \{0, \cdots, t-1\}. \tag{2.13}$$

Here, $\mathbf{w}_j$ is the vector that contains the weights from every layer at iteration j, $\gamma > 0$ is the learning rate or step size [Stra19], and $\nabla_{\mathbf{w}}$ is the vector of first partial derivatives with respect to each weight in the network. Note that, for notational convenience, the $l+1$ weight matrices $\{W_1, \cdots, W_{l+1}\}$ in the example above are collapsed into a single vector $\mathbf{w}$. Looking at Equation 2.13, it can be observed that BGD computes the gradient updates at iteration j with respect to the entire training dataset, which is computationally intractable for many real-world applications [Zhan20].
An alternative to BGD is Stochastic Gradient Descent (SGD). SGD is of major importance for NN training, as it embodies the core of many modern optimization algorithms [Hard16; Liu20; Ahn22]. In SGD, the exact computation of the gradient is replaced by a noisy gradient estimate that is computed according to Equation 2.14 based on a small, randomly sampled mini-batch $\mathbb{B} \subset \mathbb{X}_{\text{train}}$ of training data points in each iteration [Lecu98a; Bott12; Hint12; Hard16]

$$\mathbf{w}_{j+1} \leftarrow \mathbf{w}_j - \gamma \frac{1}{|\mathbb{B}|} \sum_{i=1}^{|\mathbb{B}|} \nabla_{\mathbf{w}} \mathcal{L}(\phi(\mathbf{x}_i; \mathbf{w}_j), \mathbf{y}_i) \; \forall j \in \{0, \cdots, t-1\}. \tag{2.14}$$

The number of randomly sampled training points $|\mathbb{B}|$ is known as the batch size [Smit18]. A complete pass through all $\lfloor n/|\mathbb{B}| \rfloor$ batches is referred to as an epoch [Stra19]. Both batch size and number of epochs are important hyperparameters in training NNs. Some researchers refer to Equation 2.14 as Mini-Batch Gradient Descent (MGD) and treat BGD and SGD as special cases of MGD, where $|\mathbb{B}| = n$ and $|\mathbb{B}| = 1$, respectively [Murp12; Zhan20].
Adaptive Moment Estimation (ADAM) [King15; Oliv18; Ma19] is a prominent example of an optimization algorithm based on SGD that has led to important advances in NN training, and is used for optimization purposes in this work. See Appendix A.1.2 for more detailed information.

Backpropagation

Gradient-based optimization techniques require an efficient way to compute the gradient of the employed loss function with respect to the weights and biases of an NN [Bish06]. The state-of-the-art approach for this purpose is referred to as backpropagation, which is a special case of a more general concept known as automatic differentiation [Bayd18].
Backpropagation follows a two-stage scheme [Rume86; Hast08]. The first stage involves a forward pass through the network to compute the loss based on a mini-batch of training data and its corresponding labels. The second stage consists of a backward pass through the network to propagate the errors from the output layer back to the input layer. The gradients are computed within the backward pass using the chain rule from calculus [Rume86; Lecu15], see Appendix A.1.3 for a comprehensive example.

Regularization

Given the susceptibility of NNs to overfit, regularization embodies a crucial component in achieving a low generalization error [Lawr00; Cogs16].

Definition 2.1.13 (Regularization). *"Regularization is any modification [...][applied] to a learning algorithm that is intended to reduce its generalization error but not its training error [Good16]."*

Arguably, one of the most widely used forms of regularization is the so-called ℓ_2 regularization [Zhan20]. The intuition of this form of regularization is to limit the unconstrained growth of weights by including an additional term to the loss function that penalizes large weights using the ℓ_2 norm

[Krog91]. The ℓ_2-regularized loss $\mathcal{L}_{\text{reg}}$ for a mini-batch $\mathbb{B}$ is given by Equation 2.15

$$\mathcal{L}_{\text{reg}} = \frac{1}{|\mathbb{B}|}\sum_{i=1}^{|\mathbb{B}|}\mathcal{L}((\phi(\mathbf{x}_i), \mathbf{w}), \mathbf{y}_i) + \frac{\lambda}{2}||\mathbf{w}||_2^2, \tag{2.15}$$

where λ represents the regularization parameter, which is a hyperparameter that is fitted to the validation set and controls the extent to which growing network parameters are penalized [Pogg85; Giro95; Zhan20]. In terms of conventional SGD, the updating rule at iteration $j+1$ is thus given by Equation 2.16

$$\mathbf{w}_{j+1} \leftarrow (1-\gamma\lambda)\mathbf{w}_j - \gamma\frac{1}{|\mathbb{B}|}\sum_{i=1}^{|\mathbb{B}|}\nabla_{\mathbf{w}}\mathcal{L}(\phi(\mathbf{x}_i;\mathbf{w}_j), \mathbf{y}_i). \tag{2.16}$$

For $\lambda = \lambda'/\gamma$, Equation 2.16 is frequently referred to as weight decay, as described in [Hans88; Losh19]. However, even though weight decay and ℓ_2 regularization can be made equivalent in the case of conventional SGD, this is not generally the case for adaptive gradient methods, such as ADAM, as pointed out by [Losh19; Zhan19b]. This can be observed by the fact that there is no parameter λ for which the standard ℓ_2 regularization, as given in Equation 2.15, would result in the conventional weight decay [Losh19]. To achieve similar properties to weight decay, the standard ℓ_2 regularization needs to be scale-adjusted [Zhan19b].

Learning rate and learning rate schedules

The learning rate is considered one of the most important hyperparameters when it comes to training NNs [Beng12]. It represents a scalar value that controls the step size in the direction of the negative gradient, as shown in Equation 2.13. If the learning rate is chosen to be too large, the optimization algorithm can diverge, whereas if the learning rate is chosen to be too small, the optimization might get stuck in bad local optima or converge slowly [Zhan20].
Empirical evidence suggests that the learning rate should be adjusted during training with the help of a so-called learning rate schedule to speed up training and improve generalization performance [Dark92; Li20]. A frequently employed learning rate schedule that is of relevance for this thesis is the cosine learning rate schedule, see, e.g., [Hend19; Liu20; Park20; Sehw21], which has been proposed by [Losh17]. The idea of this approach is to apply periodic warm restarts every ϵ epochs and decay the learning rate with a cosine-type annealing strategy for each batch.

Batch normalization

Batch Normalization (BN) [Ioff15] represents arguably one of the major architectural contributions in the field of DL that shortens the training time of deep NNs, enables larger learning rates, and improves generalization performance [Sant18].
Recall from Section 2.1.1 that scaling the input data prior to training increases the convergence speed of gradient-based optimization algorithms and generally leads to improved performance. In essence, BN generalizes this concept to each layer of a deep NN [Stra19]. Specifically, consider a mini-batch $\mathbb{B} = \{\mathbf{z}_1^{(i)}, \cdots, \mathbf{z}_b^{(i)}\} \subset \mathbb{R}^m$ of input samples for the i-th hidden layer, where $\mathbf{z}_j^{(i)} = W_i^T\mathbf{a}_j^{(i-1)}\ \forall j \in \{1, \cdots, b\}$. The idea of BN is to apply Equation 2.17 to each $\mathbf{z}^{(i)} \in \mathbb{B}$ [Ioff15]

$$\text{BN}(\mathbf{z}_j^{(i)}) = \eta^{(i)}\frac{\mathbf{z}_j^{(i)} - \hat{\mu}_{\mathbb{B}}}{\sqrt{\hat{\sigma}_{\mathbb{B}}^2 + \epsilon}} + \kappa^{(i)}\ \forall j \in \{1, \cdots, b\}, \tag{2.17}$$

where $\hat{\mu}_{\mathbb{B}}, (\hat{\sigma}_{\mathbb{B}}^2 + \epsilon)^{1/2} \in \mathbb{R}^m$ correspond to the estimated mean and standard deviation of the mini-batch. Note that ϵ represents a small constant that is used to avoid division by zero if the estimated variance vanishes. The parameters $\eta^{(i)}, \kappa^{(i)} \in \mathbb{R}^m$ are learned during training and serve the purpose of maintaining the expressive power of the layer [Good16]. Concretely, when setting $\eta^{(i)} = (\hat{\sigma}_{\mathbb{B}}^2 + \epsilon)^{1/2}$ and $\kappa^{(i)} = \hat{\mu}_{\mathbb{B}}$, the BN transformation represents the identity function and thus recovers the original input values [Ioff15]. Since updating $\eta^{(i)}$ and $\kappa^{(i)}$ requires the computation of the respective gradients via backpropagation, BN is a differentiable transformation.
The scheme presented above describes the application of BN during the training phase of a NN. In the prediction phase, the behavior of BN slightly changes [Zhan20]. More precisely, once the training is completed, the batch-dependent mean and variance parameters are replaced by fixed running averages that were computed across all batches during training [Bjor18; Zhan20].
It is worth noting that the reasons for the general success of BN are still not entirely understood [Huan23]. The initial motivation of BN was to encounter the so-called internal covariate shift in deep NNs, i.e., the change in the distribution of network activations across layers that is caused by continuously changing network parameters during training [Ioff15]. However, in recent years, researchers have demonstrated that the success of BN rather stems from other factors such as (1) that it smooths the optimization landscape, which leads to more stable gradient behavior and (2) that it enables the usage of larger learning rates [Bjor18; Sant18].

2.1.3. Convolutional neural networks

The discussion of NN types has so far been limited to MLPs, which represent powerful ML models for a variety of applications that involve tabular data [Zhan20]. However, in the context of data types such as images, MLPs suffer from certain deficiencies that limit their effectiveness [Lecu98a]. First, the capacity of MLPs quickly grows unreasonably large with increasing input sizes [Lecu98a]. Consider, e.g., a grayscale image with 100×100 pixels. To process images of this size, a simple MLP consisting of one hidden layer with m hidden units would contain $> 10^4 \cdot m$ network parameters in the first weight matrix alone. Second, they lack built-in invariance in terms of translations or local distortions of the input data [Lecu98a]. Third, they ignore the topology of the input data, i.e., the features can be arranged in any fixed order without affecting the training outcome of the network [Lecu98a]. This is particularly problematic with respect to the local structure of image data, which causes nearby pixels to be highly correlated.
As an alternative to MLPs, CNNs have established themselves as the state-of-the-art approach for computer vision-related tasks, such as image recognition, object detection, or semantic segmentation [Zhan20]. These networks provide effective mechanisms to encounter the deficiencies of MLPs by incorporating three architectural ideas: (1) local receptive fields, (2) shared weights, and (3) spatial or temporal subsampling, also known as pooling [Lecu98a; Niel15]. CNNs are inspired by the works of [Hube62]. They discovered that the cat's primary visual cortex contains orientation-selective simple cells with local receptive fields and complex cells [Hube62; Beng09; Lecu10]. These cells are arranged in a hierarchy to build increasingly complex and invariant object representations [Serr07]. Based on these findings, [Fuku80] proposed the neocognitron which marked the first computerized version of such cells in a trainable model [Beng09]. Specifically, they introduced simple cell layers and complex cell layers using cell planes whose cells have receptive fields with the same functionality that are located at different spatial positions [Fuku80]. [Lecu89] then established the modern framework of CNNs by presenting the LeNet, a simplified architecture of the neocognitron that enabled the incorporation of backpropagation to allow training in a supervised fashion [Lecu10; Gu18].
A CNN for classification typically consists of three components: (1) convolutional layers, (2) pooling

layers, and (3) a classification module that is represented by an MLP [Gu18].

Convolutional layer

A convolutional layer resembles the simple cells in the model of [Hube62] and is used to learn feature representations by computing the two-dimensional discrete convolution, or more precisely, the two-dimensional cross-correlation between the input and a set of filters, also known as kernels [Gu18]. A kernel $\mathbf{K} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ represents a third-order tensor with learnable weights, where d_1, d_2 are the height and width and d_3 corresponds to the number of input channels, e.g., in the case of a Red-Green-Blue (RGB) color image, $d_3 = 3$. Suppose k_j kernels are used in the j-th layer, i.e., $\mathbf{K}_1^{(j)}, \cdots, \mathbf{K}_{k_j}^{(j)}$. Convolving these kernels with the input and then applying an activation function, such as ReLU, results in a total of k_j so-called feature maps $A_1^{(j)}, \cdots, A_{k_j}^{(j)}$, where $A_i^{(j)} \in \mathbb{R}^{n_1 \times n_2}$ $\forall i \in \{1, \cdots, k_j\}$ [Lecu10]. Each unit of feature map $A_i^{(j)}$ is connected to a specific patch of units in the previous layer, which is referred to as the unit's local receptive field [Lecu98a; Beng09]. Here, one of the fundamental architectural advantages of CNNs becomes apparent, namely that the weights of kernel $\mathbf{K}_i^{(j)}$ are shared among the units in feature map $A_i^{(j)}$ [Gu18].

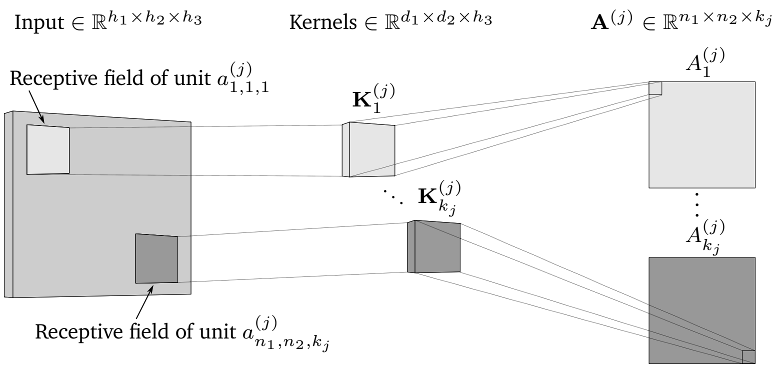

Input $\in \mathbb{R}^{h_1 \times h_2 \times h_3}$
Kernels $\in \mathbb{R}^{d_1 \times d_2 \times h_3}$
$\mathbf{A}^{(j)} \in \mathbb{R}^{n_1 \times n_2 \times k_j}$
Receptive field of unit $a_{1,1,1}^{(j)}$
$\mathbf{K}_1^{(j)}$
$A_1^{(j)}$
$\mathbf{K}_{k_j}^{(j)}$
$A_{k_j}^{(j)}$
Receptive field of unit $a_{n_1,n_2,k_j}^{(j)}$

Figure 2.6.: Local receptive fields and weight sharing mechanism in a CNN, own illustration.

The rationale for this is twofold [Lecu98a]. First, sharing weights drastically reduces the number of learnable parameters. Second, it introduces a kind of shift invariance that allows the individual kernels to detect different features in the input independent of their location [Beng07a; Lecu15]. Figure 2.6 displays the idea of local receptive fields and the weight sharing mechanism. It is worth mentioning that the height, width, and number of kernels in a convolutional layer represent hyperparameters that need to be specified in advance.
The two-dimensional cross-correlation between kernel $\mathbf{K}_i^{(j)}$ and the feature maps of the previous layer $\mathbf{A}^{(j-1)}$ is given by Equation 2.18 [Good16; Zhan20]

$$a_{p,q,i}^{(j)} = \rho^{(j)} \left(\sum_{m=1}^{d_1} \sum_{n=1}^{d_2} \mathbf{A}^{(j-1)}(p+m-1, q+n-1) \mathbf{K}_i^{(j)}(m,n) \right), \tag{2.18}$$

where $a_{p,q,i}^{(j)}$ corresponds to element (p,q) in feature map $A_i^{(j)}$. Figure 2.7 provides a schematic illustration of the cross-correlation operation that is used to compute the units of the individual feature maps. It can be seen that the computation of a single unit of feature map $A_i^{(j)}$ essentially boils down to the dot product between the respective patches of input units and the corresponding weights and bias of kernel $\mathbf{K}_i^{(j)}$.

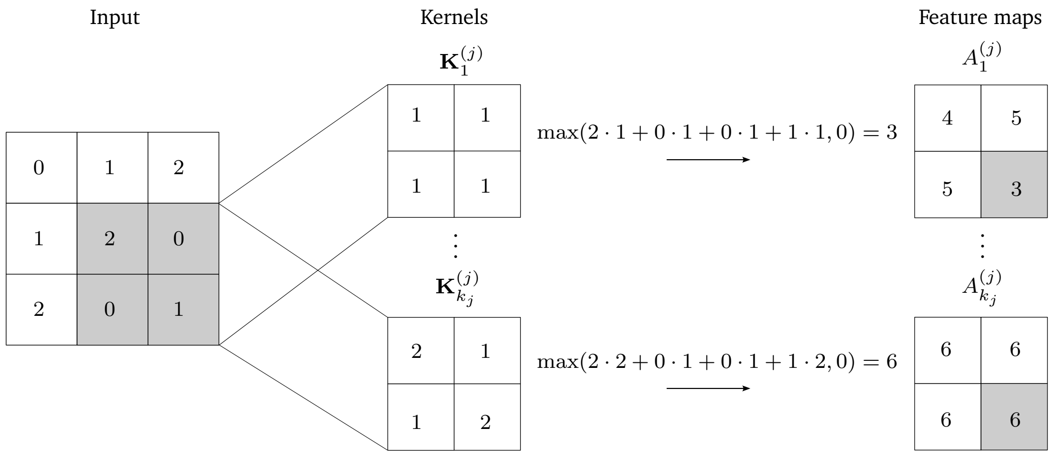


Figure 2.7.: Cross-correlation operation in a convolutional layer of a CNN. Note that the respective bias terms $w_{0,1}^{(j)}, \cdots, w_{0,k_j}^{(j)}$ are assumed to be zero, own illustration adapted from [Zhan20].

Stride and padding

According to Equation 2.18, computing all units in a feature map effectively causes a kernel to slide across the input starting from the upper-left corner and ending in the lower-right corner. The magnitude of this slide, i.e., the distance between the receptive field centers of neighboring units in both horizontal and vertical directions, is known as stride [Kriz12]. The stride in horizontal and vertical directions typically defaults to one. However, in certain situations, the stride can be chosen larger, e.g., to accelerate computation or reduce the output size [Zhan20].
Looking at Figure 2.7, it can be observed that in order to compute the cross-correlation, the respective kernel needs to fit completely inside the input. Satisfying this condition generally leads to a reduction of the output size compared to the input size [Good16]. The reduction in output size is accompanied by a loss of information on the perimeter of the image [Zhan20]. An effective way to prevent this information loss is to apply a mechanism known as zero padding, i.e., adding extra zero units to the boundaries of the input. See Appendix A.1.4 for further details.
Both stride and padding represent important hyperparameters of a convolutional layer that, combined with the input and kernel size, influence the output size of the cross-correlation operation according to Equation 2.19 [Chol17; Gero19; Zhan20]

$$\underbrace{\lfloor (h_1 - d_1 + p_v + s_v)/s_v \rfloor}_{\text{output height}} \times \underbrace{\lfloor (h_2 - d_2 + p_h + s_h)/s_h \rfloor}_{\text{output width}}, \tag{2.19}$$

where h_1, h_2 and d_1, d_2 denote the height and width of the input and kernel. s_h, s_v and p_h, p_v correspond to the horizontal and vertical extents of stride and padding, respectively.

Pooling layer

Pooling layers resemble the complex cells in the model of [Hube62]. The purpose of this layer type is to preserve relevant information while discarding unnecessary details to obtain invariance to shifts in positions, lighting, etc. [Bour10b]. Specifically, by applying a pooling operation to the output of a convolutional layer, the resulting feature maps are less sensitive to the exact location of structures, which allows subsequent convolutional layers to extract features that become increasingly invariant to local transformations [Zeil13; Gu18].

A pooling layer uses a single fixed-size kernel that is applied to each feature map individually [Lecu10]. However, instead of computing the dot product of the input units and a set of learnable weights, the kernel applies the respective pooling operation to the input units of each local receptive field. Typical pooling operations include, e.g., average and max-pooling [Bour10a]. For the purpose of this thesis, the emphasis will be placed on max-pooling as it has been demonstrated to consistently outperform average-pooling operations [Bour10a; Bour10b].
Max-pooling simply selects the maximum unit of the respective receptive field. Hence, independent of where the maximum element in a receptive field appears, the output of the pooling operation will be identical [Ranz07]. This demonstrates the decreasing sensitivity to the precise location of structures that is obtained via pooling. Note that in average-pooling, the $\max$ operation is replaced by computing the mean of the input units in the receptive field.
The stride of the pooling kernel is traditionally chosen to be larger than one in order to achieve the desired downsampling effect [Lecu10]. Specifically, choosing a kernel size of 2×2 with $s_h = s_v = 2$, effectively halves the input size, which is the typical default setting in, e.g., TensorFlow [Abad15]. See Appendix A.1.5 for a visualization of the max-pooling operation.

LeNet

As mentioned before, the LeNet represents the origin of contemporary CNNs. It gained popularity due to its outstanding performance on the Modified National Institute of Standards and Technology (MNIST) database, which contains 28×28 grayscale images of handwritten digits [Lecu89; Lecu98a]. The LeNet combines the three basic components of a CNN for classification, i.e., convolutional layers, pooling layers and a classification module in a single architecture.
In its original form, the LeNet contains two convolutional modules for feature extraction that each consist of a convolutional layer with sigmoid activation and a subsequent average-pooling layer. The convolutional layers in the first and second module use six and sixteen kernels, respectively, with a receptive field size of 5×5. The two average-pooling layers use a kernel of size 2×2 and strides $s_h = s_v = 2$, thus halving the size of the feature maps after their respective convolutional layers. The feature extractor is also referred to as the convolutional encoder, since it encodes the high-dimensional input into a lower-dimensional representation [Zhan20].
After the convolutional encoder, the respective feature maps are flattened into a single vector. This vector is then passed to the classification module which is by itself an encoder that is represented by a three-layer MLP with sigmoid activations having 120 input units, 84 hidden units, and 10 output units, respectively. The output layer uses a softmax activation to obtain probabilities for each of the ten MNIST classes. Figure 2.8 provides an illustration of the LeNet. Note that due to its specific architecture, the LeNet can be trained via backpropagation in the same way as a conventional MLP.

Visually interpreting CNN decisions

While CNNs achieve remarkable performance, they generally lack a built-in interpretability scheme that provides insights into the internal decision-making process [Zeil14; Mahe16]. However, being able to interpret the decisions of a model plays a vital role in practical applications, especially in critical areas such as medicine or financial markets [Lipt16]. Thus, in recent years, researchers have developed a variety of approaches that enable the interpretation of CNN decisions to unfold their full potential in practical applications.
A simple and effective technique for the interpretation of CNN-based model decisions is Gradient-weighted Class Activation Mapping (Grad-CAM), which provides a generalization of the conventional

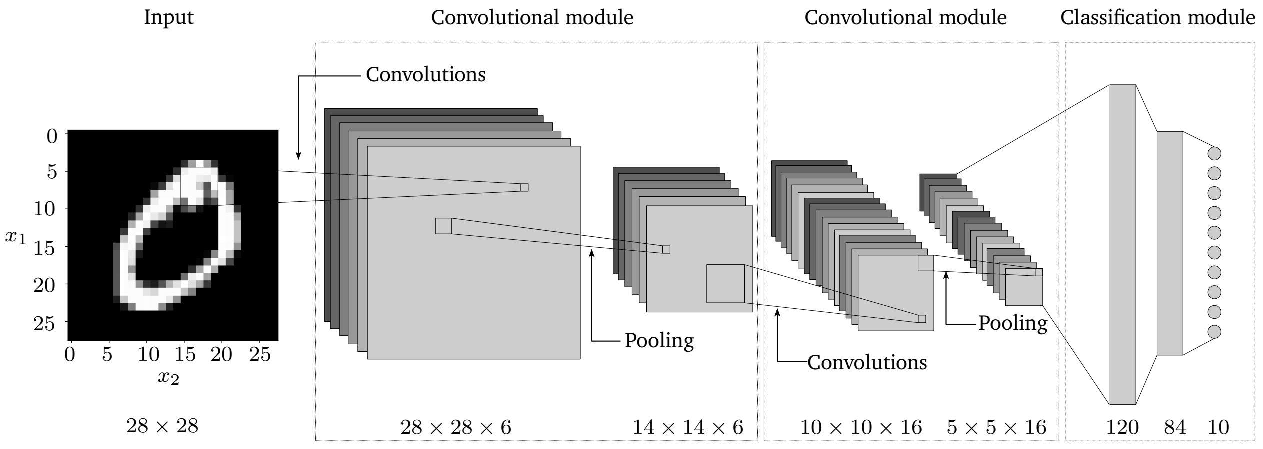


Figure 2.8.: LeNet architecture for classifying handwritten digits, own illustration adapted from [Lecu89].

Class Activation Mapping (CAM) approach presented by [Zhou16; Selv19]. The idea that underlies Grad-CAM is to compute importance weights for a particular network decision for each feature map in the last convolutional layer using gradient information [Selv19].
Consider the last convolutional layer l of a trained CNN for classification with feature maps $A_1^{(l)}, \cdots, A_{k_l}^{(l)}$, where $A_i^{(l)} \in \mathbb{R}^{n_1 \times n_2}$. The importance weight ζ_i of feature map $A_i^{(l)}$ in Grad-CAM for a given class y_j can be obtained by (1) computing the gradients of y_j prior to the softmax activation with respect to the individual units of $A_i^{(l)}$ and (2) applying an average-pooling operation on these gradients, according to Equation 2.20

$$\zeta_i = \frac{1}{n_1 n_2} \sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \frac{\partial y_j}{\partial a_{u,v,i}^{(l)}} \; \forall i \in \{1, \cdots, k_l\}. \tag{2.20}$$

Having obtained the importance weights $\zeta_1, \cdots, \zeta_{k_l}$, the next step involves computing the linear combination of importance weights and feature maps and passing the result to a ReLU activation, as shown in Equation 2.21. This results in a single matrix $G \in \mathbb{R}^{n_1 \times n_2}$, known as the Grad-CAM heatmap, that contains the visual explanations for the CNN

$$G = \mathrm{ReLU}\left(\sum_{i=1}^{k_l} \zeta_i A_i^{(l)}\right). \tag{2.21}$$

Resizing G to the original input size allows for highlighting the regions in the input that contributed to the CNN decision. Figure 2.9 depicts the visual explanations obtained via Grad-CAM for a conventional LeNet that has been trained on the MNIST dataset.

Continuous wavelet transformation

The wavelet transformation is an important tool for time-frequency analysis in the field of signal processing. Within the scope of this thesis, the presented content regarding wavelets is restricted to CWT and the essential theoretical aspects that are required for later chapters. Readers interested in diving deeper into the domain of wavelets are encouraged to consult the respective standard literature such as [Mall09].

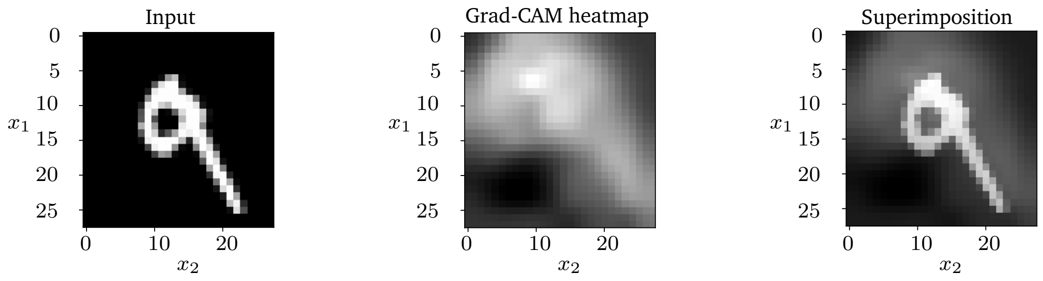


Figure 2.9.: Grad-CAM for visual interpretation of CNN decisions, own illustration.

Definition 2.1.14 (Signal). *"The term signal refers to a physical quantity that carries certain types of information and serves as a means for communication" [Gao11].*

Definition 2.1.15 (Signal processing). *Signal processing aims to extract relevant information from a signal by applying some sort of transformation [Riou91].*

The general idea of the wavelet transformation is to compute the dot product between a signal and a family of so-called wavelets which represent scaled and shifted versions of a prototype or mother wavelet [Riou92]. A wavelet is a square integrable function $\psi(t)$, i.e., $\int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty$ that satisfies the admissibility condition, which is displayed in Equation 2.22 [Mall09]

$$\int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty, \tag{2.22}$$

where $\Psi(\omega)$ corresponds to the Fourier transform of $\psi(t)$. Satisfying the admissibility condition in Equation 2.22 implies that $\Psi(\omega = 0) = 0$ and $\int_{-\infty}^{\infty} \psi(t) dt = 0$ [Mall09].
There are a vast number of mother wavelet types for different purposes, depending on the signal at hand. Figure 2.10 displays three frequently used mother wavelet types in the context of CWT.

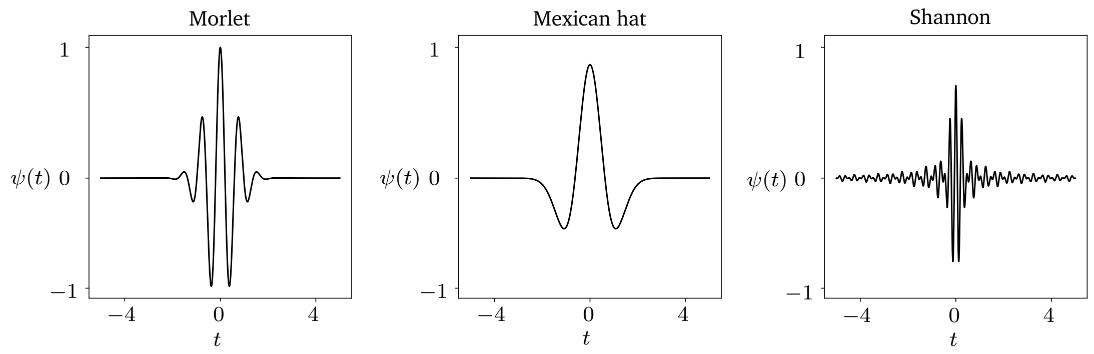


Figure 2.10.: Common wavelet types used in CWT, own illustration.

For the purpose of this thesis, the Morlet wavelet [Gros84] is of major relevance as it can identify transient components in a signal, such as bearing defect-induced vibrations [Gao11]. The Morlet wavelet is given by Equation 2.23 [Sinh05]

$$\psi(t) = \frac{1}{\sqrt[4]{\pi}} e^{i\omega_0 t} e^{\frac{-t^2}{2}}, \tag{2.23}$$

where $i = \sqrt{-1}$ is the imaginary number and ω_0 corresponds to the frequency.
CWT represents a specific form of wavelet transformation, which is defined by Equation 2.24 [Goup84; Mall09]

$$\mathrm{CWT}(f(t);\iota,\tau) = \frac{1}{\sqrt{\iota}} \int_{-\infty}^{\infty} f(t)\psi(t)^* \left(\frac{t-\tau}{\iota}\right) dt \; \forall \iota \in \{1,\cdots,v\}, \tau \in \{1,\cdots,n\}. \tag{2.24}$$

Here, $f(t)$ is the respective signal in the form of a univariate time series with n time steps, $\iota > 0$ represents the scaling parameter, τ is the shift parameter, and $\psi(t)^*$ corresponds to the complex conjugate of the selected mother wavelet. Note that the total number of scales v is a hyperparameter and must be chosen beforehand.

Definition 2.1.16 (Time series)**.** *A time series represents a collection of subsequent observations over time, such as the time-varying signal of an accelerometer [Hami94; Chat95].*

Definition 2.1.17 (Univariate time series)**.** *A univariate time series of length n is a time series in which a single feature varies over n time steps, i.e., $f(t) \in \mathbb{R} \; \forall t \in \{1,\cdots,n\}$ [Chal19; Bláz21].*

Definition 2.1.18 (Multivariate time series)**.** *A multivariate time series of length n is a time series in which m features vary over n time steps, i.e., $f(t) \in \mathbb{R}^m \; \forall t \in \{1,\cdots,n\}$ [Chal19; Bláz21].*

Evaluating the integral in Equation 2.24 for each pair (ι,τ) transforms the one-dimensional signal $f(t)$ from the time domain to a two-dimensional time-scale representation $\mathrm{CWT}(f(t)) \in \mathbb{R}^{v\times n}$, also known as a scalogram [Sinh05; Gao11]. The entries (ι,τ) of the scalogram are referred to as wavelet coefficients and express the similarity between the scaled wavelet and the corresponding section of the time series [Riou91].
Conventional methods for time-frequency analysis such as Short-Time Fourier Transformation (STFT) rely on a predefined sliding window length, which results in a fixed time-frequency resolution [Sinh05]. In contrast, the CWT enables variable window lengths with the help of the scaling parameter ι, which squeezes or stretches the mother wavelet, leading to an arbitrarily good time resolution at high frequencies and an arbitrarily good frequency resolution at low frequencies [Riou91]. This feature makes the CWT especially useful for analyzing non-stationary signals, i.e., signals whose statistical properties change over time, like the ones found in manufacturing processes [Riou91; Mall09; Gao11]. Figure 2.11 depicts the transformation of a raw accelerometer signal to a scalogram representation using the CWT with the Morlet wavelet.

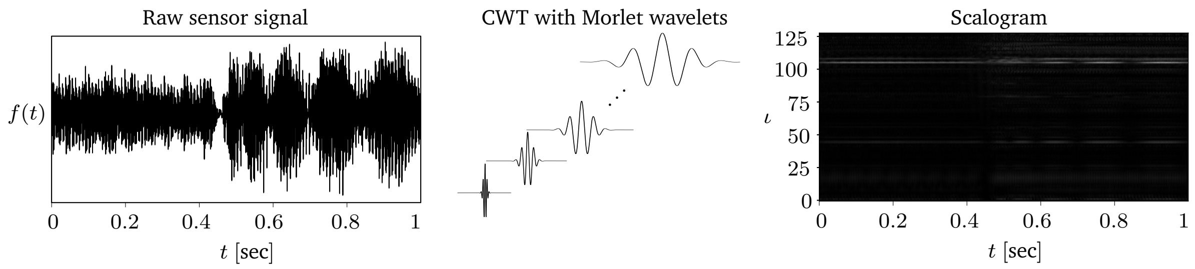


Figure 2.11.: Scalogram of an accelerometer signal. The CWT is applied to the raw sensor signal using the Morlet mother wavelet with $v = 128$ scales. Each row in the scalogram contains the absolute values of the wavelet coefficients obtained via the CWT by shifting the respective scaled wavelets across the signal, own illustration.

From an ML perspective, the scalogram of a time series can be interpreted as a single-channel image that can be processed by modern CNN architectures. As such, CWT represents an effective preprocessing step for CNN applications in manufacturing that involve non-stationary signals in the form of a time series, see, e.g., [Hübn20; Tran20a; Tran20b; Liao21].

2.1.4. Self-supervised learning

Contemporary DL approaches require excessive amounts of annotated, i.e., labelled, training data in order to reach their remarkable performance levels [Wu18]. However, in many practical situations, retrieving large amounts of labelled data is not feasible from an economic viewpoint [Baev20; Bhun21]. In order to confront this dilemma, researchers in DL investigated novel approaches that can learn effective representations from unlabelled data [Ermo21]. These efforts resulted in the emerging field of SSL.

Definition 2.1.19 (Self-supervised learning). *SSL is a form of UL that aims at learning effective representations from unlabelled data for real-world downstream tasks, such as AD, by solving a supervised pretext task using pseudo labels that are generated automatically from unlabelled data [Kole19; Alay20; Jing20; Zbon21].*

SSL has recently experienced an enormous surge in popularity due to its promising performance in learning high-quality features that match or surpass fully SL approaches [Chen21b; Bard22]. Computer vision and natural language processing represent the research fields that are by far the most active with respect to the application of SSL to learn effective representations [Chen21e]. Figure 2.12 illustrates the conceptual idea of SSL.

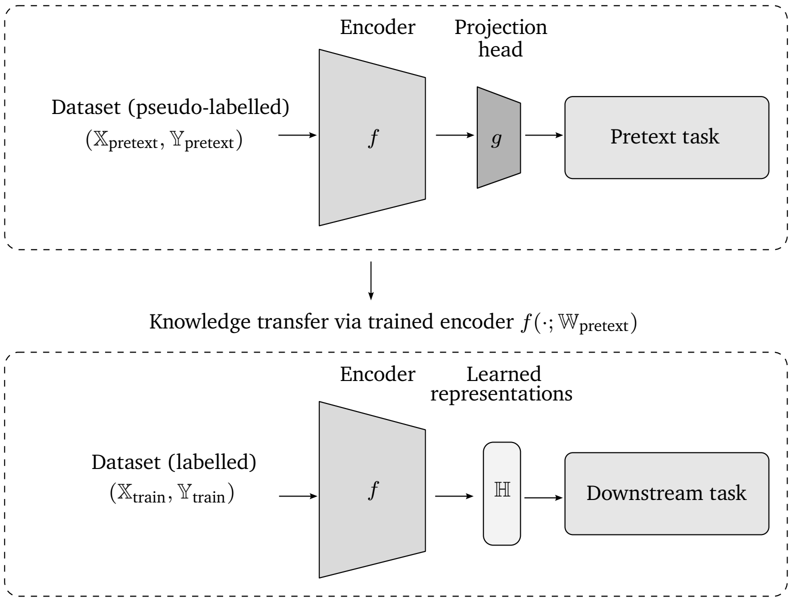

Encoder
Projection head
Dataset (pseudo-labelled)
$(\mathbb{X}_{\text{pretext}}, \mathbb{Y}_{\text{pretext}})$
f
g
Pretext task
Knowledge transfer via trained encoder $f(\cdot; \mathbb{W}_{\text{pretext}})$
Encoder
Learned representations
Dataset (labelled)
$(\mathbb{X}_{\text{train}}, \mathbb{Y}_{\text{train}})$
f
$\mathbb{H}$
Downstream task

Figure 2.12.: Conceptual idea of SSL. Top: In the first step, a pretext task is solved by training a model on an unlabelled dataset $\mathbb{X}_{\text{pretext}}$ with a set of pseudo labels $\mathbb{Y}_{\text{pretext}}$ that are automatically generated from the data and specific to the pretext task. Bottom: In the second step, the real-world downstream task is solved using the learned representations from the pretext task, i.e., $\mathbb{H} = f(\mathbb{X}_{\text{train}}; \mathbb{W}_{\text{pretext}})$, own illustration adapted from [Noro18; Jing20].

First, a self-defined pretext task is solved using a set of unlabelled pretext data $\mathbb{X}_{\text{pretext}}$, typically based on the training data $\mathbb{X}_{\text{train}}$, and a set of autogenerated pseudo labels $\mathbb{Y}_{\text{pretext}}$. For this purpose,

an encoder network f, often a CNN, maps the data to a lower dimension. A so-called projection head g, which is usually a simple MLP with a softmax output, is constructed on top of the encoded data. Both f and g are trained end-to-end in a supervised manner on the pretext task via backpropagation. The trained encoder f is consequently transferred to the subsequent downstream task, whereas the projection head is usually discarded, since empirical evidence suggests that the learned representations from f tend to be more effective [Chen20; Caro21; Ermo21]. Here, the learned representations $\mathbb{H} = f(\mathbb{X}_{\text{train}}; \mathbb{W}_{\text{pretext}})$ are extracted. These representations are consequently used, in conjunction with the available labels $\mathbb{Y}_{\text{train}}$, to solve the downstream task. Note that the description above introduces the data $\mathbb{X}_{\text{pretext}}, \mathbb{X}_{\text{train}}$ as well as the labels $\mathbb{Y}_{\text{pretext}}, \mathbb{Y}_{\text{train}}$ and the learned representations $\mathbb{H}$ with the set notation. This is because the actual dimensionality of the data depends on the problem and is irrelevant for understanding the conceptual idea.
According to [Chen20], SSL approaches can be broadly subdivided into (1) discriminative and (2) generative approaches, see Figure 2.13. Generative approaches solve a pretext task that builds a distribution over the input data and encoding space to model, e.g., image pixels, from the learned representations [Doer15; Chen20; Gril20]. These approaches typically rely on Variational Autoencoders (VAEs) or Generative Adversarial Networks (GANs) [Good14; King14; Jing20]. Discriminative approaches are more similar to the conventional SL setting and can be partitioned into handcrafted pretext tasks and pretext tasks that are based on contrastive learning in the encoding space [Chen20; Gril20].

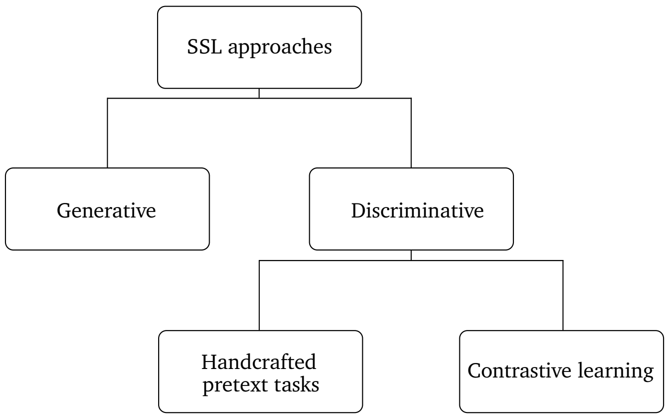


Figure 2.13.: Classification of SSL approaches, own illustration adapted from [Chen20; Gril20].

The subsequent, more detailed discussion will focus on discriminative approaches to SSL. The rationale for this is twofold. First, generative approaches play a negligible role in the remainder of this thesis and would thus disrupt the line of thought. Second, discriminative SSL methods form an integral part of contemporary approaches to self-supervised AD, see Section 3.1. The interested reader who wishes to gain more insights into generative approaches is encouraged to investigate, e.g., [Gril20], as they cite some common articles in this line of SSL research.

Handcrafted pretext tasks

Learning representations from handcrafted pretext tasks represents one of the earliest and most common forms of discriminative SSL approaches [Bukc21; Yue21]. In recent years, an excessive number of specific pretext tasks have been designed [Doer17], such as relative patch prediction [Doer15], image in-painting [Path16], solving jigsaw puzzles [Noro16], image colorization [Zhan16], or predicting image rotations [Gida18].

In the following, a short overview of three fundamental approaches in this stream of SSL research will be presented to shape a better understanding and gain intuition of how such pretext tasks are used to learn effective representations. Specifically, the ideas of relative patch prediction, solving jigsaw puzzles, and predicting image rotations are presented. The reason for choosing exactly these approaches is based on the fact that they are frequently cited as being representative of handcrafted pretext task methods, see, e.g., [Alwa20; Akba21; Wang21; Yang21; Zhen21]. Following a chronological order, the first pretext task that will be presented is the relative patch prediction task, see Figure 2.14.

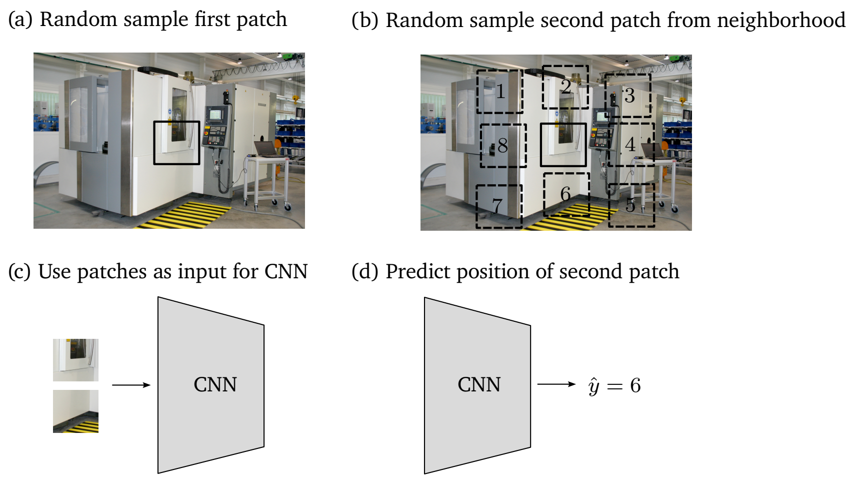

(a) Random sample first patch
(b) Random sample second patch from neighborhood
1
2
3
8
4
7
6
5
(c) Use patches as input for CNN
CNN
(d) Predict position of second patch
CNN
$\hat{y} = 6$

Figure 2.14.: Relative patch prediction pretext task, own illustration adapted from [Doer15].

In this pretext task, random pairs of patches from a large set of unlabelled images are extracted. A CNN is trained to predict the position of the second patch relative to the first patch of an image. More precisely, the second patch represents one of eight possible neighbors of the first patch and the CNN needs to specify which neighbor was sampled. The intuition here is that in order to do well on this task a model needs to be able to recognize objects and their corresponding structure in the image [Doer15].
The second pretext task is solving jigsaw puzzles [Noro16], see Figure 2.15.

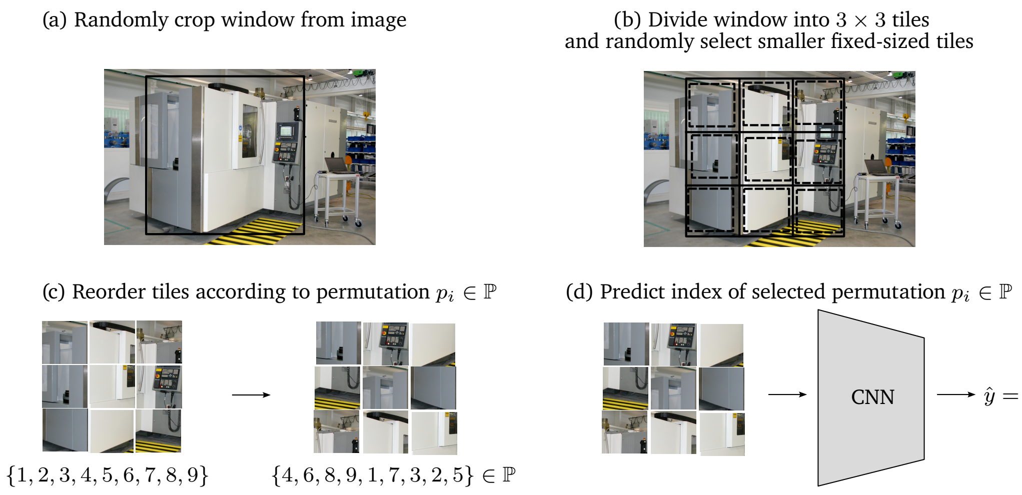

(a) Randomly crop window from image
(b) Divide window into 3×3 tiles and randomly select smaller fixed-sized tiles
(c) Reorder tiles according to permutation $p_i \in \mathbb{P}$
$\{1,2,3,4,5,6,7,8,9\}$
$\{4,6,8,9,1,7,3,2,5\} \in \mathbb{P}$
(d) Predict index of selected permutation $p_i \in \mathbb{P}$
CNN
$\hat{y} = i$

Figure 2.15.: Solving jigsaw puzzle pretext task, own illustration adapted from [Noro16].

Let $\mathbb{S}_9$ be the set of all permutations of the set $\{1, 2, 3, \cdots, 9\}$, and $\mathbb{P} \subset \mathbb{S}_9$ be a set of preselected permutations of $\mathbb{S}_9$. The idea of this pretext task is to train a CNN to correctly classify the index i of permutation $p_i \in \mathbb{P}$ that has been applied to a set of 9 cropped image tiles.
The last pretext task corresponds to predicting image rotations based on the so-called Rotation Network (RotNet) [Gida18]. As the name suggests, the idea of this setting is to train a CNN to predict the rotation $r \in \{0°, 90°, 180°, 270°\}$ that has been applied to an image. Prior to training, every image is transformed with every rotation. Thus, the dataset size is increased by a factor of four. The intuition here is that a model needs to have some understanding of the objects depicted in a given image, such as location, type, and pose, in order to perform well on this pretext task [Gida18]. Perhaps surprisingly, the idea of rotation prediction led, despite its simplicity, to a dramatic improvement in the quality of learned representations. See Figure 2.16 for an illustration of the concept.

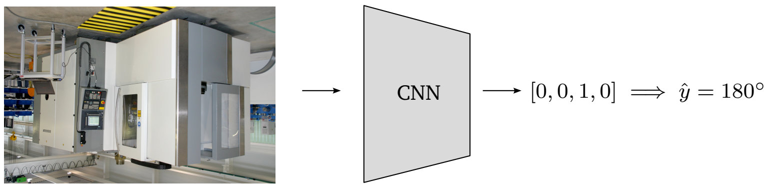


Figure 2.16.: Rotation prediction pretext task, own illustration adapted from [Gida18].

Contrastive learning

Handcrafted pretext tasks rely on domain expertise to enable the extraction of effective representations for the subsequent downstream task [Caro18]. For certain practical applications, especially those that require a lot of domain knowledge, this form of SSL is very useful to learn domain-specific representations. However, some researchers argue that these handcrafted pretext tasks are somewhat arbitrary and bear the risk of impeding the learning of general-purpose features [Chen20].
Contrastive learning represents a recently emerging and very promising alternative to handcrafted pretext tasks in the domain of discriminative SSL [Chen21c]. Dating back to the works of [Beck92; Hads06], the idea that underlies contrastive learning is to learn representations with the help of a contrastive loss function that pulls together similar (positive) data pairs and pushes away dissimilar (negative) data pairs [Oord18; Wu18; Bach19; Chen20; Tian20; Sord21]. Specifically, the objective is to maximize the similarity between multiple views, i.e., augmentations, of the same data, to learn representations that are view-invariant [Alay20; Tian20; Wei21]. In the case of image data, which are most relevant for this thesis, these augmentations are typically based on simple transformations, e.g., rotations or random color distortions [Chen20; Dwib21]. Deciding which views will result in the best representations depends on the corresponding downstream task and thus requires domain expertise as well [Tian20].
In the following, the Simple Framework for Contrastive Learning of Visual Representations (SimCLR) by [Chen20] will be presented. Since its publication in 2020, SimCLR has rapidly developed into one of the most important approaches to contrastive learning [Chen21d; Dwib21; Fan21; Jian21]. The reason for presenting SimCLR in this thesis is twofold. First, due to its general relevance for SSL. Second, many approaches that use contrastive learning for self-supervised AD rely on this framework, see Section 3.1.
SimCLR consists of four main components [Chen20]: (1) An augmentation module that randomly samples two augmentation functions $\mathcal{A}$ and $\mathcal{A}'$ from a set of augmentation functions $\mathbb{A}$ to transform a given image tensor $\mathbf{X}$ into two correlated views, $\tilde{\mathbf{X}}_i$ and $\tilde{\mathbf{X}}_j$, i.e., the positive pair. Note that the

augmentations used in SimCLR are random cropping, including resizing and flipping, color distortion and Gaussian blurring. (2) An encoder network f, which is represented by a CNN that maps the augmented data to a lower-dimensional representation $\mathbf{h} \in \mathbb{R}^d$, i.e., $\mathbf{h}_i = f(\tilde{\mathbf{X}}_i)$ and $\mathbf{h}_j = f(\tilde{\mathbf{X}}_j)$. (3) A simple MLP, g, that acts as a projection head and transforms the encoded data $\mathbf{h}_i, \mathbf{h}_j$ to the space in which the contrastive loss function is applied, $\mathbf{z}_i = g(\mathbf{h}_i), \mathbf{z}_j = g(\mathbf{h}_j)$. (4) The $\mathcal{L}_{\text{NT-Xent}}$ loss function that pulls $\mathbf{z}_i, \mathbf{z}_j$ closer together while simultaneously pushing all negative examples away. The loss function that is used in SimCLR and in most recent works on contrastive learning is based on CE [Sohn16; Oord18; Wu18; Chen21c] and referred to as Normalized Temperature-Scaled Cross Entropy Loss (NT-Xent) [Chen20]. Figure 2.17 provides an overview of the individual components of SimCLR.

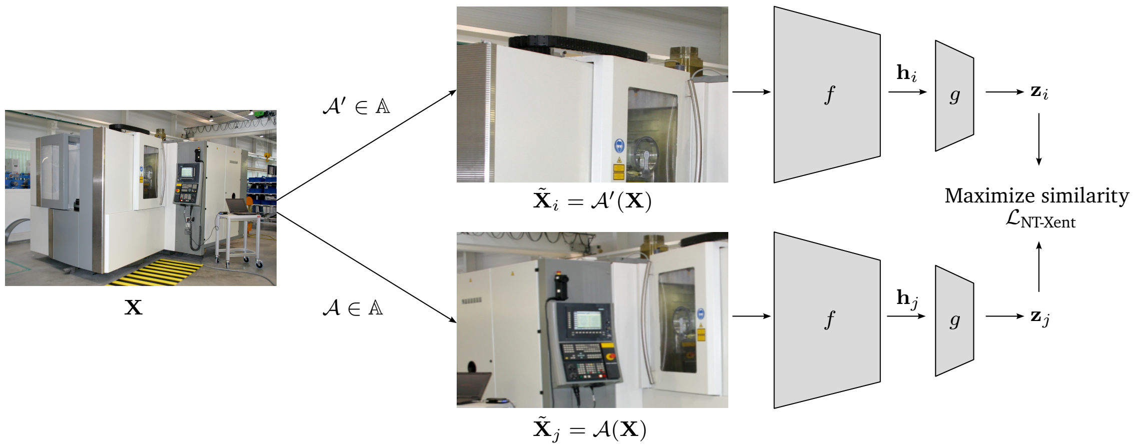

$\mathcal{A}' \in \mathbb{A}$
$\mathcal{A} \in \mathbb{A}$
$\mathbf{X}$
$\tilde{\mathbf{X}}_i = \mathcal{A}'(\mathbf{X})$
$\tilde{\mathbf{X}}_j = \mathcal{A}(\mathbf{X})$
f
$\mathbf{h}_i$
$\mathbf{h}_j$
g
$\mathbf{z}_i$
$\mathbf{z}_j$
Maximize similarity
$\mathcal{L}_{\text{NT-Xent}}$

Figure 2.17.: SimCLR overview, own illustration adapted from [Chen20; Jian20].

It is worth mentioning that during training all n data points, i.e., images in a batch, are transformed into mutually exclusive positive pairs using the randomly sampled augmentation functions $\mathcal{A}$ and $\mathcal{A}'$. This results in a total of $2n$ images, or n positive pairs. For each positive pair that is encoded via $g \circ f$, the remaining pairs are considered negative examples [Chen20].
SSL marks the last relevant component in this thesis with respect to the fundamentals of ML. With the acquired knowledge, the focus will now be shifted to the fundamentals of AD.

2.2. Anomaly detection

In this section, the required fundamentals regarding AD are presented. The section is divided into two main parts. The first part focuses on providing a solid overview with respect to the underlying core components of AD. The second part focuses on the introduction of the most relevant shallow and deep AD algorithm baselines that are going to be used to benchmark the performance of SSMSPC in Chapter 6.

2.2.1. Basic concepts and terminologies

AD is an area of research that stretches across various disciplines, such as engineering or statistics and has been studied at least since the 19th century [Ruff21b]. In fact, the work of [Edge87] dating back to 1887, marks the first formal appearance of AD in the literature [Ruff21b]. AD has established itself as a major area of interest among ML researchers due to its relevance for numerous practical applications, such as credit card fraud detection, intrusion detection, or fault detection in manufacturing processes

[Gu19; Bláz21; Ye21]. All these application scenarios share the common objective of identifying data that deviate from a known state of normality [Agga17].

Definition 2.2.1 (Anomaly). *An anomaly or outlier is a data point that substantially differs from the expected normal behavior of the data [Chan09; Pang21].*

Definition 2.2.2 (Anomaly detection). *AD, also known as outlier detection, refers to the process of identifying anomalies in data [Chan09; Chal19; Pang21].*

AD presents, among others, two unique challenges given its inherent focus on rare and generally uncertain events [Pang21]. First, the potential number, extent, and characteristics of anomalies are unknown until they actually occur [Deec21]. Second, AD is subject to an inherent class imbalance between normal and anomalous samples, i.e., there are much more normal than anomalous data [Agga17; Chal19; Hou21; Wang21; Ye21]. More precisely, the ratio of anomalies to normal samples in practice is often $< 1\%$ [Rebj21]. This makes the application of conventional SL approaches difficult, as it is typically unfeasible to collect sufficient anomalous data to train a model for binary classification [Pang21; Xu22].

Anomaly types

Researchers in AD broadly distinguish three types of anomalies: (1) point anomalies, (2) collective anomalies, and (3) contextual anomalies [Chan09; Gupt14; Agga17; Chal19].

Definition 2.2.3 (Point anomaly). *A point anomaly is characterized as a single data instance that is anomalous compared to the remaining data instances [Chan09].*

Detecting point anomalies accounts for the majority of the existing research work in AD [Chal19; Bláz21; Ruff21b].

Definition 2.2.4 (Collective anomaly). *A collective anomaly or group anomaly encompasses a collection of data instances that, considered as a whole, are anomalous with respect to the remaining data [Chan09; Pang21].*

Definition 2.2.5 (Contextual anomaly). *A contextual anomaly, also known as a conditional anomaly corresponds to a data instance that might be normal when considered solely but is anomalous with respect to the specific context in which it appears [Song07; Chan09; Ruff21b].*

Contextual anomalies are most commonly investigated in terms of time series data [Chan09; Gupt14]. Here, the so-called contextual attribute is time, which determines the position of an instance in the sequence [Chan09; Chal19]. Note that point and collective anomalies can also be contextual, e.g., in terms of anomalous instances or subsequences in time series data [Chan09; Ruff21b]. For the purpose of this thesis, contextual point and contextual collective anomalies are treated as general contextual anomalies. Regarding the proposed SSMSPC approach, contextual anomalies are of major relevance, as it is designed to detect and localize anomalies in sensor signals that take the form of a multivariate time series. Figure 2.18 provides a visualization for each of the three anomaly types.

Problem settings

Depending on the availability of labels during training, AD can be classified into three different problem settings: (1) supervised, (2) semi-supervised, and (3) unsupervised AD [Chan09].

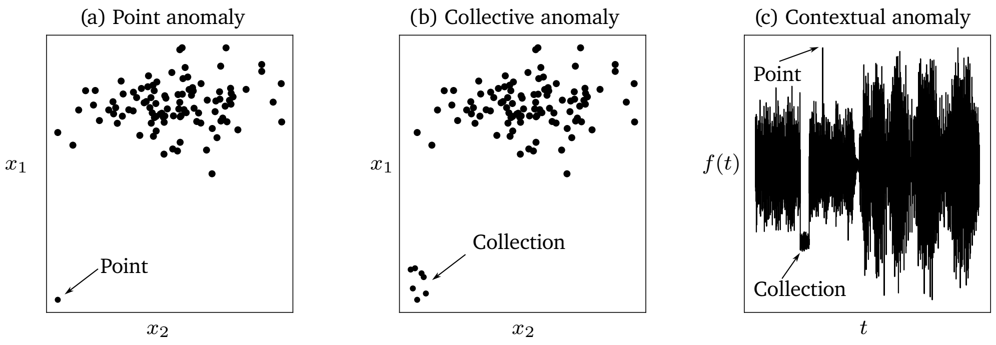


Figure 2.18.: Three types of anomalies. (a) A point anomaly corresponds to a single anomalous instance. (b) A collective anomaly represents a coherent group of anomalous instances. (c) Contextual anomalies represent both anomalous instances and coherent anomalous groups with respect to some context, such as time, own illustration adapted from [Ruff21b].

Definition 2.2.6 (Supervised AD). *Supervised AD describes a setting in which both labelled normal and anomalous instances are available for training [Chan09; Chal19].*

The supervised AD setting is equivalent to the SL setting in general ML. Due to the inherent rarity of anomalies and the corresponding difficulty of acquiring labelled anomalous instances for training, this setting is of minor interest in AD research [Chal19; Pang21].

Definition 2.2.7 (Semi-supervised AD). *Semi-supervised AD describes a setting in which only normal instances are available for training [Chan09; Chal19].*

Semi-supervised AD is arguably the most commonly studied setting in AD and is also referred to as one-class classification [Chal19; Nguy19; Berg20a; Li21a]. The rationale for this popularity is that, in practice, normal data instances are easier to collect than anomalies [Kuma19]. Methods in the field of semi-supervised AD learn to model the normal behavior of the data and flag instances that strongly deviate from this behavior as anomalies [Chan09; Agga17; Good20].

Definition 2.2.8 (Unsupervised AD). *Unsupervised AD describes a setting in which the available training data are (1) unlabelled and (2) can contain both normal and anomalous instances [Chan09; Chal19].*

The unsupervised AD setting represents the most challenging but, at the same time, most general problem setting in AD [Gu19]. Methods of unsupervised AD implicitly assume that most but not all training data correspond to the normal class [Goya20; Dai22].
It is worth mentioning that the terminology of semi-supervised and unsupervised AD is not consistently used among researchers in AD [Wang19; Ye21]. Some researchers, e.g., [Berg19; Zhan19a; Deha20; Ruff21b], refer to the semi-supervised AD setting as presented in Definition 2.2.7 as unsupervised AD. According to this terminology, semi-supervised AD is then typically framed as a setting where both labelled and unlabelled normal and anomalous data are available for training [Ruff21b].

Anomaly score and thresholding

The output of an AD model is either (1) a binary label or (2) a so-called anomaly score [Chan09]. An AD model that provides a binary label as output explicitly states whether the considered data point is to be flagged as normal ($y = 0$) or anomalous ($y = 1$) [Chan09]. This form of output is often

required to support the decision-making process in practical applications [Good16; Agga17]. Note that in this thesis, anomalous samples are assigned to the positive class, i.e., $y = 1$.
The second possible form of output corresponds to an anomaly score. Suppose an AD model is represented by a mapping $h : \mathbb{X} \to \mathbb{R}$. An anomaly score is a real number $h(\mathbf{x}) \in \mathbb{R}$ that quantifies the extent to which a given data point $\mathbf{x} \in \mathbb{X}$ is anomalous [Ruff21b]. Specifically, the higher the anomaly score, the more anomalous is the sample, given the underlying assumptions of the model. Conversely, a low anomaly score indicates that the considered sample is more consistent with the learned model of normality. Thus, an anomaly score allows to rank the data points under consideration, but it does not provide an explicit decision rule to separate anomalous from normal samples [Agga17; Deis20]. In order to assign the samples to the respective classes, a threshold mapping $b_\delta : \mathbb{R} \to \{0, 1\}$ can be derived, as shown in Equation 2.25

$$b_\delta \circ h(\mathbf{x}) = \begin{cases} 1, & h(\mathbf{x}) > \delta \\ 0, & \text{otherwise} \end{cases}, \tag{2.25}$$

where δ is the corresponding threshold. A common way to determine δ is to model the statistical distribution of the anomaly scores obtained from the training set [Agga17]. More precisely, the idea is to interpret the anomaly scores as realizations of a continuous random variable $\mathcal{X}_h$ and estimate the so-called Probability Density Function (PDF) $\hat{\theta}_h$ [Bish96].

Definition 2.2.9 (Random variable)**.** *A random variable expresses the outcome of a random experiment and is given by a mapping $\mathcal{X} : \mathbb{O} \to \mathbb{R}$, where $\mathbb{O}$ corresponds to the sample space, i.e., the set of possible outcomes of the random experiment [Grin97; Blit15; Deis20].*

Definition 2.2.10 (Continuous random variable)**.** *A continuous random variable is a random variable with a continuous sample space, i.e., $\mathbb{O} \subset \mathbb{R}$ [Grin97; Good16].*

Definition 2.2.11 (Probability density function)**.** *The PDF of a continuous random variable $\mathcal{X}$ is a mapping $\theta : \mathbb{R} \to \mathbb{R}$ that satisfies the following properties [Good16; Deis20]:*

1. $\theta(x) \geq 0 \; \forall x \in \mathbb{R}$
2. $\int_{-\infty}^{\infty} \theta(x) dx = 1$

The estimated PDF $\hat{\theta}_h$ of the anomaly scores $h(\mathbf{x})$ is then used to obtain the quantile function $\hat{\Theta}_h^{-1}$ by computing the inverse of the corresponding Cumulative Distribution Function (CDF) $\hat{\Theta}_h$ [Yan20; Lee21; Bieg22a; Bieg22b].

Definition 2.2.12 (Cumulative distribution function)**.** *The CDF of a continuous random variable $\mathcal{X}$ is a mapping $\Theta : \mathbb{R} \to [0; 1]$ that expresses the probability that $\mathcal{X} \leq \delta$ and is given by $\Theta(\delta) = \int_{-\infty}^{\delta} \theta(x) dx$, where $\theta(x)$ corresponds to the respective PDF of $\mathcal{X}$ [Blit15; Deis20; Zhan20].*

Definition 2.2.13 (Quantile)**.** *The ξ-quantile $\in [0; 1]$ of a random variable $\mathcal{X}$ defines the threshold δ such that $\Theta(\delta) = \xi$, i.e., the probability of $\mathcal{X} \leq \delta$ is ξ, while the probability of $\mathcal{X} > \delta$ is $1 - \xi$ [Take06].*

Definition 2.2.14 (Quantile function)**.** *The inverse of a CDF is known as the quantile function and represents a mapping $\Theta^{-1} : [0; 1] \to \mathbb{R}$ [Blit15].*

Evaluating the quantile function at a given ξ-quantile yields the corresponding threshold δ, i.e., $\Theta^{-1}(\xi) = \delta$. Figure 2.19 provides a schematic overview of the preceding statements based on a normally distributed random variable $\mathcal{X}$. See Section 2.3 for further details regarding the normal

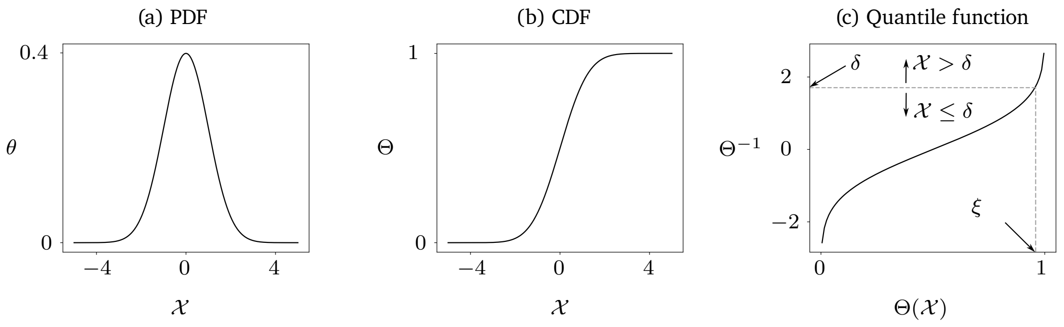


Figure 2.19.: Threshold selection scheme in AD. (a) Based on the PDF of a random variable $\mathcal{X}$, the corresponding (b) CDF is computed. (c) Inverting the CDF yields the quantile function that provides a threshold δ for every ξ-quantile, own illustration.

distribution. The ξ-quantile represents a hyperparameter that controls the resulting threshold δ. Its choice corresponds to a trade-off decision between false alarms (type I error) and missed anomalies (type II error). More precisely, if δ is set too low, the model flags more samples as anomalous, which increases the type I error, whereas if δ is set too high, the model misses anomalies, which in turn increases the type II error [Agga17]. Since the associated cost for type I errors and type II errors differ in severity depending on the individual application, this trade-off decision strongly affects the practical applicability of the resulting AD model [Ruff21b]. To find a suitable quantile, a common way is to specify the so-called significance level $\alpha \in [0; 1]$ that represents the desired false alarm rate and evaluate the quantile function at the $\xi = (1 - \alpha)$-quantile, i.e., $\Theta^{-1}(1 - \alpha) = \delta_\alpha$ [Yu11; Gras14; Magg14; Samu16; Yan20]. Alternatively, if there are no specific requirements in terms of α, the ξ-quantile can be chosen in such a way that the resulting threshold maximizes the corresponding f1-score on the validation set [Good20].
Methods that follow the procedure described above can be divided into (1) parametric and (2) nonparametric approaches [Chan09]. Parametric approaches assume a certain functional form of the PDF and fit the adjustable parameters, such as μ and σ in the case of a normal distribution, with the help of the available data [Bish96]. However, parametric approaches bear the risk that the assumed functional form is inappropriate to model the true underlying PDF [Bish06; Murp12]. Conversely, nonparametric approaches do not assume knowledge of the functional form of the PDF, but rather model the entire PDF based on the available data [Bish96].
Kernel Density Estimation (KDE) [Rose56; Parz62] arguably represents the most widely used nonparametric approach for density estimation [Samu16; Gero19; Ruff21b]. Given a set of n anomaly scores $h(\mathbf{x}_1), \cdots, h(\mathbf{x}_n)$, where $h(\mathbf{x}) \in \mathbb{R}\ \forall \mathbf{x} \in \mathbb{X}$, KDE estimates the value of the underlying PDF $\hat{\theta}_h$ for an anomaly score $h(\mathbf{x}_0)$ according to Equation 2.26 [Hast08; Agga17; Deis20]

$$\hat{\theta}_h(h(\mathbf{x}_0)) = \frac{1}{n\nu} \sum_{i=1}^{n} k\left(\frac{h(\mathbf{x}_0) - h(\mathbf{x}_i)}{\nu}\right), \tag{2.26}$$

where $k(\cdot)$ is a kernel function and $\nu > 0$ corresponds to the so-called bandwidth or smoothing parameter [Deis20]. A frequently employed kernel function is the Gaussian kernel, which is given by Equation 2.27 [Agga17; Kuhn19; Deis20]

$$k(u) = e^{-\frac{||u||^2}{2}}. \tag{2.27}$$

The bandwidth parameter ν is a hyperparameter that controls the width of the kernel [Murp12; Gero19]. Choosing ν too large leads to an excessive smoothing effect, i.e., the estimated PDF $\hat{\theta}$

underfits the data. Conversely, choosing ν too small results in a very noisy estimate, i.e., $\hat{\theta}$ overfits the data. Figure 2.20 visualizes the effects of varying ν on the estimated PDF $\hat{\theta}$ using KDE with a Gaussian kernel.

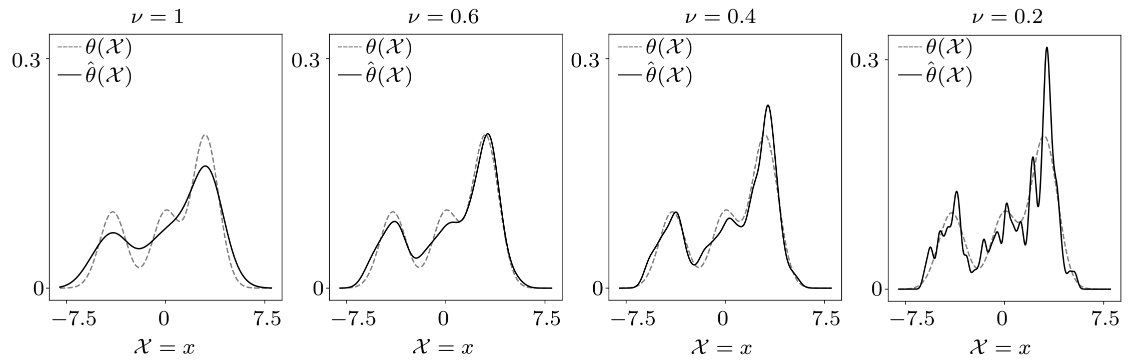


Figure 2.20.: KDE with a Gaussian kernel and varying bandwidth parameter ν. Larger values of ν lead to an increasing smoothing effect, while lower values result in a very noisy estimate $\hat{\theta}$ for the true underlying PDF θ, own illustration adapted from [Bish06].

Evaluation

The performance of an AD model can be evaluated based on (1) a specified threshold δ or (2) in terms of anomaly scores, i.e., threshold-independent [Agga17].
Evaluating the performance of an AD model with respect to a specific threshold is equivalent to evaluating a binary classification model with imbalanced data. There are generally two reasons for this. First, by setting a threshold, the resulting output of an AD model is a binary label. Second, by definition, there are more normal than anomalous samples in the data. Recall from Section 2.1.1 that the primary evaluation metrics for such a setting are precision, recall, and f1-score that measure the trade-off between type I error and type II error [Agga17]. Thus, the threshold-dependent evaluation scheme provides insights into the practical applicability of an AD model regarding its capability to distinguish normal from anomalous data instances for a specific threshold.
The drawback of the above scheme is that it does not provide a measure of the overall performance of the AD model [Brad97]. Specifically, it reflects the performance of a single problem-dependent threshold that represents one possible trade-off between type I error and type II error. However, in many situations, the associated costs for type I error and type II error require extensive domain knowledge or are even unknown upfront [Jame13; Ruff21b]. Thus, threshold-independent schemes that summarize the model performance across all possible trade-off decisions provide a more holistic picture [Hast08].
The standard threshold-indepedent evaluation measure for AD models is the Area Under Receiver Operating Characteristic (AUROC) [Abat19; Carr20; Ruff21b; Shen21; Wu21]. The Receiver Operating Characteristic (ROC) curve of a binary classifier represents a two-dimensional graph that plots recall over false alarm rate for every possible decision threshold in the test set [Fawc06; Ruff21b]. As the name implies, the AUROC corresponds to the area under this curve [Brad97; Fawc06]. It provides a threshold-independent evaluation measure that quantifies the overall performance of a model, accounting for all possible trade-off decisions. Due to this, the AUROC embodies a suitable metric to compare the performance of different models [Ding14]. Given an AD model $h : \mathbb{X} \to \mathbb{R}$, a set of normal instances $\mathbb{U} \subset \mathbb{X}$, and a set of anomalous instances $\mathbb{V} \subset \mathbb{X}$, the AUROC is defined by Equation

2.28, as the mean computed over all normal and anomalous data pairs [Camp16; Agga17]

$$\text{AUROC} := \underset{\mathbf{u}\in\mathbb{U},\mathbf{v}\in\mathbb{V}}{\text{mean}} \begin{cases} 1, & \text{if } h(\mathbf{v}) > h(\mathbf{u}) \\ \frac{1}{2}, & \text{if } h(\mathbf{v}) = h(\mathbf{u}) \\ 0, & \text{if } h(\mathbf{v}) < h(\mathbf{u}) \end{cases} \in [0;1]. \tag{2.28}$$

Equation 2.28 shows that the AUROC ranges from 0 to 1 (or 0% to 100%), where AUROC $= 1$ indicates a perfect classifier. Thus, for a given model, the AUROC can be interpreted as the probability of ranking a randomly chosen anomalous sample higher than a randomly chosen normal sample [Hanl82; Fawc06].

One of the main reasons for the popularity of the AUROC is that for a random classifier, i.e., a model that assigns the normal and anomalous classes randomly, the AUROC is always 0.5 regardless of any imbalances between normal and anomalous instances in the test set [Ruff21b]. In this case, the ROC corresponds to the main diagonal [Fawc06]. Thus, if the ROC lies above the main diagonal, the corresponding classifier exploits some information in the data that are useful for discrimination [Fawc06]. Conversely, if the ROC lies below the main diagonal, the corresponding classifier achieves worse than random performance [Flac05; Fawc06]. More precisely, since inverting the decision in such a scenario leads to a point above the main diagonal, it can be said that the classifier exploits useful information for discrimination but applies this information wrongly [Flac05]. Figure 2.21 provides an illustration of several ROC curves and the corresponding AUROC for different classifiers.

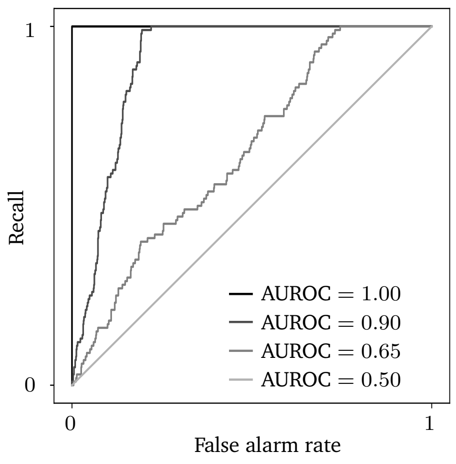


Figure 2.21.: ROC curves and corresponding AUROCs for different classifiers, own illustration.

Approaches

Researchers have proposed a wide variety of solution approaches to address the problem of AD. Depending on their specific characteristics and shared principles, these approaches can be categorized into distinct groups, see, e.g., [Berg20a; Tack20; Cho21; Geor21; Sehw21]. In this thesis, the categorization presented by [Ruff21b] will be followed, as it represents the most recent and holistic presentation at the time of this writing. According to [Ruff21b], AD approaches can be categorized into four main groups that encompass both shallow and deep approaches: (1) reconstruction-based models [Pidh18; Deec19; Tang20; Sale21], (2) classification models [Schö99; Tax04; Ruff18; Berg20a], (3) probabilistic models [Rose56; Parz62; Zong18; Nali19; Ren19], and (4) distance-based models [Fix51; Cove67; Breu00; Liu08; Berg20b].

Reconstruction-based models correspond to the majority of solution approaches in AD [Gola18; Ruff18; Berg20a; Kim20; Liu21]. Models falling into this category are trained to reconstruct normal data instances well, using the so-called reconstruction error as the anomaly score, which is typically defined as the MSE between the input and the reconstruction of the model [Ruff21a]. The implicit assumption of these approaches is that a model, which is able to reconstruct normal instances will fail to reconstruct anomalies due to their deviation from the learned model of normality [Berg20a; Ruff21a]. Autoencoders (AEs), Principal Component Analysis (PCA), and Kernel-PCA (KPCA) [Schö97] are among the most prominent examples in this category [Gola18].
Probabilistic models estimate the PDF of the normal data in a parametric or nonparametric fashion [Ruff21b]. KDE and the Deep Autoencoding Gaussian Mixture Model (DAGMM) [Zong18] represent two commonly used baselines in this category.
Classification methods avoid full density estimation but rather aim to directly learn an enclosing one-class decision boundary to separate normal from anomalous instances that offers a reasonable compromise between type I error and type II error [Berg20a; Ruff21b]. One-Class Support Vector Machine (OC-SVM)[Schö99], Support Vector Data Description (SVDD) [Tax04] and Deep Support Vector Data Description (Deep-SVDD) [Ruff18] are commonly used methods from this category.
Distance-based methods correspond to so-called lazy learning algorithms that do not encompass a training phase in the narrow sense but evaluate new data instances by direct comparison with the training instances instead [Ruff21a; Ruff21b]. Prominent methods falling into this category are Isolation Forest (IF) [Liu08] and PatchCore [Roth22]. Figure 2.22 provides an overview of the categorization presented by [Ruff21b].

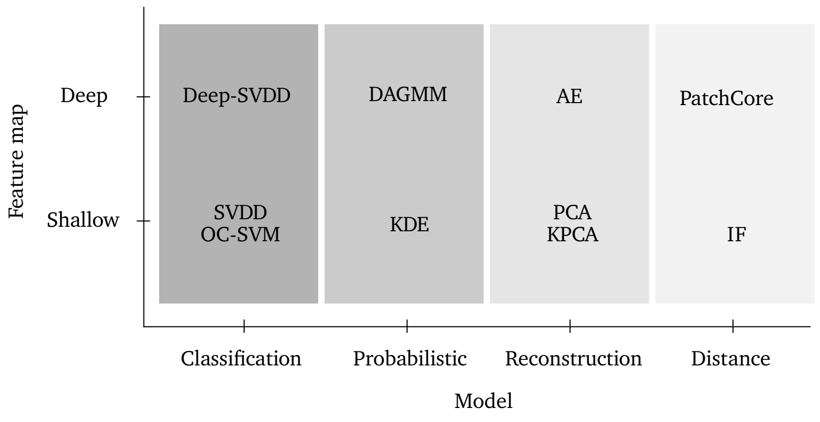


Figure 2.22.: Unifying view of different AD approaches, own illustration adapted from [Ruff21b].

Self-supervised anomaly detection

Self-supervised AD models mark a major milestone in AD research, given their ability to learn effective representations from unlabelled data [Ruff21a]. With respect to AD in general and semi-supervised AD in particular, this ability is of major interest since it is generally straightforward to collect normal data instances. The idea to use the learned representations from a self-supervised pretext task for AD was initially proposed by [Gola18] in the context of predicting geometric transformations that were applied to normal image data [Shen22]. Since then, self-supervised AD models have achieved state-of-the-art detection performance on common benchmark datasets [Hend19; Tack20; Sehw21; Qiu21]. In fact, at the time of this writing, some of the best-performing AD models rely on SSL [Lizn21].

From a general SSL perspective, as presented in Section 2.1.4, self-supervised AD methods focus either on handcrafted pretext tasks or employ a contrastive learning scheme [Tack20; Cho21; Ruff21a]. In order to derive the anomaly score, researchers operating in this field follow mainly two approaches. The first approach encompasses the evaluation of the softmax activation, i.e., the final output of the classifier that is used to solve the self-supervised pretext task [Gola18; Hend19; Ruff21a]. The second approach focuses on a two-stage framework, in which the learned representations from the pretext task are used to solve an AD downstream task [Sehw21; Sohn21]. In both scenarios, the success of the respective AD model mainly depends on the pretext task and the resulting learned representations. Thus, designing a suitable pretext task is of major importance for self-supervised AD [Li21a].
Regarding the unifying view of [Ruff21b], self-supervised AD models generally rely on deep feature maps and can be assigned to different model categories, depending on their individual characteristics. As will be shown in Chapter 5, SSMSPC corresponds to a classification model that uses a deep feature map and, as such, falls into the same category as Deep-SVDD.

2.2.2. Baseline algorithms

Based on the categorization presented in Figure 2.22, the following paragraphs will provide a short algorithmic introduction to the most relevant AD baselines that are going to be used to benchmark the performance of SSMSPC in Chapter 6. It is worth highlighting that the presented models represent the standard shallow and deep baselines used in the literature to assess the performance of AD models. Relevant self-supervised AD approaches will be presented in Section 3.1.

Autoencoder

AEs are among the most prominently used AD methods in the literature [Abat19; Chal19; Deha20; Hu20; Lai20; Deng21]. An AE is an NN, which is represented by a mapping $\phi : \mathbb{X} \rightarrow \mathbb{X}$. This mapping is composed of an encoder mapping $\phi_e : \mathbb{X} \rightarrow \mathbb{L}$ and a decoder mapping $\phi_d : \mathbb{L} \rightarrow \mathbb{X}$, such that $\phi(\mathbf{x}) = (\phi_d \circ \phi_e)(\mathbf{x})$, where generally $\dim(\mathbb{X}) > \dim(\mathbb{L})$ [Agga17]. Thus, the encoder ϕ_e compresses the data into a lower-dimensional representation, also known as the latent space $\phi_e(\mathbf{x}) \in \mathbb{L}$, while the decoder uses the encoded representation to reconstruct the original input $\mathbf{x}$ without learning the identity mapping [Good16]. The general loss function of an AE is the MSE loss between an input $\mathbf{x}$ and the corresponding reconstruction $\phi(\mathbf{x})$ [Ruff21b].
Depending on the application at hand, an AE can be represented by different NN types. For instance, tabular data can be processed with an AE that is represented by a simple MLP, while image data are typically processed with a convolutional AE.

(Kernel) principal component analysis

From a general ML perspective, PCA represents an UL method that is mainly used for linear dimensionality reduction [Bish06; Deis20]. More precisely, PCA linearly projects the data to a lower-dimensional orthonormal subspace that (1) provides optimal preservation of the information in the data measured in terms of an MSE reconstruction error and (2) removes any linear correlation of the input data [Tax01; Good16]. PCA is considered the default reconstruction baseline in general AD and the standard approach for process monitoring applications in manufacturing [Chia01; Maso02; Ge13; Harr20; Ruff21a]. There are various ways to derive PCA, e.g., via singular value decomposition or eigendecomposition [Barb12; Good16]. In this thesis, the eigendecomposition derivation of PCA will be presented.
Consider a z-score scaled data matrix $X \in \mathbb{R}^{n \times m}$ and the corresponding sample covariance matrix

$\hat{\Sigma} \in \mathbb{R}^{m \times m}$, which is given by $\hat{\Sigma} = \frac{1}{n-1} X^T X$ [Stra16]. The eigendecomposition of $\hat{\Sigma}$ is defined by Equation 2.29 [Deis20]

$$\hat{\Sigma} = Q \Delta Q^{-1}, \tag{2.29}$$

where $Q \in \mathbb{R}^{m \times m}$ is an orthogonal matrix whose columns correspond to the m eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_m$ of $\hat{\Sigma}$, Δ is a diagonal matrix where the diagonal entries are the corresponding eigenvalues $\lambda_1, \cdots, \lambda_m$ of $\hat{\Sigma}$, and Q^{-1} is the inverse of Q [Deis20]. The eigenvalues and the respective eigenvectors are sorted in descending order. The total variation v in the data is represented by the sum of all eigenvalues, i.e., $v = \lambda_1 + \cdots + \lambda_m$ [Stra16]. Thus, the i-th eigenvector $\mathbf{u}_i$ accounts for a fraction of λ_i / v of the total variation in the data [Stra16]. By selecting the first p eigenvectors from Q that account for a fraction of $(\lambda_1 + \cdots + \lambda_p)/v$, the so-called loadings matrix is obtained $P = Q[:, 1:p] \in \mathbb{R}^{m \times p}$ [Bieg22a]. These p eigenvectors are known as principal components [Stra16]. Note that p, in this sense, is a hyperparameter that needs to be chosen carefully depending on the problem at hand. With the help of the loadings matrix, the PCA score matrix T is obtained by multiplying the original data matrix X with the loadings matrix P, i.e., $T = XP \in \mathbb{R}^{n \times p}$. Thus, the loadings matrix acts as an encoder that maps the original data from an m-dimensional space to a p-dimensional space, where generally $p < m$. This step completes the dimensionality reduction part of PCA. In order to employ PCA for AD, the transpose of the loadings matrix is used to reconstruct the input X accordingly, i.e., $\hat{X} = TP^T \in \mathbb{R}^{n \times m}$. Consequently, the transpose of the loadings matrix P^T acts as a decoder. The reconstruction error is then again measured via the MSE between input and reconstruction. It is worth mentioning that PCA is equivalent to a linear AE, i.e., an AE that uses linear activation functions [Ruff21a].

KPCA was presented by [Schö97] and is a form of nonlinear PCA. Specifically, the basic idea is to map the data to a possibly infinite dimensional feature space $\mathbb{G}$ in which the data align along a linear hyperplane and apply the standard PCA in this feature space [Schö97; Agga17; Harr20]. This mapping $\mathcal{G} : \mathbb{X} \to \mathbb{G}$, however, does not need to be computed explicitly since only the dot product in this feature space is required, which can be evaluated implicitly with a kernel function $k(\cdot)$ [Ge13]. This is the essential aspect of the so-called kernel trick [Agga17; Harr20].

One-class support vector machines

OC-SVMs were first presented by [Schö99] and are inspired by the traditional Support Vector Machine (SVM) [Cort95; Vapn95] for classification. The following presentation of OC-SVM and SVDD is restricted to the main principles and omits an in-depth treatment of SVMs, as this would exceed the scope of the thesis. Interested readers are encouraged to investigate the standard literature such as [Vapn95; Bish06; Hast08; Murp12] that cover SVMs in a holistic manner.

The objective of an OC-SVM is to construct a hyperplane in a high-dimensional feature space $\mathbb{G}$ that separates the normal training data with the maximum margin from the origin with the help of the kernel trick [Schö99; Khan14; Ruff18]. For a data matrix $X \in \mathbb{R}^{n \times m}$, finding this hyperplane translates into solving the quadratic program defined by Equations 2.30, 2.31, and 2.32 [Schö99]

$$\min_{\mathbf{w}, d, \xi} \quad \frac{1}{2} ||\mathbf{w}||^2 - d + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i \tag{2.30}$$

$$\text{subject to} \quad \mathcal{G}(\mathbf{x}_i)\mathbf{w} \geq d - \xi_i \ \forall i \in \{1, \cdots, n\} \tag{2.31}$$

$$\xi_i \geq 0 \ \forall i \in \{1, \cdots, n\}. \tag{2.32}$$

Here, d corresponds to the distance from the origin to the hyperplane $\mathbf{w} \in \mathbb{G}$. $\xi_i \in \mathbb{R}$ correspond to so-called slack variables that act as regularization terms and allow certain constraints to be violated,

and $\nu \in [0;1]$ is a hyperparameter that controls the penalization trade-off [Schö99].
The resulting decision function f that assigns a new sample $\mathbf{x} \in \mathbb{X}$ to either the normal or the anomalous class is thus given by Equation 2.33 [Schö99]

$$f(\mathbf{x}) = \text{sign}(\mathcal{G}(\mathbf{x})\mathbf{w} - d) = \begin{cases} 1, & \text{if } \mathcal{G}(\mathbf{x})\mathbf{w} - d > 0 \\ -1, & \text{otherwise} \end{cases}. \tag{2.33}$$

Consequently, a point lying above the hyperplane in the feature space is flagged as normal, while all other points are flagged as anomalous. Note that in order to be in alignment with the terminology used in the literature, the anomalous class when presenting OC-SVM is assigned the label $y = -1$. Evaluating the decision function involves computing the dot product in the high-dimensional feature space $\mathbb{G}$. This can be done implicitly with the kernel trick by solving the dual form of the above quadratic program with the help of Lagrangian multipliers [Schö99; Agga17; Ruff21b].

Support vector data description

SVDD [Tax99; Tax01; Tax04] is closely related to OC-SVM. In fact, OC-SVM and SVDD are equivalent for kernels with a constant norm such as the Gaussian kernel [Ruff18; Ruff21a]. Instead of fitting a hyperplane that separates the data in the feature space with maximum margin from the origin, the idea in SVDD is to find a hypersphere of minimum volume with center $\mathbf{c} \in \mathbb{G}$ and radius $r > 0$ that encloses the training data in the feature space [Ruff18; Ruff21a]. Thus, the objective of SVDD is to solve the quadratic program defined by Equations 2.34, 2.35, and 2.36 [Tax04]

$$\min_{\mathbf{c},r,\xi} \quad r^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i \tag{2.34}$$

$$\text{subject to} \quad ||\mathcal{G}(\mathbf{x}_i) - \mathbf{c}||^2 \leq r^2 + \xi_i \; \forall i \in \{1, \cdots, n\} \tag{2.35}$$

$$\xi_i \geq 0 \; \forall i \in \{1, \cdots, n\}. \tag{2.36}$$

Here, the slack variables ξ_i are again used to loosen some of the constraints, ensuring that almost all samples are enclosed by the hypersphere [Tax04]. $\nu \in [0;1]$ controls the trade-off between the volume of the hypersphere and the penalization induced by ξ_i [Tax01; Tax04; Ruff18]. The corresponding decision function f is then given by Equation 2.37 [Tax04; Ruff21b]

$$f(\mathbf{x}) = \text{sign}(||\mathcal{G}(\mathbf{x}) - \mathbf{c}||^2 - r^2) = \begin{cases} 1, & \text{if } ||\mathcal{G}(\mathbf{x}) - \mathbf{c}||^2 - r^2 > 0 \\ -1, & \text{otherwise} \end{cases}. \tag{2.37}$$

Consequently, when a sample lies within the hypersphere ($f(\mathbf{x}) < 0$), it is considered normal, while a sample lying outside the hypersphere is considered anomalous. Similarly to OC-SVM, the kernel trick is used to avoid the explicit computation of $\mathcal{G}(\mathbf{x})$ in the possibly infinite-dimensional feature space [Tax04].

Deep support vector data description

Deep-SVDD [Ruff18] is a deep AD method for image data that has rapidly evolved into an important baseline to benchmark the performance of new AD models [Hend19; Berg20a; Shen20; Tack20; Chen21a; Qiu21; Shi23]. As the name suggests, Deep-SVDD is inspired by SVDD. However, in contrast to the original SVDD, Deep-SVDD trains a CNN encoder $\phi : \mathbb{X} \to \mathbb{L}$ via backpropagation to learn useful feature representations and maps the data into a minimum-volume hypersphere with center

$\mathbf{c} \in \mathbb{L}$ [Ruff18]. Given n training samples $\mathbb{X}_{\text{train}} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ and assuming a semi-supervised AD setting, the loss function in Deep-SVDD is given by Equation 2.38 [Ruff18]

$$\mathcal{L}_{\text{Deep-SVDD}} = \frac{1}{n}\sum_{i=1}^{n} ||\phi(\mathbf{x}_i) - \mathbf{c}||^2 + \frac{\lambda}{2}||\mathbf{w}||^2, \tag{2.38}$$

where $\mathbf{w}$ is the weight vector that contains the weights from all layers of the CNN. The second term in Equation 2.38 corresponds to the ℓ_2 regularization with regularization parameter $\lambda > 0$. Thus, $\mathcal{L}_{\text{Deep-SVDD}}$ minimizes the MSE of all training instances to the hypersphere center, which results in a contraction of the hypersphere volume. After training, Deep-SVDD outputs an anomaly score $h(\mathbf{x})$ that measures the squared error of a new data point $\mathbf{x}$ to the center $\mathbf{c}$ in the feature space [Ruff18], see Equation 2.39

$$h(\mathbf{x})_{\text{Deep-SVDD}} = ||\phi(\mathbf{x}) - \mathbf{c}||^2. \tag{2.39}$$

Note that according to [Ruff18], the center $\mathbf{c}$ is fixed prior to training as the mean network output for a subset of the training data resulting from an initial forward pass. Furthermore, the weights of ϕ are initialized with the encoder weights of a convolutional AE that has been pretrained on the available training data.

Deep autoencoding Gaussian mixture model

DAGMM [Zong18] is a deep AD method designed for tabular data that consists of (1) a compression network and (2) an estimation network [Zong18]. The compression network is represented by a deep AE $(\phi_d \circ \phi_e)$, which combines the reconstruction error $\mathcal{L}_{\text{MSE}}$ and the encoded representation $\phi_e(\mathbf{x})$ in a vector $\mathbf{z}$ that is used as an input for the estimation network. The estimation network ϕ_g is represented by a simple MLP encoder with a softmax output of size k, i.e., $\phi_g(\mathbf{z}) \in \mathbb{R}^k$. See Figure 2.23 for an illustration of the DAGMM architecture.

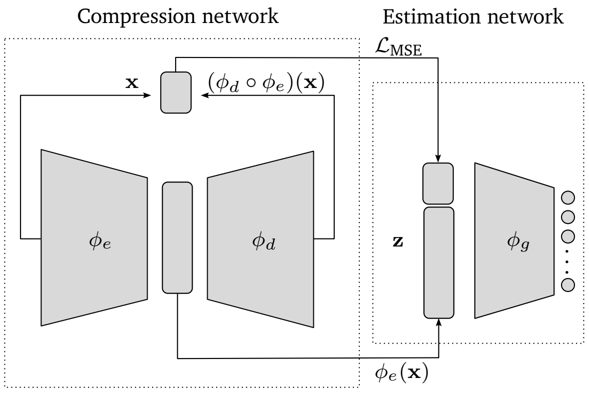


Figure 2.23.: DAGMM architecture, own illustration adapted from [Zong18].

The task of the estimation network is to assign a given input $\mathbf{z}$ to one of k normal distributions with unknown mean μ_i and covariance matrix Σ_i. For a set of n training samples, the required parameters for each of the k normal distributions are estimated according to Equation 2.40

$$\hat{p}_j = \frac{1}{n}\sum_{i=1}^{n}\phi_g(\mathbf{z}_i)_j, \quad \hat{\mu}_j = \frac{\sum_{i=1}^{n}\phi_g(\mathbf{z}_i)_j\mathbf{z}_i}{\sum_{i=1}^{n}\phi_g(\mathbf{z}_i)_j}, \quad \hat{\Sigma}_j = \frac{\sum_{i=1}^{n}\phi_g(\mathbf{z}_i)_j(\mathbf{z}_i - \hat{\mu}_j)(\mathbf{z}_i - \hat{\mu}_j)^T}{\sum_{i=1}^{n}\phi_g(\mathbf{z}_i)_j}, \tag{2.40}$$

where $\hat{p}_j$ represents the estimated probability that a training sample is assigned to normal distribution j. The estimated parameters are then used for the computation of the so-called sample energy $\mathcal{E}(\mathbf{z})$ that models the probability of observing the input sample [Zong18], as shown in Equation 2.41

$$\mathcal{E}(\mathbf{z}) = -\log\left(\sum_{i=1}^{k} \hat{p}_i \frac{e^{-\frac{1}{2}(\mathbf{z}-\hat{\mu}_i)^T \hat{\Sigma}_i^{-1}(\mathbf{z}-\hat{\mu}_i)}}{\sqrt{\det(2\pi\hat{\Sigma}_i)}}\right), \tag{2.41}$$

where $\det(\cdot)$ represents the determinant of a matrix. After training, the sample energy $\mathcal{E}(\mathbf{z})$ of a new sample is used as the anomaly score [Zong18]. The training objective in DAGMM, which is displayed in Equation 2.42, combines the individual components of the compression network and the estimation network into a single loss function that is trained in an end-to-end fashion [Zong18]

$$\mathcal{L}_{\text{DAGMM}} = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - (\phi_d \circ \phi_e)(\mathbf{x}_i)||_2^2 + \frac{\lambda_1}{n}\sum_{i=1}^{n} \mathcal{E}(\mathbf{z}_i) + \lambda_2 \sum_{j=1}^{k} \mathcal{P}(\hat{\Sigma}_j). \tag{2.42}$$

Here, λ_1, λ_2 are hyperparameters that control the trade-off between the second and third terms of the loss function. $\mathcal{P}(\hat{\Sigma}_j)$ is a regularization term that penalizes small values on the main diagonal of the respective covariance matrices.

Isolation forest

IF [Liu08] is a tree-based AD method that aims to explicitly isolate, i.e., separate anomalies from normal data, rather than learning to model the normal behavior of the data [Liu08]. More precisely, an IF represents a collection of individual so-called isolation trees in which each tree is tasked with isolating individual instances in the training set into single nodes, also known as leaf nodes [Agga17]. The main observation in IF is that anomalies are susceptible to isolation, as they correspond to rare data instances that deviate from the majority of the data [Liu08]. Due to these characteristics, anomalies are isolated closer to the root node of an individual tree than normal instances, i.e., the so-called path length $p \in \mathbb{R}$ from the root node to the respective leaf node is shorter [Gopa19].
Given a set of n training data points $\mathbb{X}_{\text{train}} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ and k isolation trees that constitute the IF. Every isolation tree in the IF recursively partitions $\mathbb{X}_{\text{train}}$ by randomly selecting a feature x_i and a split value q that separates the data in the current node into two sub-nodes at each iteration such that all data points for which $x_i < q$ are assigned to one node, while all remaining data points are assigned to the other node [Gopa19]. Assuming that there are no duplicates in the data, the recursion terminates when (1) a prespecified height limit for the tree is reached or (2) all data have been isolated into separate leaf nodes [Liu08].
Once the IF is constructed, the anomaly score $h(\mathbf{x})$ for a sample $\mathbf{x}$ is given by Equation 2.43 as the average path length across all isolation trees in the IF, i.e., the average number of edges a sample traverses in a tree until it reaches a leaf node [Liu08]

$$h(\mathbf{x})_{\text{IF}} = 2^{-\frac{\frac{1}{n}\sum_{i=1}^{k} p_i}{\tau(n)}}, \tag{2.43}$$

where $\tau(n)$ is the average path length of the IF for n training samples.

PatchCore

PatchCore [Roth22] is a powerful and, at the time of this writing, very recent deep AD method for image data that has been explicitly designed for the application in industrial settings, e.g., the

detection of defective manufactured parts [Roth22]. PatchCore operates in a semi-supervised AD setting and similarly to IF, employs a lazy learning strategy, i.e., it does not require an actual training phase in the sense of gradient-based optimization. More concretely, PatchCore leverages the mid-level feature representations of a specific form of CNN known as Residual Network (ResNet) [He15] that has been pretrained on the ImageNet dataset [Kriz12].
Given a training set $\mathbb{X}_{\text{train}}$ that contains only normal image data, PatchCore extracts so-called locally aware mid-level patch features $\mathbf{p} \in \mathbb{P}$ from mid-level feature maps by passing the training data through the pretrained ResNet and stores them in a memory bank $\mathbb{M}$. With the help of this memory bank, the anomaly score $h(\mathbf{X})$ for a new image $\mathbf{X} \in \mathbb{X}$ is given by the maximum distance score between the patch features and each nearest neighbor $\mathbf{m}^* \in \mathbb{M}$ [Roth22], as displayed in Equations 2.44 and 2.45

$$\mathbf{p}^*, \mathbf{m}^* = \underset{\mathbf{p}\in\mathbb{P}}{\text{argmax}}\ \underset{\mathbf{m}\in\mathbb{M}}{\text{argmin}}\ ||\mathbf{p} - \mathbf{m}||_2 \tag{2.44}$$

$$h(\mathbf{X})_{\text{PatchCore}} = ||\mathbf{p}^* - \mathbf{m}^*||_2. \tag{2.45}$$

The presentation of PatchCore completes this section. Having introduced the required fundamentals with respect to ML and AD, the focus will now shift to SPC.

2.3. Statistical process control

In this section, the relevant fundamentals of SPC are presented. The section is partitioned into two parts. In the first part, the underlying main assumptions and core components that constitute the general SPC framework are introduced. Following this, the relevant aspects of MSPC will be discussed.

2.3.1. Basic concepts and terminologies

SPC is a research field that emerged in the 1920s following the pioneering works of Walter A. Shewhart on using statistical methods to monitor and minimize variations in discrete manufacturing processes [Shew31; Wood99; Oakl03; Mont09; Ferr14]. The objective of SPC is to monitor the performance of a process over time to detect any anomalous events that occur and may affect the quality of the process output [Kour95; MacG95; Kour96; Stou00; Wood00; Ferr07]. Thus, SPC is essentially a form of AD. It is worth noting that although SPC was originally developed in the context of discrete manufacturing processes, its application is not limited to this specific area, but it can be applied to any type of process [Chan09; Mont09; Bisg12; DIN3534-2].

Definition 2.3.1 (Process). *"A process is the transformation of a set of inputs, which can include materials, actions, methods and operations, into desired outputs, in the form of products, information, services or - generally - results [Oakl03]."*

Definition 2.3.2 (Manufacturing process). *A manufacturing process is a process that outputs some sort of tangible product [Diet91].*

Definition 2.3.3 (Discrete manufacturing process). *A discrete manufacturing process is a manufacturing process that produces distinct products and typically involves mechanical processing, assembly, etc. [Liu18; Wang18; Yu23].*

Definition 2.3.4 (Quality). *"[The] degree to which a set of inherent characteristics of an object fulfills requirements [DIN9000]."*

Statistical control

Shewhart hypothesized that the quality produced by a manufacturing process and therefore the manufacturing process itself, is generally subject to two sources of variability: (1) common causes and (2) assignable causes [Shew31].

Definition 2.3.5 (Common causes). *Common causes of variability account for unidentifiable random changes that are consistently present and form an inherent part of a process, such as natural vibrations and temperature fluctuations [Shew31; Noor11; ISO7870-2].*

Definition 2.3.6 (Assignable causes). *Assignable causes of variability account for real changes that are not inherently part of a process and can theoretically be identified and eliminated, such as broken tools, environmental changes, and human failure [Shew31; Noor11; ISO7870-2].*

While assignable causes can be eliminated by adjusting the existing process, the elimination of common causes requires a fundamental change in the process itself and can be bound to severe resource allocations [ISO7870-2]. Common causes and assignable causes of variation are the basic components that constitute the notion of a statistically controlled process.

Definition 2.3.7 (Statistical control). *"A process is considered to be in a state of statistical control [...] if it is affected by random (or common, or chance) causes only, i.e., if no extraordinary, unexpected, or special (or assignable) causes have entered the system [ISO7870-1]."*

Consequently, in order to bring a process into a state of statistical control, it is necessary to identify and eliminate all assignable causes [Shew31]. Once a process operates in a state of statistical control, it is possible to predict within limits the expected variation of the process in the future [Shew31]. Having introduced the idea of statistical control, SPC can be defined as follows:

Definition 2.3.8 (Statistical process control). *SPC is a concept of monitoring the condition of statistically controlled processes over time to detect, locate, and eliminate assignable causes [Kour96; Ferr07].*

Control chart

In a broad sense, SPC encompasses a collection of seven tools that are also referred to as "the magnificent seven" in the quality management literature [Mont09; Noor11]: (1) histogram, (2) check-sheet, (3) pareto chart, (4) cause-and-effect diagram, (5) defect concentration diagram, (6) scatter diagram, and (7) control chart. Among these seven tools, the control chart [Shew31] is by far the most important and widely used tool for process monitoring in SPC [Jack85; Kour96; Stou00; Ferr07; Guh07; Mont09; Kete15; Jone21]. In fact, due to the prevalence of control charts, many researchers use the terms control chart and SPC interchangeably. Following this convention, the underlying thesis will use the term SPC as a synonym for the application of control charts for process monitoring.

Definition 2.3.9 (Control chart). *"[A] chart with control limits on which some statistical measure of a series of samples is plotted in a particular order to steer the process with respect to that measure" [ISO7870-1].*

Definition 2.3.10 (Control limit). *"[A threshold] determined statistically from the variation of the process due to the random causes alone." [ISO7870-1].*

The purpose of a control chart is to distinguish common causes from assignable causes [Noor11]. This is achieved by (1) taking samples from a process over time, (2) computing the values of a statistical measure, such as the mean or standard deviation, (3) plotting this measure inside the control chart, and (4) evaluating it against the control limits of an in-control process [Wood00]. If the statistical measure remains within the established control limits, the process is assumed to be in statistical control, or simply in control [Mont09]. Conversely, if the statistical measure falls outside the control limits, it is assumed that an assignable cause affects the process and the process is deemed to be out of control [Wood00; Mont09; ISO7870-2]. Note that in practical applications, it can be convenient to additionally include so-called warning limits. Such limits describe a more narrow region than the actual control limits, with the aim of providing an early warning signal for the monitoring personnel that the process might soon enter an out-of-control state [ISO7870-2]. If a process is out of control, a search for the assignable cause is triggered from the responsible personnel, which is also known as root-cause analysis [Kour95; Ferr07; Qiu21]. Here, it is important to mention that a control chart can only detect anomalies but provides no support to localize or eliminate the assignable causes, which represents an essential weakness for practical applications [Mont09].

Phase I and phase II monitoring

The application of a control chart involves two consecutive phases [Jin01; Bers07; Ferr07; Chan10; Jone14; Magg14; Kete15; Ahma21; Bieg22a; Bieg22b; ISO7870-2]. The first phase is the so-called model building phase or offline modeling phase, which is also known as phase I in SPC terminology [Ferr07; Ferr14; Tong17; Bieg22a; Bieg22b; Yu23]. The second phase is referred to as the online monitoring phase or phase II [Kour02; Ge13; Tong17]. Together, these two phases constitute the general SPC framework [Bieg23].

The objective in phase I is to build the control chart, i.e., fit the control limits for the subsequent online monitoring phase based on an in-control process [Wood04; Bers07; Jone14; Wood14; Wei16; Ahma21; Piri21; ISO7870-2]. This is accomplished by (1) bringing the underlying process into a state of statistical control and (2) assembling a so-called reference or historic in-control dataset that contains only samples in which the output of the process is in alignment with the respective quality requirements [Bers07; Ferr07; Mont09; Zwet21; ISO7870-2]. The historic in-control dataset is then used to fit the required control chart parameters.

Once the control chart is established, the actual process monitoring phase, i.e., phase II commences. The objective in phase II is to assess whether the process remains in control by continuously drawing samples from the process and comparing the respective statistical measure against the control limits that were established in phase I [Wood00; Mont09; Wood14; Kete15; Wood17; Qiu21; Zwet21].

Recall from the beginning of the section that SPC is essentially a form of AD. Taking into account the preceding statements, it can be concluded that the application of a control chart and thus SPC itself describes a semi-supervised AD problem setting [Wu21; Xie22; Bieg23]. From this point of view, phase I corresponds to the training phase of a shallow or deep AD model using only normal data, while phase II represents the actual AD phase. This analogy marks one of the key findings for the subsequent parts of the thesis.

Control chart types

Arguably the most simple and widely adopted univariate control chart is the Shewhart control chart [Stou00; Mont09; Shao13]. This type of control chart consists of a center line that typically represents the estimated mean $\hat{\mu}$ of a univariate statistical measure and two control limits, namely a Lower

Control Limit (LCL) and an Upper Control Limit (UCL) that are located at $\hat{\mu} \pm 3\hat{\sigma}$ [ISO7870-2]. The specific choice of the control limits is based on a fundamental assumption of Shewhart charts regarding the underlying distribution of the statistical measure under consideration. Specifically, a Shewhart control chart implicitly assumes that the monitored statistical measure represents a continuous random variable $\mathcal{X}$ that follows a normal distribution [Ferr14]. The PDF θ_{norm} of a normally distributed random variable is a symmetric bell-shaped curve that is given by Equation 2.46

$$\theta_{\text{norm}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \tag{2.46}$$

where μ and σ represent the true mean and standard deviation, respectively. For a normally distributed random variable, it can be shown that the probability of $\mu - 3\sigma \leq \mathcal{X} \leq \mu + 3\sigma$ is 99.73%. Thus, assuming that $\mathcal{X}$ is actually distributed according to θ_{norm} and taking into account the probability of a type I error which in this case is $100\% - 99.73\% = 0.27\%$, it is unlikely to observe values that fall outside these control limits.
Recall the threshold selection scheme in AD from Section 2.2.1. Establishing the control limits of a Shewhart chart in phase I is equivalent to a parametric threshold selection scheme, where the LCL and UCL represent the thresholds δ_1 and δ_2 that are retrieved by evaluating the quantile function $\Theta_{\text{norm}}^{-1}$ at the respective quantiles $\xi_1 = 0.00135, \xi_2 = 1 - 0.00135$, i.e., $\delta_1 = \Theta_{\text{norm}}^{-1}(\xi_1 = 0.00135)$ and $\delta_2 = \Theta_{\text{norm}}^{-1}(\xi_2 = 1 - 0.00135)$. Accordingly, following the categorization in Figure 2.22, a Shewhart control chart is essentially a shallow probabilistic AD model, where the respective statistical measure represents the anomaly score. Figure 2.24 schematically visualizes a Shewhart control chart and shows the corresponding estimated normal distribution.

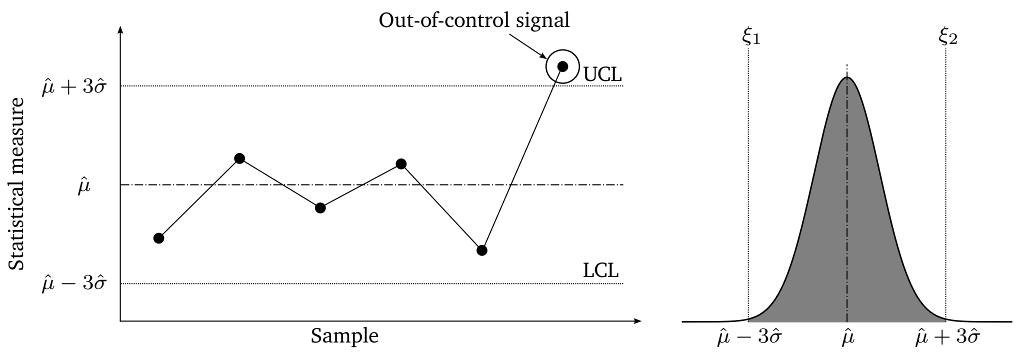


Figure 2.24.: Univariate Shewhart control chart. A Shewhart control chart is a shallow probabilistic AD model that parametrically fits a univariate normal distribution, own illustration adapted from [ISO7870-2].

The multivariate counterpart of the univariate Shewhart chart is the so-called Hotelling's T^2 [Hote47] chart, which represents the most commonly used control chart for monitoring multivariate process observations [Jack85; Mont09; Ahma21]. A Hotelling's T^2 chart plots the values of the Hotelling's T^2 statistic for each process observation over time. The Hotelling's T^2 statistic for a single process sample $\mathbf{x} \in \mathbb{R}^m$ is defined according to Equation 2.47 [Hote47; Jin01; Ferr07]

$$T^2 = (\mathbf{x} - \hat{\mu})^T \hat{\Sigma}^{-1} (\mathbf{x} - \hat{\mu}) \in \mathbb{R}, \tag{2.47}$$

where $\hat{\mu} \in \mathbb{R}^m$ is the estimated mean vector and $\hat{\Sigma}^{-1} \in \mathbb{R}^{m \times m}$ corresponds to the inverse of the estimated covariance matrix, both of which are obtained in phase I.

Taking a closer look at Equation 2.47, it can be seen that the Hotelling's T^2 statistic for a single observation represents the squared Mahalanobis distance, which expresses the distance of a sample to the mean of a population in terms of standard deviations. Due to this, a Hotelling's T^2 chart has a natural lower bound, i.e., LCL = 0, which indicates that the drawn sample coincides with the estimated mean of the population. Thus, for monitoring purposes, it is only necessary to determine a suitable UCL.
Similarly to the univariate Shewhart chart, the UCL of a Hotelling's T^2 chart can be fitted parametrically. However, doing so is equivalent to assuming that the underlying data follow a multivariate normal distribution, which is a wrong assumption for many practical situations and results in poor monitoring performance [Samu16]. Thus, instead of fitting the UCL parametrically, researchers typically use KDE to fit the PDF of the corresponding T^2 values in a nonparametric fashion [Samu16; Harr20; Yuan20; Bieg22a; Bieg22b; Yu23]. In this case, the Hotelling's T^2 chart corresponds to a nonparametric shallow probabilistic AD model. Figure 2.25 provides a schematic illustration of a Hotelling's T^2 chart fitted with KDE and the corresponding contours of the Hotelling's T^2 statistic using the example of two-dimensional process observations.

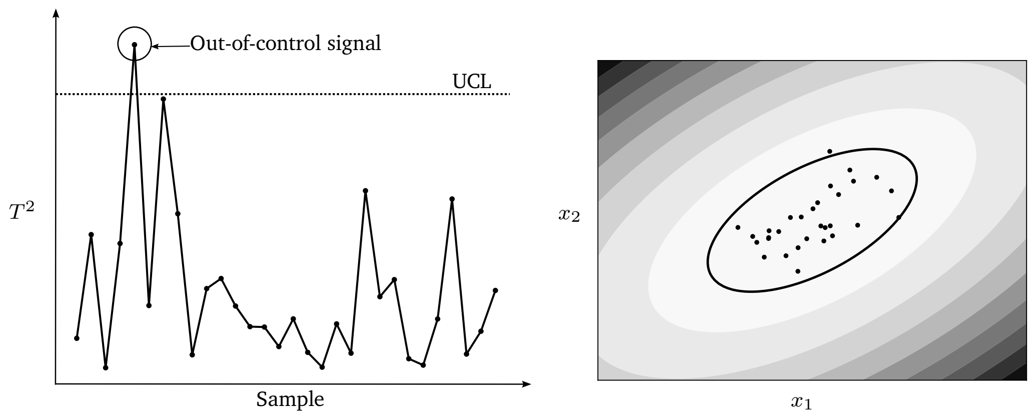


Figure 2.25.: Hotelling's T^2 control chart fitted with KDE. A Hotelling's T^2 control chart fitted with KDE is a shallow nonparametric probabilistic AD model. The KDE has been fitted with a Gaussian kernel. The black contour represents the 0.99 quantile of the PDF of T^2 values, own illustration.

Aside from the univariate Shewhart chart and its multivariate counterpart, the Hotelling's T^2 chart, there is a wide variety of other univariate and multivariate control charts, such as Cumulated Sum (CUSUM), Exponentially Weighted Moving Average (EWMA), Multivariate CUSUM (MCUSUM), and Multivariate EWMA (MEWMA) charts. However, since these charts are of minor relevance to this thesis, the interested reader is referred to the respective standard literature for further reference [Chia01; Oakl03; Mont09; Noor11; Harr20].

2.3.2. Multivariate statistical in-process control

The traditional approach to SPC as presented by Shewhart assumes that univariate physical measurements of predefined quality characteristics of manufactured parts are sampled in subgroups at equidistant time intervals from the process [Ferr07; Mont09; Magg14]. This form of SPC is referred to as univariate post-process SPC [Magg14]. Univariate post-process SPC was originally designed for the data-poor discrete manufacturing environments of the 1920s and 1930s [MacG97; Ferr14]. In these environments, actual process data were difficult to collect, limiting the amount of available process-related data to univariate physical quality measurements of manufactured parts [Wood99]. Almost a century after the presentation of Shewhart's work, the manufacturing environment in general and the discrete manufacturing environment in particular have changed dramatically in the presence of digitization and Industry 4.0. Nowadays, collecting large amounts of high-frequency process data during the process itself is relatively simple and affordable. Still, even though the prerequisites to applying SPC in discrete manufacturing have fundamentally changed, univariate post-process SPC based on Shewhart control charts continues to represent the state-of-the-art approach for practical applications [Mont09; Ferr14; Wees16; ISO7870-2]. In fact, the most recent versions of the International Organization for Standardization (ISO) norm series ISO 7870, which standardizes control charts restricts itself almost exclusively to univariate post-process SPC focusing, e.g., on Shewhart, CUSUM and EWMA control charts, while neglecting new developments in academia [ISO7870-2; ISO7870-4; ISO7870-6].

Paradigm shift from univariate post-process SPC to MSPC

In light of the changed prerequisites for the application of SPC in discrete manufacturing, the traditional univariate post-process SPC suffers from five major drawbacks. First, it neglects the vast amount of accessible process data that become available through digitization [Ferr14; Magg14; Wood17]. Specifically, instead of evaluating the process using process data collected during the process itself, the process condition is evaluated based on physical quality measurements of the process output. Interestingly, while the ISO norms focus on quality measurements rather than process measurements for SPC, ISO 3534-2 states that a "*control chart operates most effectively when the measure is a process variable which is correlated with an ultimate product or service characteristic*" [DIN3534-2]. Ignoring the process data is considered the main problem of univariate post-process SPC [MacG97; Magg14]. Second, the physical measurement of manufactured parts is a resource-intensive process, both in terms of time and cost.
Third, evaluating the process condition based on manufactured parts sampled at equidistant time intervals leads to a delay between the occurrence of an anomalous event in the process and the detection of its effects on the physical part [Gras15a]. Consequently, once a post-process chart signals an out-of-control state, all parts that have been produced since the last in-control measurement are subject to a 100% inspection.
Fourth, in the case of multivariate process observations, univariate post-process SPC requires building one control chart for each feature of the observation [ISO7870-7]. This leads to many different control charts that need to be monitored, while simultaneously resulting in missing alarms or false negatives since the interdependencies between the individual features are ignored [ISO7870-7]. This stresses the necessity to employ multivariate control charts, such as the Hotelling's T^2 chart, which are capable of summarizing the information contained in multivariate process samples in a single monitoring statistic.
Lastly, real-world discrete manufacturing processes often violate the overly simplistic assumptions of, e.g., Shewhart control charts, requiring more advanced methods to model the normal behavior of a process [Sun03; Gras15b; Di M16].
Motivated by the aforementioned drawbacks, researchers in discrete manufacturing encouraged a paradigm shift to transcend from univariate post-process SPC to so-called MSPC [Ferr07; Ferr14; Magg14; Gras15a; Di M16].

Definition 2.3.11 (Multivariate statistical in-process control). *An approach to SPC in which multivariate process data, e.g., high-frequency sensor signals, are collected during the process itself and are processed* *with the help of ML methods to fit the parameters of a multivariate control chart for in-process monitoring [Li20; Qiu21; Bieg22a; Bieg22b; Bieg23].*

Incidentally, prior to recognizing the necessity for a paradigm shift in discrete manufacturing, researchers in the process industries stressed the urgency to transcend from univariate post-process SPC to MSPC already back in the 1990s, see, e.g., [MacG94; MacG95; Kour96; MacG97]. This phenomenon can be explained by the advanced state of digitization within this particular field, which results from the necessity to effectively prevent process accidents that may cause fatalities or severe environmental damages [Chen19]. As a consequence, many approaches that were presented recently in the context of discrete manufacturing are inspired by research findings originating in the process industries. Figure 2.26 provides a visualization for the paradigm shift from univariate post-process SPC to MSPC.

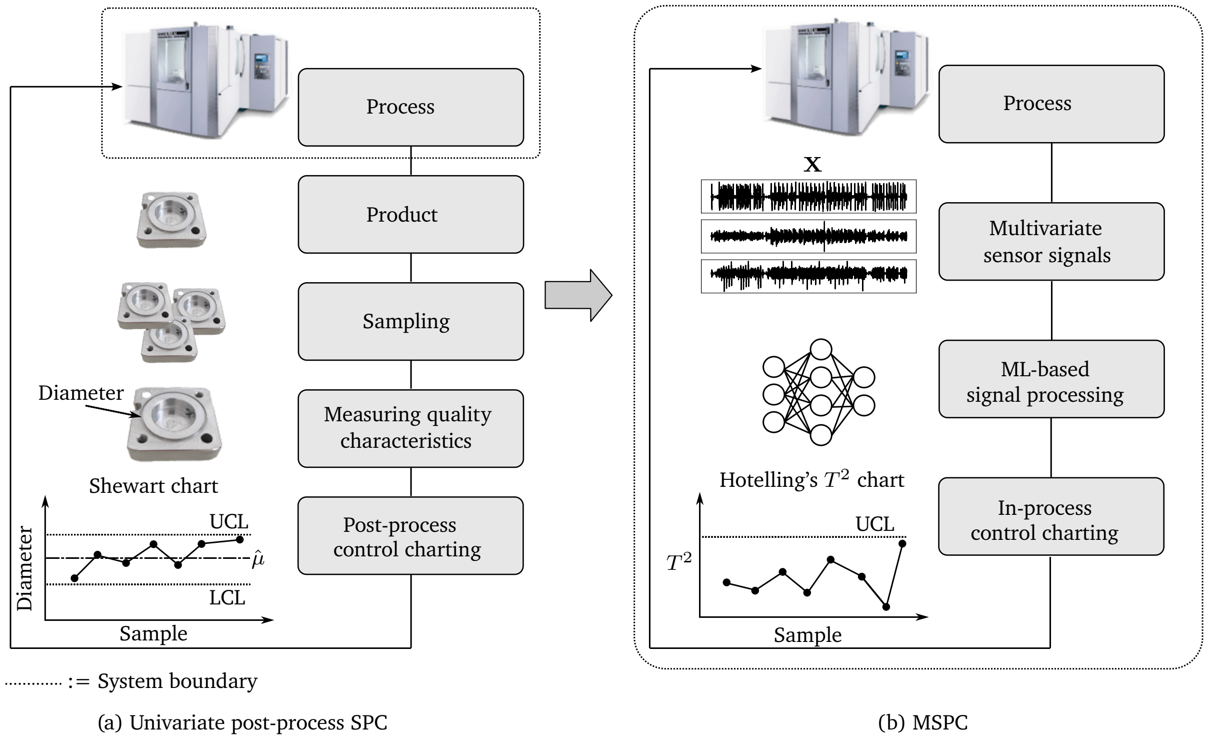

Process
Product
Sampling
Diameter
Measuring quality characteristics
Shewart chart
UCL
$\hat{\mu}$
LCL
Diameter
Sample
Post-process control charting
:= System boundary
(a) Univariate post-process SPC
Process
$\mathbf{X}$
Multivariate sensor signals
ML-based signal processing
Hotelling's T^2 chart
UCL
T^2
Sample
In-process control charting
(b) MSPC

Figure 2.26.: Paradigm shift in SPC. (a) Univariate post-process SPC: Univariate physical measurements of predefined quality characteristics of manufactured parts are sampled in subgroups at equidistant time intervals. (b) MSPC: Multivariate process data in the form of, e.g., high-frequency sensor signals are sampled continuously during the process and are analyzed with ML, own illustration adapted from [Bieg23].

Summarizing, MSPC, as presented here, characterizes a research field that establishes a connecting link between ML, AD, and SPC. Hence, all the concepts and methods presented in the previous sections are applicable to MSPC.
As a concluding remark, it shall be noted that the previous discussion of SPC is limited to the necessary contents for this thesis and represents only a small subset of the overall SPC research field. Other active research areas, such as control chart pattern recognition and profile monitoring, are not covered, and the reader is referred to the respective standard literature [Mont09; Noor11; Harr20].

2.4. Summary

Chapter 2 provides the necessary theoretical foundations along the three pillars of ML, AD, and SPC to answer the research questions formulated in Chapter 1 and lay the groundwork for the development of SSMSPC.
Section 2.1 gives a concise introduction to the fundamentals of ML. In this context, SSL is presented as a novel field that allows to learn effective representations from raw data with the help of, e.g., handcrafted pretext tasks and autogenerated pseudo labels that are created from unlabelled data.
Section 2.2 presents the required fundamentals of AD. Here, self-supervised AD is introduced as a major milestone in AD research, providing new means to develop powerful approaches for practical applications. The section concludes with a presentation of the most relevant baselines to benchmark the performance of new AD methods.
Section 2.3 addresses the field of SPC from the viewpoint of discrete manufacturing. It is demonstrated that SPC corresponds to a semi-supervised AD problem setting that involves the application of control charts as a visual monitoring tool for the detection of anomalies. In this context, it is shown that the existing control chart methods correspond to shallow probabilistic AD models that do not provide any means to localize anomalies, making the search for the root cause of an out-of-control situation a tedious endeavor. Following this, the paradigm shift from univariate post-process SPC to MSPC is discussed. It is shown that the current univariate post-process SPC scheme applied in practice is a relic from the past, neglecting the opportunities brought by digitization and Industry 4.0 that change the prerequisites for the application of SPC in discrete manufacturing. To this end, MSPC, as defined in this thesis, establishes the connecting link between ML, AD, and SPC that provides new opportunities to use multivariate control charts powered by ML to monitor discrete manufacturing processes based on process data acquired during the running process.

3. State of the art

The following chapter presents the state of the art by analyzing the relevant related work that has been identified in the course of two separate systematic literature reviews, covering the period from 2018 to 2023. This specific time frame has been selected to capture a holistic picture of the recent developments in the respective research areas. Section 3.1 introduces the related work regarding self-supervised AD from the viewpoint of fundamental ML research. Section 3.2 gives an overview of the related MSPC research from the perspectives of (1) continuous manufacturing processes such as those found in the process industry and (2) discrete manufacturing processes. The overall objective of this chapter is to derive the research gap in Section 3.3 that shall demonstrate the necessity for the development of SSMSPC. To accomplish this, the first step involves deducing the potential of SSL for MSPC, while the second step consists of demonstrating that the incorporation of SSL in MSPC is still mostly unexplored and has yet to gain momentum. The chapter concludes with a short summary of the most important results. Note that parts of this chapter are adapted from [Bieg23].

3.1. Self-supervised anomaly detection

This section introduces the related work with respect to self-supervised AD from the viewpoint of fundamental ML research. The following discussion places a special emphasis on the individual pretext tasks and demonstrates how the resulting learned representations can be used for AD purposes. Table B.1 in Appendix B.1 provides further details regarding the systematic literature review.
[Gola18] proposed Geometric Transformations (GeoTrans), a self-supervised AD approach for image data that follows the semi-supervised AD setting presented in Section 2.2.1. GeoTrans involves a handcrafted pretext task in which a CNN is trained to discriminate between a variety of geometric transformations that are applied to a training dataset containing only images of the normal class. Specifically, assuming a set of k geometric transformations, each transformation is applied to all the data in the training set, increasing the number of training samples by a factor of k. The intuition behind GeoTrans is that by training a CNN to distinguish the applied transformations, it must learn salient geometrical features that are unique to the individual classes [Gola18]. After completing the pretext task, GeoTrans detects anomalies in an unseen image by creating k augmented versions of the image using the predefined geometric transformations and evaluating the softmax output against the estimated distribution of the softmax output for normal images. The approach outperforms baselines such as OC-SVM and DAGMM in terms of AUROC on a variety of benchmark datasets.
[Hend19] introduced another self-supervised AD method for image data following the semi-supervised AD setting. Similarly to GeoTrans, their handcrafted pretext task involves training a CNN to discriminate among a set of geometric transformations. More precisely, they first rotate an image according to the rotation prediction scheme proposed by [Gida18] which is followed by a vertical and horizontal translation. The task of the network is to correctly classify the rotation and the respective translations. During testing, a previously unseen image is augmented using all possible combinations of rotations, vertical and horizontal translations. The anomaly score is then expressed as the sum of the assigned probabilities for the applied geometric transformations. The proposed approach is evaluated on the

CIFAR-10 benchmark dataset against baselines such as Deep-SVDD, GeoTrans, OC-SVM using the AUROC and shows superior performance.
[Berg20a] proposed an approach that is referred to as GOAD and is applicable to various types of data, including, e.g., images and tabular data. GOAD operates in a semi-supervised AD setting and uses a handcrafted pretext task where the training data are augmented with a set of l random affine transformations. Depending on the data type, the objective of the pretext task is to train an NN to discriminate between the employed transformations. As opposed to previous research, GOAD uses the center triplet loss [He18], which learns l clusters that have low intra-class variation and high inter-class variation. After completing the pretext task, GOAD detects anomalies in an unseen data point by first generating l augmented versions using the random affine transformations. The anomaly score is then given by aggregating the probabilities that the augmented versions are assigned to the right clusters. GOAD is evaluated on a variety of benchmark datasets, including image and tabular data, and outperforms other baselines such as Deep-SVDD, DAGMM, OC-SVM, and GeoTrans. The evaluation measure is AUROC for image data and f1-score for tabular data.
[Tack20] presented a self-supervised AD approach for image data that is based on contrastive learning and is referred to as Contrasting Shifted Instances (CSI). Their method is demonstrated to be applicable in a semi-supervised and supervised AD setting. CSI is based on SimCLR, which is introduced in Section 2.1.4. Prior to applying the SimCLR framework, the authors augment the training data with a set of so-called shifting transformations, such as rotations. In addition to SimCLR, they employ a handcrafted pretext task, in which the CNN has to predict the applied shifting transformation to the given image. The authors refer to this pretext task as classifying shifted instances [Tack20]. The final loss function of CSI represents the linear combination of the individual losses from each pretext task. Having completed the pretext task, the final anomaly score consists of two components that are composed via a linear combination, similarly to the loss function. Given an unseen image, the first component is derived by augmenting the image with each shifting transformation and passing the augmented images through the trained encoder to extract the learned representations. Following this, the mean over the product of the cosine similarity to the nearest train image and the norm of the learned representations is computed. The second component is based on the linear classifier that has been used to classify the shifted instances in the second pretext task. CSI is evaluated against, e.g., GOAD on a wide variety of image benchmark datasets using the AUROC as the evaluation measure and achieves state-of-the-art performance.
Inspired by Deep-SVDD, [Shen20] proposed a Temporal Hierarchical One-Class Network (THOC) for detecting anomalies in time series data that operates under the unsupervised AD setting. The authors use a recurrent NN to capture temporal dynamics in the time series data. A differentiable hierarchical clustering mechanism is employed that fuses the features from every hidden layer of the NN. For each time resolution, these features are used to define multiple hyperspheres that describe the normal behavior of the time series data. Based on the distance between the extracted features and the hypersphere centers, a multiscale SVDD objective is defined that allows THOC to be trained end-to-end. In addition to that, the authors include a self-supervised pretext task that involves multistep-ahead prediction. Specifically, by employing a simple linear model with a learnable weight matrix at each hidden layer of the NN, the pretext task involves predicting the value of time step t using the features extracted at time step $t - b$. For AD purposes, an anomaly score is defined that measures the extent to which the current observation deviates from the normal behavior as represented by the hyperspheres. To validate the effectiveness of THOC, the authors compare its performance to baselines such as IF, Deep-SVDD, and DAGMM on a variety of time series benchmark datasets. The evaluation measures are precision, recall, and f1-score.
[Qiu21] proposed Neural Transformation Learning for Anomaly Detection (NeuTraL AD) that is

designed for data types other than images, i.e., tabular data or time series data, and operates in a semi-supervised AD setting. Instead of handcrafting the data transformations to be used in the pretext task, NeuTraL AD derives an objective function termed deterministic contrastive loss that allows to jointly learn the appropriate data transformation as well as the anomaly scoring function. More precisely, given a fixed set of learnable transformations that are represented by simple NNs, the training data, such as scalograms of a time series, are augmented and passed to a CNN encoder network. Both the parameters of the transformations and the parameters of the encoder are trained using the deterministic contrastive loss. This loss encourages the idea that each augmented data point is similar to its original (unaugmented) version, while at the same time being dissimilar to the other augmentations of the same data point. In contrast to other approaches, the deterministic contrastive loss is not only the training objective, but it also acts as the anomaly score for testing purposes. NeuTraL AD is evaluated on a variety of time series and tabular benchmark datasets. The proposed method outperforms baselines such as OC-SVM, IF, Deep-SVDD, and DAGMM in terms of AUROC.
[Sohn21] presented a two-stage framework for self-supervised AD in image data that is designed for the semi-supervised AD setting and is closely related to the general SSL scheme presented in Figure 2.12. The first stage of the framework consists of learning effective representations from normal data by solving a self-supervised pretext task. After completing the representation learning phase, the employed encoder network is fixed, and the representations are used to fit a shallow AD model such as OC-SVM in the so-called AD downstream task. The key observation in this work, is to identify the importance of decoupling the representation learning phase from the actual downstream task. The authors evaluate their proposed framework using a variety of commonly used pretext tasks such as rotation prediction or contrastive learning, and find it to outperform common baselines such as Deep-SVDD, GeoTrans, and GOAD on a variety of image benchmark datasets using the AUROC as evaluation measure.
[Li21a] built on the two-stage framework by [Sohn21] and proposed a novel pretext task referred to as CutPaste. Their approach operates in a semi-supervised AD setting and is designed for image data. The proposed CutPaste augmentation first cuts a patch from an image that is then pasted to a random location within the same image. The intuition behind this augmentation strategy is to produce so-called spatial irregularities that approximate real anomalies. After augmenting all normal training data with the proposed CutPaste augmentation, the representation learning phase consists of predicting whether a given image has been augmented. With the self-supervised pretext task completed, a simple shallow probabilistic AD model based on the learned representation is constructed which then provides the anomaly score for an unseen image. Here, the authors suggest either fitting KDE or a parametric model such as a normal distribution, depending on the availability of data. CutPaste outperforms other approaches such as Deep-SVDD and simple rotation prediction in terms of AUROC using the MVTec [Berg19] AD benchmark dataset.
[Fu22] introduced Masked Anomaly Detection (MAD) for time series data that is used in a semi-supervised AD setting. The pretext task in MAD involves randomly masking a portion of the input time series, whose original values shall then be predicted by the remaining unmasked data. The intuition here is that the underlying NN learns the normal behavior of the data by being able to fill in the blanks that have been masked. Consequently, assuming an anomalous input, the prediction for a given mask would differ. Once the pretext training phase is completed, the anomaly score for an unseen time series is defined as the sum of deviations between the original instance and the prediction of the masked instances across all time steps. For evaluation purposes, the authors compare their proposed pretext task on two industrial benchmark datasets, i.e., the Tennessee Eastman Process (TEP) [Down93] and the Hardware In-The-Loop Augmented Industrial Control System (HAI) [Shin20]. In terms of

the evaluation measures, the recall on an individual time step basis and the AUROC are used for the TEP and HAI, respectively. However, as opposed to other research, the presented validation does not include other commonly used baselines but only considers the comparison between the MAD pretext task and the next-step prediction task across a variety of model architectures.
[Shen22] proposed a self-supervised AD approach based on contrastive learning for tabular data following the semi-supervised AD setting. For a given instance, their pretext task involves a contrastive learning objective that consists of maximizing the mutual information between a masked section of the instance and the remaining unmasked part. Specifically, they employ two separate NNs that map the masked part and the remaining part into a feature space of equal dimension in which the contrastive loss is applied. Similarly to [Qiu21], the training objective represents the anomaly score function to evaluate unseen data instances. The authors validate their approach on a wide variety of tabular benchmark datasets using baselines such as OC-SVM, IF, DAGMM, and GOAD. As the evaluation measure, the f1-score is used.
[Wang23] proposed Contrastive One-Class Anomaly Detection (COCA) for time series data that operates in a semi-supervised AD setting. The proposed pretext task involves augmenting the time series data by jittering and scaling to increase the number of training samples. Following this, the augmented time series data are mapped to a lower-dimensional feature space via an encoder network. These feature representations are then fed into an AE. The reconstructions and the original input to the AE are consequently passed to an MLP that creates two output projections. In this feature space, the contrastive loss is applied, which involves the computation of a weighted average of a set of variance and invariance terms. After completing the pretext training stage, the anomaly score for an unseen sample is defined based on the cosine similarity between the two output projections and the output of the AE encoder network. To validate the effectiveness of COCA, the authors compare it to baselines such as OC-SVM, IF, Deep-SVDD, and DAGMM on a variety of benchmark datasets. The f1-score is used as the final evaluation measure.
Table 3.1 summarizes the previously discussed approaches with respect to (1) the AD problem setting, (2) the input data type, (3) the ML model, (4) the pretext task category, (5) the pretext task objective, (6) the derivation of the anomaly score, and (7) the employed evaluation metrics.

Table 3.1.: Summary of the identified related work for self-supervised AD, own illustration.

No.	Paper	Problem setting	Input data	ML model	Pretext task	Pretext objective	Anomaly score	Evaluation metrics
1	[Gola18]	Semi-supervised	Image	CNN	Handcrafted	Classifying instances	Pretext task	AUROC
2	[Hend19]	Semi-supervised	Image	CNN	Handcrafted	Classifying instances	Pretext task	AUROC
3	[Berg20a]	Semi-supervised	Image, tabular	CNN, MLP	Handcrafted	Classifying instances	Pretext task	AUROC, precision, recall, f1-score
4	[Tack20]	Semi-supervised, supervised	Image	CNN	Handcrafted, contrastive learning	Classifying instances, contrasting instances	Pretext task	AUROC

No.	Paper	Problem setting	Input data	ML model	Pretext task	Pretext objective	Anomaly score	Evaluation metrics
5	[Shen20]	Unsupervised	Time series	Recurrent NN	Handcrafted	Classifying instances	Pretext task	Precision, recall, f1-score
6	[Qiu21]	Semi-supervised	Tabular, time series	CNN	Contrastive learning	Contrasting instances	Pretext task	AUROC
7	[Sohn21]	Semi-supervised	Image	CNN	Handcrafted, contrastive learning	Classifying instances or contrasting instances	Downstream task	AUROC
8	[Li21a]	Semi-supervised	Image	CNN	Handcrafted	Classifying instances	Downstream task	AUROC
9	[Fu22]	Semi-supervised	Time series	Any sequence-to-sequence model	Handcrafted	Classifying instances	Pretext task	AUROC, recall
10	[Shen22]	Semi-supervised	Tabular	MLP	Contrastive learning	Contrasting instances	Pretext task	f1-score
11	[Wang23]	Semi-supervised	Time series	MLP, AE	Contrastive learning	Contrasting instances	Pretext task	f1-score

3.2. Multivariate statistical in-process control

In this section, the related work regarding MSPC is presented both from the perspective of continuous manufacturing processes and discrete manufacturing processes. According to the definition of MSPC in Section 2.3.2, the following discussion targets in-process monitoring approaches operating in a semi-supervised AD setting in which multivariate process observations are processed with the help of ML to fit the control limits of a control chart. Within this context, emphasis will be placed on the employed ML approaches and the construction of the respective control charts. The rationale for including research targeting continuous manufacturing processes is twofold. First, it represents the origin of MSPC as a research field. Second, approaches developed in this area serve as a natural source of inspiration for MSPC research addressing discrete manufacturing processes. Table B.2 in Appendix B.2 provides further details regarding the systematic literature review.

3.2.1. Continuous manufacturing processes

[Yu18] presented a deep belief network for process monitoring. The authors argue that anomalous fluctuations in process data may be reduced or vanish with the extraction of layer-by-layer features in deep networks. To encounter this effect, they suggest an enhancement strategy to augment the anomalous fluctuations. The augmented features are then passed to a SVDD to compute the anomaly score. Compared to baselines such as PCA, KPCA, AE, and SVDD, the approach shows superior performance on a self-designed numerical example and the TEP. The PCA and KPCA baselines use a Hotelling's T^2 chart, which is based on the learned representations, and a so-called Squared Prediction

Error (SPE) chart that monitors the reconstruction errors. The remaining baselines use the respective anomaly scores as the monitoring statistic. The UCLs are established with KDE using the 0.975 quantile of the estimated PDFs. The evaluation metrics are recall, false alarm rate, and detection delay.
[Chen19] proposed a monitoring approach that employs a variational recurrent AE to capture nonlinear and temporal dependencies in the process data. They use the estimated negative variational score of a data instance as the anomaly score. Based on these scores, the authors define a control chart. The approach is validated with conventional baselines such as PCA, KPCA and AE using both a self-designed simulation study and the TEP. For each of the baseline approaches, two control charts are constructed. Specifically, the Hotelling's T^2 statistic is computed based on the learned representations of the respective encoder to establish a Hotelling's T^2 chart. In addition to that, the reconstruction error of each baseline is used to build an SPE chart. All UCLs are set at the 0.99 quantile of the respective PDFs. The proposed approach is shown to outperform the considered baselines in terms of recall and false alarm rate.
[Yu20] presented a Manifold Regularized Stacked Autoencoder (MRSAE) for process monitoring. Their approach consists of adding a manifold regularization term to the general AE loss that shall improve the learned representations and consequently the AD performance by preserving global and local structures in the data. A total of two control charts are designed for monitoring purposes. The first control chart represents a Hotelling's T^2 chart that is constructed based on the learned representations of the encoder. In addition to that, the authors use the reconstruction error of the AE to establish an SPE chart. The respective UCLs of both charts are fitted with the help of KDE using the 0.99 quantile. In order to validate the proposed method, its performance is evaluated on a self-designed numerical example, the TEP, and the Fed-Batch Fermentation Penicillin Process (FBFP). MRSAE is shown to outperform established baselines such as PCA and KPCA across all three validation studies using recall and false alarm rate as evaluation metrics. Each baseline uses a Hotelling's T^2 chart and an SPE chart for AD with UCLs that are obtained via KDE.
[Kong20] presented the Inner Product-Based Stacked Autoencoder (IPSAE) in which a regularization strategy is introduced that relies on the inner product between the outputs of a hidden layer prior to applying the nonlinear activation function to confront the overfitting problem of conventional stacked AEs. The authors use the learned representations from the encoder and the reconstruction error to design two control charts. Concretely, the first control chart monitors the squared error between an encoded data point and the mean of the encoded data. The second control chart corresponds to the standard SPE chart. Both UCLs are fitted via KDE based on the respective anomaly scores using the 0.95 quantile. The IPSAE is evaluated on a self-designed numerical example, and on the TEP. It is compared to common baselines, including PCA and KPCA. All baselines are equipped with a Hotelling's T^2 chart and an SPE chart employing a UCL obtained with KDE at the 0.95 quantile. IPSAE outperforms all baselines in terms of recall and false alarm rate.
[Tang20] proposed the Gaussian Mixture Variational Autoencoder (GMVAE) that combines a Gaussian mixture model with a VAE to monitor processes with multiple operating modes. Similar to previous approaches, they define two control charts covering the learned representations of the encoder and the reconstruction error. However, the authors design their approach in such a way that the respective anomaly scores approximately follow a χ^2-distribution, which consequently allows to determine the UCLs parametrically. For validation purposes, the GMVAE is evaluated on the TEP and in a real-world hot strip mill process. As baselines for comparison, the authors employ, e.g., a conventional VAE that uses two control charts based on the proposed monitoring statistics and a multimode version of PCA that employs Hotelling's T^2 and SPE control charts. GMVAE is shown to outperform all considered baselines using recall and false alarm rate as evaluation metrics.
[Yu21] presented an approach featuring a Convolutional Long Short-Term Memory Autoencoder

(CLSTM-AE) to learn effective representations from process data. The authors construct a Hotelling's T^2 chart based on the learned representations of the encoder and an SPE chart to monitor the reconstruction errors. The corresponding UCLs are obtained via KDE using the 0.99 quantile of the estimated PDFs. CLSTM-AE is validated on a simulated numerical experiment, the TEP, and the Continuous Stirred Tank Reactor (CSTR) benchmark. The approach shows superior performance compared to baselines such as PCA and KPCA as measured in terms of recall and false alarm rate. Each baseline features a Hotelling's T^2 chart and an SPE chart that are constructed using the learned representations and the reconstruction errors, respectively.
[Li22] introduced the Slow Feature Analysis-Aided Autoencoder (SFA-AE) to better capture the spatio-temporal relationship in the process data. In order to reliably detect anomalies, the authors define a total of three control charts. Specifically, two Hotelling's T^2 charts that cover the learned representations at different stages of the data processing pipeline, and an SPE chart to monitor the reconstruction errors. The respective UCLs are fitted via KDE using the 0.99 quantile of the estimated PDFs. The effectiveness of the proposed approach is validated with the TEP and is shown to outperform other AE baselines, using recall, false alarm rate, and f1-score as evaluation measures. For the baselines, Hotelling's T^2 charts and SPE charts are constructed, and the respective UCLs represent the 0.99 quantile of the estimated PDFs obtained with KDE.
[Lu23] proposed a Cascaded Bagging-PCA (CBPCA) combined with a CNN that incorporates an SSL mechanism for process monitoring. In a first step, three representations of the process data are generated using the wavelet transform. For each of these representations, the authors fit n PCA models by randomly sampling a subset of size m from the training data and randomly selecting the principal components for the respective model. Given a representation of a process observation $\mathbf{x} \in \mathbb{R}^d$, a reconstruction error image $X \in \mathbb{R}^{n \times d}$ is generated that contains the element-wise reconstruction errors for every PCA model. Based on X, a conventional PCA model is fitted to obtain $\hat{X}$, which represents a reconstruction of X. Both X and $\hat{X}$ are then assigned to a pseudo label and passed through a CNN whose task is to discriminate between the two classes. The above procedure is done simultaneously for each of the three representations. For process monitoring purposes, the authors establish a binary control chart that states whether a sample is normal or anomalous by evaluating if the CNNs correctly predict the pseudo labels of the input images. The approach is validated with the TEP and compared to a variety of PCA, AE, and CNN baselines. Recall and the false alarm rate are used as evaluation metrics.
Similar to [Lu23], other recent research published by, e.g., [Ai23; Li23] also employ an SSL scheme for process monitoring. However, these approaches operate in a supervised AD setting and are consequently omitted in the above discussion.
Table 3.2 summarizes the results of the preceding presentation regarding (1) the input data, (2) the ML model, (3) the monitoring statistic, (4) the derivation of the control limit, (5) the evaluation metrics, (6) the validation study, and (7) whether the suggested method employs SSL.

Table 3.2.: Summary of the identified related work for MSPC in continuous manufacturing processes, own illustration.

No.	Paper	Input data	ML model	Monitoring statistic	Control limit	Evaluation metrics	Validation study	SSL
1	[Yu18]	Time series	Deep belief network, SVDD	Distance to hyper-sphere center	Non-parametric (KDE)	Recall, false alarm rate, detection delay	Simulation, TEP	✗

No.	Paper	Input data	ML model	Monitoring statistic	Control limit	Evaluation metrics	Validation study	SSL
2	[Chen19]	Time series	Variational recurrent AE	Negative variational score	Non-parametric (KDE)	Recall, false alarm rate	Simulation, TEP	✗
3	[Yu20]	Time series	MRSAE	T^2, SPE	Non-parametric (KDE)	Recall, false alarm rate	Simulation, TEP, FBFP	✗
4	[Kong20]	Time series	IPSAE	SPE	Non-parametric (KDE)	Recall, false alarm rate	Simulation, TEP	✗
5	[Tang20]	Time series	GMVAE	T^2, SPE	Parametric	Recall, false alarm rate	TEP, hot strip mill	✗
6	[Yu21]	Time series	CLSTM-AE	T^2, SPE	Non-parametric (KDE)	Recall, false alarm rate	Simulation, TEP, CSTR	✗
7	[Li22]	Time series	SFA-AE	T^2, SPE	Non-parametric (KDE)	Recall, f1-score, false alarm rate	TEP	✗
8	[Lu23]	Time series	CBPCA, CNN	Softmax	Logical AND	Recall, f1-score, false alarm rate	TEP	✓

3.2.2. Discrete manufacturing processes

[Lejo18] investigated three ML approaches, namely AE, OC-SVM, and IF, regarding their suitability to monitor the stamping step in a press hardening process of automotive components. They use the force and speed signals of the slide over time as the basis for model training. Regarding the UCL for the AE, a fixed offset to the maximum reconstruction error on the train set is established. For the other approaches, the resulting decision functions as implemented in the scikit-learn library [Pedr11] are used. The authors find that the AE achieves the best performance measured in terms of precision, recall, and f1-score.
[Lind19] evaluated the applicability of two different ML approaches for process monitoring. Specifically, the authors investigate a simple clustering approach and a Long Short-Term Memory Autoencoder (LSTM-AE) using real-world process data obtained from the hydraulic pumps of an industrial press that produces wheel rims for the automotive industry. Regarding the first approach, cluster centers are computed that correspond to the individual process states of the press. For each of these states, the anomaly score is represented by the mean distance of the process data from the respective cluster center. In the AE approach, the reconstruction error of the process data is used as the anomaly score. In order to demonstrate the effectiveness of both approaches, a control chart based on the mean distances and an SPE chart are utilized to monitor the variation between normal and anomalous samples. Since this paper represents a feasibility study, the authors do not provide specifics regarding threshold selection schemes, nor do they evaluate the performance in terms of a comparison study with other baseline approaches.

[Lee19] presented a control chart based on the VAE for high-dimensional, nonlinear, and non-normally distributed process data. More precisely, they construct a Hotelling's T^2 chart using the representations learned by the encoder. In addition to that, they set up an SPE chart based on the reconstructions of the VAE. The control limits of the individual charts are determined parametrically using the 0.95 quantile of the respective PDFs. For validation purposes, the authors use a self-designed simulation study and process data from a real-world manufacturing process of Thin-Film Transistor Liquid-Crystal Displays (TFT-LCDs). The proposed approach outperforms common baselines such as PCA, KPCA, and AE each of which use Hotelling's T^2 and SPE charts with parametric UCLs. The performance comparison is conducted qualitatively via visual inspection of the individual control charts and quantitatively based on the false alarm rate.

[Lind20] presented a follow-up to the work of [Lind19] by introducing a novel approach for the detection and prediction of anomalies in process data. More precisely, the proposed method is based on an LSTM-AE that is combined with other NN structures in a cooperative prediction pipeline. Regarding the AD scheme, the authors preprocess the data with the wavelet transform prior to passing it through the AE. Similar to [Lind19], the proposed LSTM-AE relies on the reconstruction error that is plotted in an SPE chart to discriminate normal from anomalous process behavior. The validation of the approach is based on a real-world dataset obtained from the same discrete manufacturing process that was utilized in [Lind19]. However, as in their previous work, the authors omit an extensive comparison study featuring other baselines and assess the performance of the LSTM-AE only visually by comparing the reconstruction errors of normal and anomalous data.

[Prot20] visualized the two-dimensional learned representations of a VAE encoder to discriminate normal from anomalous process behavior based on a real-world Computerized Numerical Control (CNC) milling dataset that was acquired in the aerospace industry. In a preliminary preprocessing step, the authors extract statistical features from the raw sensor signals. Following this, a two-stage training scheme is employed, where the first stage encompasses a pretraining of the VAE to reconstruct its input. In the second stage, the learned representations serve as the input for an MLP to classify the respective process operation. In this stage, the encoder and the MLP are trained end-to-end. After training, the encoded normal data are arbitrarily encircled, and all samples that fall outside this boundary are flagged as anomalous. The authors restrict the evaluation of the process monitoring approach to a qualitative discussion and a visual comparison with rudimentary monitoring approaches. There is no comparison study conducted that includes state-of-the-art baselines.

[Ahma20] introduced a monitoring approach for manufacturing processes based on image and video data. Specifically, they employ a fast region-based CNN that extracts statistical features from the process data. These features are then used to compute the monitoring statistic of an MEWMA control chart. In order to discriminate normal from anomalous process behavior, the authors establish conventional Shewhart-based control limits. The effectiveness of the proposed approach is validated by a visual inspection of the control chart based on a simulated video. No further baseline comparison is conducted.

[Serg21] proposed two novel monitoring statistics to be used in conjunction with VAEs that are based on the first and second-order Taylor expansions of the expected reconstruction error. They demonstrate that the conventional monitoring schemes, which rely on control charts covering both the learned representations and the reconstruction errors have, non-negligible shortcomings. The authors validate their claims in a self-designed simulation study and a real-world hot steel rolling process. The UCLs correspond to the 0.95 quantile of the anomaly scores obtained from the validation set. The proposed monitoring statistics are shown to outperform common baseline statistics, such as Hotelling's T^2 and SPE, in terms of recall and false alarm rate.

[Fath21] employed a CNN-AE to monitor the melt pool images obtained from a laser powder bed

fusion process. As the result of an initial pretraining stage, the authors extract the learned representations by the encoder from melt pool images that contain both normal and anomalous samples. These representations are then passed to a clustering algorithm that separates normal from anomalous instances to create the historic in-control dataset containing only normal samples. This dataset is consequently used to retrain the AE. Following this, a Hotelling's T^2 chart is established in the encoding space, while a Shewhart chart is built on the variances of the reconstruction errors. The control limits are set via KDE. As opposed to the Hotelling's T^2 chart, the Shewhart chart relies on both LCL and UCL. The proposed approach shows superior performance on an experimental laser powder bed fusion process for a variety of different type I error settings using evaluation metrics such as precision, recall, and f1-score. The baseline for comparison is the neighboring-effect modeling method based on handcrafted features.
[Bieg22a] investigated the performance of different AE architectures that are combined with conventional multivariate control charts to monitor the force-displacement curves of a real-world sheet metal forming process. The authors conduct experiments with Hotelling's T^2 and SPE charts that are established using the learned representations as well as the reconstruction errors. The UCLs are set at the 0.99 quantile of the respective PDFs, which are estimated with KDE. The results of their study are in alignment with the findings of [Serg21], which suggest to solely relying on the reconstruction error for monitoring purposes. In addition to the different AE architectures, two baseline approaches, namely PCA and a naive mean curve, are used for validation. The PCA approach follows the conventional setup and employs both a Hotelling's T^2 chart and an SPE chart. The mean curve relies on the reconstruction error between the mean force-displacement curve and the current force-displacement curve to establish a Hotelling's T^2 chart. Surprisingly, the naive mean curve approach matches the performance of the best AE approach in terms of precision, recall, and f1-score.
[Bieg22b] extracted relevant information about the process condition of a real-world CNC milling process from the written notes of machine operators to automatically create the historic in-control dataset. Based on the in-control dataset, the authors demonstrate that a simple shallow PCA baseline approach can be successfully applied to monitor the underlying CNC milling process. In an initial preprocessing step, statistical features are extracted from the raw process data. Following this, both Hotelling's T^2 and SPE charts are established based on the learned representations and the reconstruction errors, respectively. The UCLs are set at the 0.99 quantile of the corresponding PDFs that are approximated with KDE. In order to assess the performance, the authors use precision, recall, and f1-score as the evaluation metrics.
[Oshi23] presented an LSTM-AE approach to monitor a CNC turning process of Inconel 718 for the aerospace industry using the raw time series signals obtained from acoustic emission and sound sensors. The reconstruction error is used as the anomaly score and passed to a sigmoid activation that allows to interpret the resulting score as the probability of an anomalous process. In the presented study, the authors compare the performance of two LSTM-AEs that are trained with the raw acoustic emission signal and sound signal, respectively, to investigate which signal is more suitable for the task at hand. The comparison is conducted by visual inspection of the respective control charts. The approach is not validated against other baselines.
[Sun23] proposed an improved version of the conventional AE to monitor machining processes. Specifically, they adjust the loss function of the AE in such a way that it impels the normal data to be distributed identically in the encoding space. The anomaly score is then represented by a distance factor that is defined in the encoding space and reflects the degree to which a given sample is anomalous. For validation purposes, the authors conduct extensive experiments with a cutting tool breakage dataset and a cutting tool wear dataset featuring a CNC milling process. The proposed approach is shown to outperform common baselines such as Deep-SVDD, OC-SVM, AE, and IF, using

AUROC as evaluation metric.
Table 3.3 summarizes the presented research according to the same categories used in Table 3.2.

Table 3.3.: Summary of the identified related work for MSPC in discrete manufacturing processes, own illustration.

No.	Paper	Input data	ML model	Monitoring statistic	Control limit	Evaluation metrics	Validation study	SSL
1	[Lejo18]	Time series	AE, OC-SVM, IF	SPE, distance to hyper-plane, average path length	Non-parametric	Precision, recall, f1-score	Press hardening process	✗
2	[Lind19]	Time series	Clustering, LSTM-AE	Distance to cluster center, SPE	Not specified	Visual comparison	Press process	✗
3	[Lee19]	Time series	VAE	T^2, SPE	Parametric	Visual comparison, false alarm rate	Simulation, TFT-LCD process	✗
4	[Lind20]	Time series	LSTM-AE	SPE	Not specified	Visual comparison	Press process	✗
5	[Prot20]	Time series	VAE	Distance to cluster center	Manual selection	Qualitative discussion, visual comparison	CNC milling process	✗
6	[Ahma20]	Image, video	CNN	MEWMA	Shewhart-based	Visual comparison	Simulation	✗
7	[Serg21]	Image	VAE	Taylor expansion of reconstruction error	Non-parametric	Recall, false alarm rate	Simulation, hot steel rolling process	✗
8	[Fath21]	Image	CNN-AE	T^2, reconstruction error variance	Non-parametric (KDE)	Precision, recall, f1-score	Laser powder bed fusion process	✗
9	[Bieg22a]	Time series	AE	T^2	Non-parametric (KDE)	Precision, recall, f1-score	Press process	✗
10	[Bieg22b]	Time series	PCA	T^2, SPE	Non-parametric (KDE)	Precision, recall, f1-score	CNC milling process	✗
11	[Oshi23]	Time series	LSTM-AE	SPE	Not specified	Visual comparison	CNC turning process	✗

No.	Paper	Input data	ML model	Monitoring statistic	Control limit	Evaluation metrics	Validation study	SSL
12	[Sun23]	Time series	AE	Distance factor	Non-parametric	Visual comparison	CNC milling process	✗

3.3. Derivation of the research gap

The preceding presentation of the state of the art encompasses a total of 31 research papers that were identified as a result of two systematic literature reviews. Of these 31 articles, 11 fall into the category of self-supervised AD. The remaining 20 papers address the research field of MSPC, taking into account both continuous manufacturing processes and discrete manufacturing processes, represented by 8 and 12 articles, respectively.
Using the insights gained from analyzing the related work, the following section will derive the research gap that justifies the need for the proposed SSMSPC approach in the underlying thesis.

3.3.1. Important findings: self-supervised anomaly detection

The investigated related work regarding self-supervised AD as summarized in Table 3.1, provides five main findings.
First, $> 90\%$ of the identified articles address the semi-supervised AD setting. This aspect supports the claim from Section 2.2.1 that SSL is particularly appealing for this problem setting, as it is comparatively straightforward to collect large quantities of normal data instances that can consequently be used to learn effective representations of the normal class. These representations, in turn, possess strong discriminative power to detect deviations from the normal state. The successful application of self-supervised AD in semi-supervised AD settings highlights its general eligibility for MSPC.
Second, regarding the classification of SSL approaches as depicted in Figure 2.13, the majority of the authors ($> 70\%$) consider a handcrafted pretext task in their approach as opposed to a pure contrastive learning scheme. Recall from Section 2.1.4 that handcrafted pretext tasks represent one of the earliest and most commonly applied forms of discriminative SSL and allow to incorporate domain knowledge to learn domain-specific representations. The ability to integrate domain knowledge in the representation learning phase is of major relevance for practical applications, such as MSPC. Despite the dominance of handcrafted pretext tasks, it is worth noting that the application of contrastive learning schemes is continuously gaining momentum, as can be seen by the fact that more than half of the reviewed articles starting in 2021 rely on this form of SSL.
Third, all articles that feature a handcrafted pretext task rely on a pretext task objective that involves the classification of transformed or augmented instances. The appealing aspect of this procedure is that, depending on the design, the resulting training dataset size for the representation learning phase grows proportionally to the number of augmentations. This feature is especially relevant for MSPC in discrete manufacturing processes, where even the acquisition of normal data can become challenging due to limiting factors such as, e.g., excessive processing times.
Fourth, most approaches ($> 80\%$) derive the anomaly score based on the pretext task, exploiting the softmax activation of the employed feature extractor in various ways, while only two articles use the learned representations to solve an AD downstream task according to the two-stage framework presented by [Sohn21]. Even though the majority of articles rely on the softmax activation, the results of [Sohn21] and [Li21a] demonstrate the power of decoupling the representation learning

phase from the actual AD downstream task. The main benefit of this procedure is that it allows increased flexibility, as the employed AD model in the downstream task is not tasked with learning representations but remains consistent with the actual AD objective [Sohn21]. Regarding MSPC, this provides the opportunity to leverage the learned representations from the pretext task to establish conventional multivariate control charts.
Fifth, $> 90\%$ of the presented articles provide extensive benchmark studies, highlighting the fact that AD methods incorporating SSL consistently outperform leading shallow and deep AD approaches across a variety of input data types. This gives a strong indication that MSPC approaches based on SSL could exceed the performance of existing MSPC approaches.
Summarizing the above findings, it can be concluded that SSL provides enormous potential to advance the research field of MSPC.

3.3.2. Important findings: multivariate statistical in-process control

Having identified the potential of SSL for MSPC, the subsequent discussion presents the main findings from analyzing the related work of MSPC, focusing on continuous manufacturing processes and discrete manufacturing processes.

Continuous manufacturing processes

With respect to the investigated MSPC research targeting continuous manufacturing processes, as summarized in Table 3.2, it can be observed that almost all approaches (7 out of 8) rely on reconstruction-based models featuring some form of AE or PCA. This supports the statement from Section 2.2.1 that models falling into this category are most widely employed in the AD literature, which is due to their intuitive working mechanisms and suitability for the semi-supervised AD setting.
In terms of the monitoring statistics, it can be seen that the Hotelling's T^2 statistic as well as the SPE statistic are utilized in 50% and $> 60\%$ of the identified articles, respectively, emphasizing the relevance of these statistics in the field. It is also worth noting that every second paper analyzes the Hotelling's T^2 chart in conjunction with the SPE chart. In these scenarios, the learned representations are monitored by the Hotelling's T^2 chart.
In order to fit the control limits of the respective control charts, 75% of the approaches favor a nonparametric derivation based on KDE, confirming the previous claims from Section 2.2.1 that it represents one of the most frequently used density estimation techniques. To maintain a low false alarm rate, the control limits correspond in most cases to the 0.99 quantile of the estimated PDF.
Focusing on the individual validation studies, it stands out that all approaches are evaluated on a common benchmark dataset, namely the TEP, which allows effective comparison between newly proposed methods. More than half of the articles incorporate a self-designed simulation study as the first evaluation step. Interestingly, a real-world continuous manufacturing process is evaluated in just a single instance.
Of all the investigated articles, only one presents a discriminative SSL scheme for representation learning to be used in the context of MSPC. The fact that this article has just recently been published in 2023 emphasizes that SSL is still an underexplored area in this particular field.

Discrete manufacturing processes

Regarding the related work in MSPC focusing on discrete manufacturing processes, it can be noted that most of the analyzed articles ($> 90\%$) incorporate a form of reconstruction-based approach, similarly to the investigated research addressing continuous manufacturing processes. Here, the AE is

again the major AD model, being used in 10 out of 12 papers.
The employed monitoring statistics are dominated by the SPE statistic, which is used in every second approach. This is followed by the T^2 statistic being incorporated in one third of the articles examined. It is worth mentioning that only a few papers, such as [Ahma20; Bieg22a; Bieg22b], explicitly establish the link between AD and MSPC, while the remaining researchers place their work implicitly in the general AD umbrella. This leads to discrepancies in the terminology used for identical concepts.
In terms of the control limits, 50% of the articles suggest the derivation based on a nonparametric estimation of the underlying PDF, while every fourth article explicitly states the usage of KDE. In three cases, the authors omit the control limit derivation and simply compare normal and anomalous samples visually in the control chart.
Regarding the validation studies, it stands out that $> 90\%$ of the presented approaches are evaluated using a real-world discrete manufacturing process. However, the corresponding datasets for these processes are not made publicly available, which impedes a comparison between the developed approaches across different articles. In addition, only a few researchers actually compare their approach to other baselines. In fact, $> 40\%$ omit a comparison with commonly used baselines. This, and the fact that $\geq 25\%$ of the investigated articles correspond to investigation studies regarding the applicability of common AD models, lead to the conclusion that MSPC research in discrete manufacturing processes is less developed and still in its infancy when compared to continuous manufacturing processes. Two contributing factors to this phenomenon can be found in the ongoing digitization efforts in the discrete manufacturing industry and the general scarcity of publicly available benchmark datasets. These factors stand in direct relation to the paradigm shift from univariate post-process SPC to MSPC that has been presented in Section 2.3.2, highlighting that there is still plenty of research potential in this field.
As a concluding remark, none of the presented articles involves the application of SSL.

3.3.3. Research gap

Summarizing, the preceding discussion revealed the inherent potential of SSL to advance the research field of MSPC by employing discriminative self-supervised pretext tasks that learn more effective representations of the normal class.
Still, the state of the art regarding MSPC predominantly involves the application and investigation of established shallow and deep AD methods that fall into the category of reconstruction-based approaches, with the AE representing the most frequently used model. In terms of applying SSL in the context of MSPC, only one article could be identified that involved a self-supervised, handcrafted pretext task focusing on continuous manufacturing processes. However, with respect to discrete manufacturing processes, no research could be found that involved the application of discriminative SSL approaches.
Thus, it can be concluded that the application of SSL in the context of MSPC represents a novel research field that is still mostly unexplored and requires more attention from the research community. Within this new research field, the underlying thesis addresses the following research gap, which is depicted in Figure 3.1:

Research gap

There is no approach that combines the paradigm of SSL with MSPC in the context of discrete manufacturing processes.

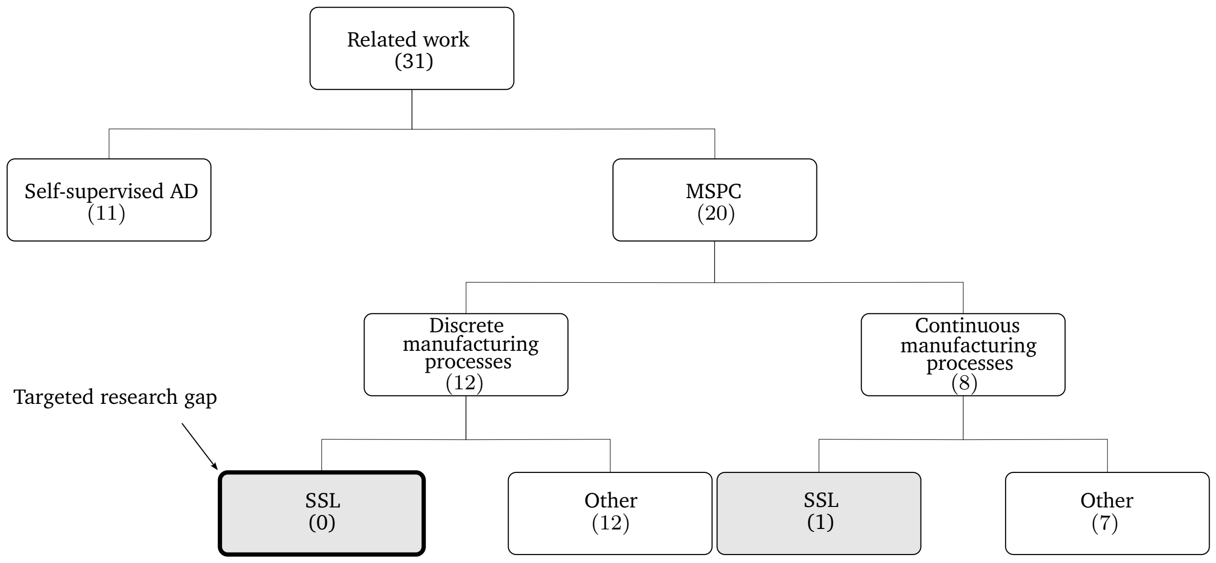


Figure 3.1.: Breakdown of related work and identified research gap, own illustration.

3.4. Summary

Chapter 3 presents the state of the art regarding self-supervised AD and MSPC based on two systematic literature reviews in order to identify the research gap that is targeted by this thesis.
The first literature review addresses related work in the field of self-supervised AD from a fundamental ML research perspective. The analysis of the considered approaches indicates that SSL possesses enormous potential to advance the field of MSPC.
The second literature review focuses on the related work with respect to MSPC from the viewpoint of continuous manufacturing processes and discrete manufacturing processes. The analysis of the identified research reveals that the application of SSL in the context of MSPC represents a novel, unexplored research field that requires further research efforts. Based on this finding, it is shown that there is currently no approach that combines the paradigm of SSL with MSPC in terms of discrete manufacturing processes. This key observation marks the research gap that is targeted with the development of SSMSPC.

4. Overall objective

The following chapter presents the overall objective of this thesis and is subdivided into three main sections. Section 4.1 concretizes the research objective and the research questions that were outlined in Section 1.1 using the findings from the previous two chapters. Section 4.2 elaborates on the requirements. This encompasses the presentation of the internal requirements that SSMSPC places on a discrete manufacturing process and the derivation of the minimum external requirements that need to be satisfied by SSMSPC to be of practical relevance. Based on the concretized research objective and the requirements, Section 4.3 formalizes the problem statement. A short summary of the most important findings concludes the chapter. Note that parts of this chapter are adapted from [Bieg23].

4.1. Concretized research objective

Based on the identified research gap in Section 3.3 and the persisting need for a paradigm shift from univariate post-process SPC to MSPC in practical applications, as shown in Section 2.3.2, the research objective in this thesis encompasses:

Research objective

The development of an SSL approach for MSPC in discrete manufacturing processes.

This approach is referred to as SSMSPC. The objective of SSMSPC is to provide a novel approach for MSPC based on SSL that possesses the capability to both detect and locate anomalies based on multivariate process data in discrete manufacturing processes. Note that for the remainder of this thesis, the usage of the term MSPC implicitly refers to MSPC in discrete manufacturing processes if not stated otherwise.
The research objective specified above raises three central research questions that shall be answered in the course of this thesis and are discussed in more detail below.

Research question 1

How can SSL be incorporated into an MSPC system for discrete manufacturing processes?

In Section 2.2.1, it is argued that the design of a suitable pretext task represents a key element of self-supervised AD. This claim is confirmed by the individual approaches presented in Section 3.1, in which different pretext tasks, either handcrafted or via a contrastive learning scheme, result in state-of-the-art performance. Aside from the pretext task design, the derivation of the anomaly score once the pretext task is completed represents another important aspect to consider. Here, it is shown that researchers primarily rely on the softmax output of the respective feature extractor from the pretext task, while some articles derive the anomaly score from a decoupled AD downstream task. The first research question addresses the above issues in the context of MSPC. Chapter 5 develops the main results to answer this research question.

Research question 2

What are the benefits of SSL compared to existing shallow and DL approaches for MSPC in discrete manufacturing processes?

Section 3.1 highlights the potential of SSL for MSPC. This potential is represented by five main findings resulting from an analysis of related work in the field of self-supervised AD. Within this context, it is shown that AD methods based on SSL consistently outperform leading shallow and deep AD baselines across various data types. In this regard, the second research question addresses the investigation of the benefits of SSL with respect to MSPC. In order to answer this question, the developed SSMSPC approach will be evaluated in Chapter 6 based on two real-world discrete manufacturing datasets against a variety of shallow and deep baselines, including current self-supervised AD approaches.

Research question 3

How can SSMSPC be applied in practice to replace univariate post-process SPC, which is accepted in industry and embodied in today's norms?

Section 2.3.2 presents the paradigm shift from univariate post-process SPC to MSPC. This paradigm shift is proposed to encounter the evident drawbacks resulting from the application of univariate post-process SPC in the light of recent technological advances in the discrete manufacturing environment. In Section 3.2.2, it is demonstrated that there exists a large and continuously evolving body of research regarding MSPC involving the investigation of advanced ML methods to detect anomalies in process data. However, instead of incorporating these approaches into practical applications, the ISO norms, even in their latest releases and despite their awareness of the existing research efforts, remain faithful to the conventional SPC scheme. This discrepancy leads to the assumption that practitioners consider the existing MSPC research as inappropriate to replace the accepted standard. Thus, the third research question aims at finding an answer to bridge this widening gap by investigating how SSMSPC can be used in real-world applications to replace univariate post-process SPC. To this end, a usability study is conducted in Chapter 6 that compares SSMSPC with the conventional univariate post-process SPC scheme in a real-world discrete manufacturing process with human machine operators.

4.2. Requirements

This section elaborates on the requirements, which are categorized into internal and external requirements. Section 4.2.1 addresses the internal requirements that need to be satisfied by a discrete manufacturing process in order to be eligible for SSMSPC. Conversely, Section 4.2.2 derives the minimum external requirements that need to be met by SSMSPC to be of practical relevance for monitoring discrete manufacturing processes.

4.2.1. Internal requirements

SSMSPC places three major requirements on a given discrete manufacturing process to ensure its eligibility, taking into account (1) the process characteristics, (2) the input data, and (3) the availability of a historic in-control dataset. In other words, SSMSPC can be applied to any discrete manufacturing process for which the internal requirements are satisfied. These requirements will be discussed in the following.

Machining process

The underlying discrete manufacturing process is assumed to represent a machining process (milling, turning, grinding, drilling, etc.) in which high-frequency process data, such as vibrations and cutting forces, are recorded throughout the production cycle of a part. While it is valid to assume that SSMSPC will work for other discrete manufacturing processes as well, it has been developed and validated in this specific context.

Multivariate time series data associated with final quality characteristics

SSMSPC further assumes that the process data are recorded in the form of a multivariate time series. These process data must be associated with the final quality characteristic of the produced part. Since the inherent objective of any machining process is to produce parts that meet the quality specifications of a customer, this requirement is necessary to ensure that the monitoring scheme is in alignment with the economic objective of the process. Recall from Section 2.3.2 that this is an essential requirement for a control chart scheme to operate most effectively [DIN3534-2].

Historical in-control dataset

Since SSMSPC represents an approach for MSPC, it operates by definition under the semi-supervised AD setting. Thus, for training purposes, SSMSPC requires the existence of a training dataset containing only data that represent the normal class. Formulated in SPC terminology, the training dataset must originate from a process that operates in a state of statistical control.

4.2.2. External requirements

Having addressed the internal requirements, the focus will now be shifted to the minimum external requirements that SSMSPC needs to fulfill in order to be of practical relevance.

High recall

Section 2.2.1 emphasizes the importance of the trade-off decision between type I error and type II error in terms of the practical applicability of an AD model. Missed anomalies pose a serious threat in the discrete manufacturing domain, as they could result in unplanned downtimes or worse, defective parts that are passed to the customer undetected. This indicates that for this particular field of application, a type II error is more severe than a type I error, which translates into recall being more important than precision [Chen21f]. Thus, the first external requirement is that SSMSPC achieves a high recall.

Low false alarm rate

As has been mentioned in Section 2.1.1, it is comparatively straightforward to achieve perfect recall by simply flagging every instance as anomalous. However, this would lead to numerous false alarms that cause unnecessary distractions of the responsible personnel and lead to wasted time and resources [Wood17]. In fact, [Wood17] state that an excessive number of false alarms bears the risk that practitioners ignore the alarms raised by the employed monitoring system. Hence, in combination with a high recall, it is necessary that SSMSPC maintains a low false alarm rate, which corresponds to the second external requirement.

Interpretability

Recall from Section 2.3.1 that the detection of an out-of-control situation triggers a root-cause analysis by the responsible personnel to identify the assignable causes and bring the process back into a state of statistical control. In this regard, it has been argued that a control chart provides no effective support to help infer which elements of the process are considered anomalous. However, in order to be of practical value, [Jack91] state that any MSPC system should be able to provide an answer to the question of what the problem is when an anomalous process condition is signalled. From an ML perspective, this requires that the underlying model decisions are interpretable in order to understand the reasoning behind its decision-making process [Ruff21b]. With respect to multivariate time series data, this translates into highlighting the anomalous regions in the raw data that are considered anomalous [Abdu21; Li21b]. Thus, the third external requirement is that SSMSPC provides effective support for the root-cause analysis by highlighting the anomalous regions in the multivariate time series data.

Usability

Usability describes the "extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [ISO9241-11]. In order to present a suitable alternative for practical applications, a deployed SSMSPC system must possess a higher usability than univariate post-process SPC. This represents the fourth external requirement.
Table 4.1 summarizes the internal and external requirements.

Table 4.1.: Internal and external requirements, own illustration.

Internal requirements	External requirements
Machining process	High recall
Multivariate time series data associated with final quality characteristics	Low false alarm rate
Historical in-control dataset	Interpretability
	Usability

It must be re-emphasized that the external requirements derived above represent the minimum external requirements that SSMSPC has to fulfill to be of practical relevance. Depending on the discrete manufacturing process that is to be monitored, there might be additional external requirements imposed on SSMSPC, such as real-time capability.

4.3. Problem statement

As has been pointed out in the preceding paragraph, the concretized research objective of this thesis encompasses the development of SSMSPC, a novel approach for MSPC based on SSL that is capable to detect and locate anomalies in multivariate time series data. Taking into account the previously

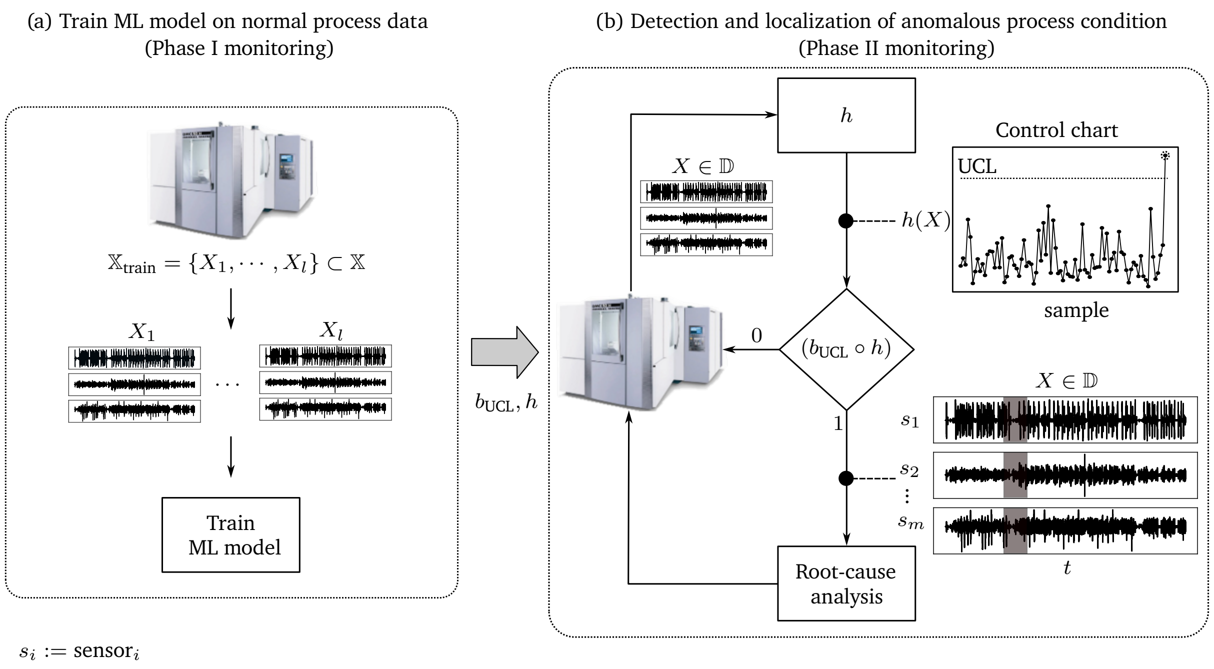

(a) Train ML model on normal process data
(Phase I monitoring)
$\mathbb{X}_{\text{train}} = \{X_1, \cdots, X_l\} \subset \mathbb{X}$
X_1
X_l
Train ML model
b_{UCL}, h
(b) Detection and localization of anomalous process condition
(Phase II monitoring)
h
$X \in \mathbb{D}$
$h(X)$
Control chart
UCL
sample
0
$(b_{\text{UCL}} \circ h)$
1
s_1
s_2
s_m
t
Root-cause analysis
$s_i :=$ sensor$_i$

Figure 4.1.: Problem statement for SSMSPC. (a) Phase I monitoring: Train an ML model to learn the mappings h and b_{UCL} based on $\mathbb{X}_{\text{train}} = \{X_1, \cdots, X_l\} \subset \mathbb{X}$. (b) Phase II monitoring: Continuously monitor new samples from the process $X \notin \mathbb{X}_{\text{train}}$ and plot $h(X)$ in the control chart. Once the control chart signals an anomalous process condition, the model needs to highlight the anomalous regions in the process data to support the responsible personnel in the root-cause analysis, own illustration adapted from [Bieg23].

presented internal and external requirements, the problem statement for SSMSPC can be formalized as follows:
Assume a machining process equipped with m sensors that record process data throughout the machining cycle of a part. Suppose the machining cycle of a part lasts n time steps. Let $\mathbb{D} \subset \mathbb{R}^{n \times m}$ be the set of all possible process data of this machining process. Thus, every observation $X \in \mathbb{D}$ represents the process data collected during the machining cycle of a specific part in the form of a multivariate time series. Let further $\mathbb{X} \subset \mathbb{D}$ represent the set of all normal process data originating from the machining process when it operates in statistical control.
Based on a training dataset $\mathbb{X}_{\text{train}} \subset \mathbb{X}$, the objective of SSMSPC is to learn a mapping $h_{\text{SSMSPC}} : \mathbb{D} \to \mathbb{R}$ such that for a given sample $X \in \mathbb{D}$, $h_{\text{SSMSPC}}(X)$ represents the anomaly score. In addition to h_{SSMSPC}, a UCL needs to be determined for practical applications that defines the corresponding threshold mapping $b_{\text{UCL}} : \mathbb{R} \to \{0, 1\}$, according to Equation 4.1

$$b_{\text{UCL}} \circ h_{\text{SSMSPC}}(X) = \begin{cases} 1, & h_{\text{SSMSPC}}(X) > \text{UCL} \\ 0, & \text{otherwise} \end{cases}. \tag{4.1}$$

After training, the task of SSMSPC is to provide a machine operator with the information whether the process data $X \in \mathbb{D}$ for the currently produced part originate from a statistically controlled process, i.e., $X \in \mathbb{X}$. In the case of an anomalous process condition, i.e., $X \notin \mathbb{X}$, the control chart must raise an alarm. Once an alarm has been raised, SSMSPC shall support the responsible personnel in the subsequent root-cause analysis by highlighting the anomalous regions in the raw time series data. Figure 4.1 depicts the general problem statement for which SSMSPC is designed. Note how this illustration shows the equivalence between the general SPC framework and the semi-supervised AD problem setting that is derived in Section 2.3.1.

4.4. Summary

Chapter 4 presents the overall objective of this thesis, which encompasses the development of SSMSPC, a novel approach for MSPC based on SSL to detect and locate anomalies in discrete manufacturing processes.
Section 4.1 concretizes the research objective and the research questions outlined in Section 1.1. Here, it is elaborated how the three central research questions are going to be answered in this thesis.
Section 4.2 presents the internal requirements that need to be met by a discrete manufacturing process to be eligible for SSMSPC, as well as the minimum external requirements that must be satisfied by SSMSPC to be of practical relevance. SSMSPC requires that the underlying discrete manufacturing process corresponds to a machining process in which process data are recorded throughout the production cycle of a part. These process data are assumed to be associated with the final quality characteristic of the produced part and have to be recorded in the form of a multivariate time series. Lastly, a historic in-control dataset is required for training purposes. In order to be of practical value, SSMSPC must achieve a high recall, while simultaneously maintaining a low false alarm rate. In addition to this, it is required that the model decisions are interpretable. Concretely, effective support for the root-cause analysis must be provided by highlighting the anomalous regions in the process data. Furthermore, it is argued that a deployed SSMSPC system must possess a higher usability compared to a conventional univariate post-process SPC system to be accepted as a suitable alternative.
Finally, Section 4.3 combines the results from the preceding two sections and formalizes the problem statement for SSMSPC.

5. Self-supervised multivariate statistical in-process control

This chapter introduces the proposed SSMSPC approach, which represents the central contribution of this thesis and provides the means to answer the first research question. Section 5.1 provides an overview of the three main components of SSMSPC and describes the underlying framework. In the following course of the chapter, the three components are elaborated in more detail. Specifically, Section 5.2 presents the details of the proposed pretext task. Section 5.3 provides necessary insights regarding the subsequent AD downstream task. Section 5.4 presents the control chart extension that enables SSMSPC to highlight anomalous regions in the process data. A concise summary of the most important findings concludes the underlying chapter. Note that parts of this chapter are adapted from [Bieg23].

5.1. Overview

As stated above, SSMSPC consists of three components: (1) a handcrafted pretext task that is referred to as Location + Transformation prediction, (2) an AD downstream task in which the learned representations are used to compute the Hotelling's T^2 statistic, and (3) a control chart extension to locate anomalies by highlighting the anomalous regions in the process data.

Recall the problem statement for SSMSPC that is depicted in Figure 4.1. With respect to the general SPC framework, the handcrafted pretext task and the AD downstream task are embedded in the model building phase, i.e., phase I, since both components are used to fit the control limit of a multivariate control chart. Conversely, the control chart extension is integrated in the subsequent monitoring phase, i.e., phase II, and is used in conjunction with the control chart obtained from phase I.

SSMSPC follows the two-stage framework by [Sohn21] that has been briefly introduced in Section 3.1 and shall be presented in more detail in the following.

As is shown in Figure 5.1, the key feature of the considered two-stage framework is to decouple the representation learning phase from the actual downstream task. Specifically, the first stage consists of solving a self-supervised pretext task based on a training dataset $\mathbb{X}_{\text{train}}$ that exclusively represents data from the normal class. For this purpose, an encoder network f maps the input data $\mathbf{x}$ to a lower dimensional representation $f(\mathbf{x})$. These representations are consequently fed through a projection head g, which, in case of a handcrafted pretext task that involves classifying instances, is typically represented by an MLP with a softmax output. Both the encoder f and the projection head g, are trained in an end-to-end fashion using the backpropagation algorithm presented in Section 2.1.2 to solve the respective pretext task. The second stage involves the AD downstream task in which a conventional AD model is trained based on the learned representations of the normal training data $f(\mathbb{X}_{\text{train}})$. Recall from Section 2.1.4 that the projection head g is discarded for the downstream task due to the empirically confirmed superiority of the learned representations that can be obtained from the encoder f.

The reason that SSMSPC follows the two-stage framework instead of deriving the anomaly score directly from the pretext task is threefold. First, it represents an approved framework that has been explicitly designed to be applied in the context of semi-supervised AD problem settings. Second, as has been pointed out in Section 3.1, decoupling the representation learning phase from the actual downstream task is shown to be more effective for AD, when compared to other approaches relying solely on the pretext task. Third, by separating the pretext task from the downstream task, it is comparatively straightforward to construct conventional multivariate control charts from the learned representations.

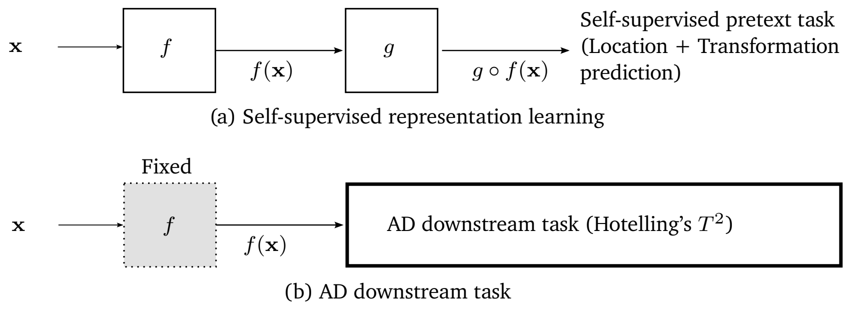

$\mathbf{x}$
f
$f(\mathbf{x})$
g
$g \circ f(\mathbf{x})$
Self-supervised pretext task (Location + Transformation prediction)
(a) Self-supervised representation learning
Fixed
$\mathbf{x}$
f
$f(\mathbf{x})$
AD downstream task (Hotelling's T^2)
(b) AD downstream task

Figure 5.1.: Two-stage framework for self-supervised AD according to [Sohn21]. (a) The first stage involves solving a self-supervised pretext task to learn effective representations. (b) Based on the learned representations, the second stage encompasses an AD downstream task, own illustration adapted from [Sohn21; Bieg23].

5.2. Pretext task

This section introduces the handcrafted Location + Transformation prediction pretext task, which embodies the first component of SSMSPC. In this context, Section 5.2.1 provides an overview of the individual pretext task components and clarifies the corresponding objectives. Following this, Section 5.2.2 gives insights into the proposed data augmentation functions. Lastly, Section 5.2.3 presents the applied encoder network in combination with the overall loss function.

5.2.1. Overview

The inspiration for the proposed Location + Transformation prediction pretext task originates from a simple thought experiment. This thought experiment centers around the objective of SSMSPC, which is to detect and locate anomalies in multivariate time series data based on a learned model of an in-control process.

A model that shall detect and locate anomalies needs to learn representations that enable it to identify how and where an anomalous time series sample deviates from the known in-control pattern. Evidently, one way to accomplish this would be with the help of a large amount of anomalous training data. However, as is pointed out in Section 2.2, anomalies represent rare events and as such cannot be acquired in large quantities. Now, the intuition is that if a model is trained to identify how and where an in-control sample has been altered by an artificially introduced anomaly, it can learn effective representations that generalize to anomalies occurring in real-world applications.

This intuition is incorporated into the Location + Transformation prediction pretext task. More precisely, assume the existence of a historic in-control dataset $\mathbb{X}_{\text{train}} \subset \mathbb{X} \subset \mathbb{R}^{n \times m}$ for some machining

process, as defined in Section 4.3. For a given $X \in \mathbb{X}_{\text{train}}$ that has been augmented by one of k augmentation functions in one of p equally-sized windows, the objective in the Location + Transformation prediction pretext task is to classify the augmentation and the window in which this augmentation has been applied in a multitask fashion.
The above objective can be broken down into three consecutive steps: (1) data augmentation, (2) scalogram computation, and (3) multitask multiclass classification, as shown in Figure 5.2.

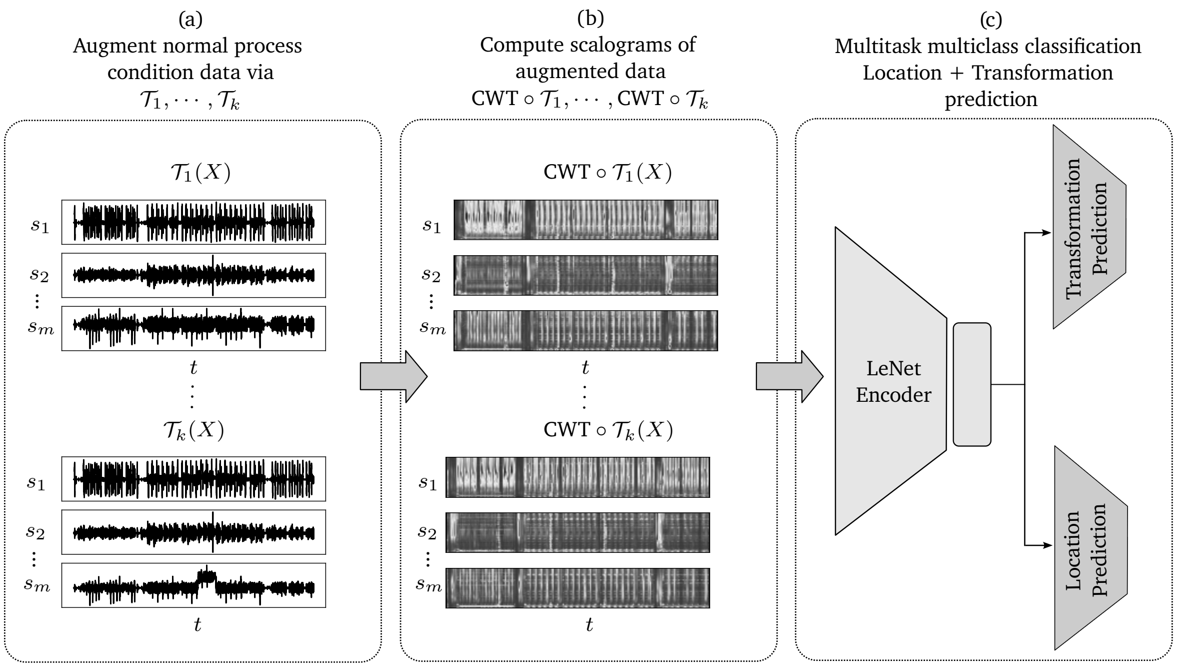

(a)
Augment normal process condition data via
$\mathcal{T}_1, \cdots, \mathcal{T}_k$
$\mathcal{T}_1(X)$
s_1
s_2
s_m
t
$\mathcal{T}_k(X)$
(b)
Compute scalograms of augmented data
CWT $\circ \mathcal{T}_1, \cdots,$ CWT $\circ \mathcal{T}_k$
CWT $\circ \mathcal{T}_1(X)$
CWT $\circ \mathcal{T}_k(X)$
(c)
Multitask multiclass classification
Location + Transformation prediction
LeNet Encoder
Transformation Prediction
Location Prediction

Figure 5.2.: Location + Transformation prediction pretext task proposed in SSMSPC. (a) Data augmentation: Augment normal process condition dataset with k predefined augmentation functions $\mathcal{T}_1, \cdots, \mathcal{T}_k$ in randomly selected windows $\mathcal{W}_1, \cdots, \mathcal{W}_p$. (b) Scalogram computation: Compute the CWT of each augmented training sample. (c) Multitask multiclass classification: Pass scalograms to a LeNet-type encoder network f with two projection heads attached to it. The objective is to classify both the type and the location of the applied augmentation, own illustration adapted from [Bieg23].

The first step in the Location + Transformation prediction pretext task involves data augmentation. For this purpose, $\mathbb{X}_{\text{train}}$ is augmented by a set of k different augmentation functions $\mathbb{T} := \{\mathcal{T}_1, \cdots, \mathcal{T}_k | \mathcal{T}_i : \mathbb{R}^{n \times m} \to \mathbb{R}^{n \times m}\}$. Specifically, each $\mathcal{T}_i \in \mathbb{T}$ augments the entire training set. This increases the number of available training samples for the pretext task by a factor of k. For a given $X \in \mathbb{X}_{\text{train}}$, the augmentation $\mathcal{T}_i$ is applied in one of p equally-sized windows $\mathcal{W}_j \in \{\mathcal{W}_1, \cdots, \mathcal{W}_p\}$, for all m sensors simultaneously.
In the second step, the augmented training data are transformed using CWT to obtain the respective scalogram representation. More precisely, for an augmented sample $\mathcal{T}_i(X)$, CWT is applied to each of the m features separately, and the individual scalograms are stacked on top of each other. Following this procedure results in a third-order tensor $(\text{CWT} \circ \mathcal{T}_i)(X) \in \mathbb{R}^{v \times n \times m}$ that can be interpreted as an m-channel image. Here, v corresponds to the number of scales of the corresponding mother wavelet. Recall from Section 2.1.3 that the rationale for using CWT in the data augmentation process of SSMSPC is twofold. First, it is capable of representing univariate time series data in a two-dimensional

time-scale representation that can be interpreted as an image and, as such, allows the application of state-of-the-art CNN architectures to process time series data. Second, it is especially useful for analyzing non-stationary signals that typically occur in manufacturing processes. Specifically, instead of other commonly used approaches such as STFT, CWT allows variable window sizes, leading to an arbitrarily good time resolution at high frequencies, and an arbitrarily good frequency resolution at low frequencies.
Transforming each sample $X \in \mathbb{X}_{\text{train}}$ in the described way yields the pretext dataset $\mathbb{X}_{\text{pretext}}$ with $|\mathbb{X}_{\text{pretext}}| = k \cdot |\mathbb{X}_{\text{train}}|$. For each $\mathbf{X} \in \mathbb{X}_{\text{pretext}}$, the corresponding label $\mathbf{y} \in \mathbb{Y}_{\text{pretext}}$ is represented by the augmentation function $\mathcal{T}_i$ and the window $\mathcal{W}_j$ in which it was applied. Algorithm 1 summarizes the preceding statements for the generation of $\mathbb{X}_{\text{pretext}}$ and $\mathbb{Y}_{\text{pretext}}$ in pseudocode.
The third step consists of training a LeNet-type encoder network f that is equipped with two projection heads g_{trans} and g_{loc} on the actual Location + Transformation prediction pretext task, using $\mathbb{X}_{\text{pretext}}$ and $\mathbb{Y}_{\text{pretext}}$. The projection heads g_{trans} and g_{loc} are represented by two MLPs whose architectures differ only in the softmax output layer. Concretely, the softmax output layer of g_{trans} is of size k, corresponding to the number of augmentation functions. Conversely, the softmax output layer of g_{loc} is of size $p + 1$, taking into account the p windows and considering the case when the applied augmentation corresponds to the identity function, and thus no window is chosen.

Algorithm 1 Pseudocode to generate the pretext dataset $\mathbb{X}_{\text{pretext}}$ and the set of pseudo labels $\mathbb{Y}_{\text{pretext}}$, own illustration adapted from [Bieg23].

```
Require: $\mathbb{T} = \{\mathcal{T}_1, \cdots, \mathcal{T}_k\}, \mathbb{X}_{\text{train}}, p, \text{CWT}$
 1: $\mathbb{X}_{\text{pretext}} \leftarrow \{\}$
 2: $\mathbb{Y}_{\text{pretext}} \leftarrow \{\}$
 3: Compute window bounds for $p$ windows $\{\mathcal{W}_1, \cdots, \mathcal{W}_p\}$
 4: for $i \in \{1, \cdots, k\}$ do
 5:     for $X \in \mathbb{X}_{\text{train}}$ do
 6:         Randomly choose a window $\mathcal{W}_j$
 7:         Augment $X$ in window $\mathcal{W}_j$ with $\mathcal{T}_i$
 8:         Compute scalogram $(\text{CWT} \circ \mathcal{T}_i)(X)$
 9:         Append $(\text{CWT} \circ \mathcal{T}_i)(X)$ to $\mathbb{X}_{\text{pretext}}$
10:         Append $i, j$ as pseudo label $\mathbf{y}$ to $\mathbb{Y}_{\text{pretext}}$
11:     end for
12: end for
13: Shuffle $\mathbb{X}_{\text{pretext}}, \mathbb{Y}_{\text{pretext}}$
14: return $\mathbb{X}_{\text{pretext}}, \mathbb{Y}_{\text{pretext}}$
```

5.2.2. Data augmentations

Recall from the presentation of the related self-supervised AD work in Section 3.1 that the data augmentation design strongly affects the outcome of the representation learning phase. However, as is shown in various research articles, such as [Gida18; Chen20; Li21a], it is generally difficult to anticipate upfront which augmentations result in effective representations. Thus, designing augmentation functions for a given AD task is characterized by a trial-and-error process. Specifically, based on the overall objective of what shall be accomplished with the data augmentation, e.g., replicating spatio-temporal irregularities, it is necessary to test a variety of different augmentation designs and choose the ones that lead to the best results.

With respect to SSMSPC, a total of $k = 4$ augmentation functions are proposed: (1) *Identity*, (2) *CutPaste*, (3) *MeanShift*, and (4) *MissingSignal*. Here, the augmentations (2) - (4) shall replicate contextual anomalies, as this specific anomaly type appears most often in time series data [Chan09; Agga17]. In the following, the individual augmentation functions are presented in more detail.

Identity

The first augmentation that is proposed in the context of SSMSPC is simply the identity function, i.e., $\mathcal{T}(X) = X$. Augmenting $\mathbb{X}_{\text{train}}$ with the identity function translates into incorporating the original in-control process data into the pretext dataset as one of the four classes. The intuition here is that the underlying model must learn to distinguish in-control data from artificially anomalous data in order to succeed on the pretext task. Note that the effectiveness of including the original train data into the pretext dataset has been demonstrated in several recent self-supervised AD approaches, e.g., [Gola18; Tack20].

CutPaste

The second augmentation is referred to as *CutPaste* and is inspired by the works of [Li21a]. As opposed to the original *CutPaste* augmentation that was designed for image data, the proposed *CutPaste* augmentation in SSMSPC is specifically designed for time series data.
In the first step, a cutting window $\mathcal{W}_c \in \{\mathcal{W}_1, \cdots, \mathcal{W}_p\}$ is chosen randomly. Following this, the cutting segment is determined by randomly selecting two points that mark the start and end points of the segment within the predefined bounds of $\mathcal{W}_c$. Having determined the cutting segment, the next step involves the random selection of the pasting window $\mathcal{W}_j$ from the p available windows. Lastly, the cutting segment is pasted to a random location within the pasting window. Note that for every $X \in \mathbb{X}_{\text{train}}$, the procedure above is applied to each of the m sensors simultaneously. Algorithm 2 summarizes the preceding statements in pseudocode for replicability. Figure 5.3 provides a visualization of the *CutPaste* augmentation using $p = 5$ windows.

Algorithm 2 Pseudocode for the *CutPaste* augmentation, own illustration adapted from [Bieg23].

```
Require: $\mathbb{X}_{\text{train}}, p,$ window bounds $b$
 1: for $X \in \mathbb{X}_{\text{train}}$ do
 2:     $\mathcal{W}_c \leftarrow$ sample cut window from $\mathcal{U}(1, p)$
 3:     $b_{\text{lower}}^{\mathcal{W}_c}, b_{\text{upper}}^{\mathcal{W}_c} \leftarrow b[\mathcal{W}_c]$
 4:     $c_1 \leftarrow$ sample from $\mathcal{U}(b_{\text{lower}}^{\mathcal{W}_c}, b_{\text{upper}}^{\mathcal{W}_c})$
 5:     $c_2 \leftarrow$ sample from $\mathcal{U}(b_{\text{lower}}^{\mathcal{W}_c}, b_{\text{upper}}^{\mathcal{W}_c})$
 6:     Ensure $c_2 > c_1$
 7:     cut_snippet $\leftarrow X[c_1 : c_2]$
 8:     $\Delta \leftarrow c_2 - c_1$
 9:     $\mathcal{W}_j \leftarrow$ sample paste window from $\mathcal{U}(1, p)$
10:     $b_{\text{lower}}^{\mathcal{W}_j}, b_{\text{upper}}^{\mathcal{W}_j} \leftarrow b[\mathcal{W}_j]$
11:     $p_1 \leftarrow$ sample from $\mathcal{U}(b_{\text{lower}}^{\mathcal{W}_j}, b_{\text{upper}}^{\mathcal{W}_j} - \Delta)$
12:     $p_2 \leftarrow p_1 + \Delta$
13:     $X[p_1 : p_2] \leftarrow$ cut_snippet
14: end for
15: return $X$
```

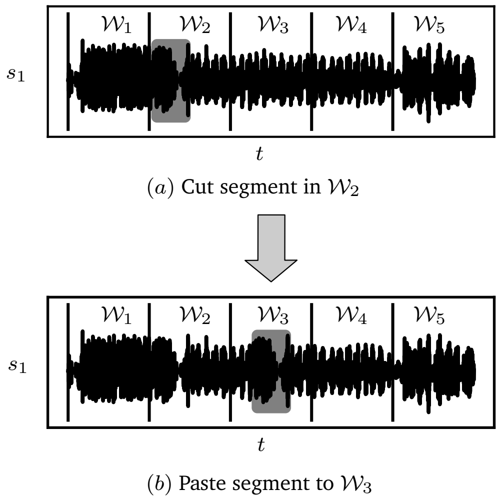


Figure 5.3.: *CutPaste* augmentation with $p = 5$ windows. (a) Cut a randomly sized segment from a randomly chosen cut window $\mathcal{W}_c \in \{\mathcal{W}_1, \cdots, \mathcal{W}_p\}$. (b) Paste segment to a random location within a randomly chosen paste window $\mathcal{W}_j$, own illustration adapted from [Bieg23].

MeanShift

The third augmentation used in SSMSPC is the *MeanShift* augmentation. This augmentation function is inspired by the fact that a contextual anomaly, independent of whether it is a contextual point or group anomaly, leads to changes in the absolute values of the sensor readings in the affected regions. Given a process sample $X \in \mathbb{X}_{\text{train}}$, the *MeanShift* augmentation strategy involves shifting a preselected time series segment by the mean of the entire time series.
First, a window $\mathcal{W}_j \in \{\mathcal{W}_1, \cdots, \mathcal{W}_p\}$ is randomly selected. Next, the shift segment is determined within $\mathcal{W}_j$. For this purpose, similar to *CutPaste*, two points, i.e., the start and end points of the segment, are chosen randomly within the bounds of $\mathcal{W}_j$. Having accomplished this, the mean for each of the m features is computed and added to every sample in the specified segment. Algorithm 3 provides the details for the proposed *MeanShift* augmentation in pseudocode. Figure 5.4 depicts how the *MeanShift* algorithm operates on an exemplary time series using $p = 5$ windows.

Algorithm 3 Pseudocode for the *MeanShift* augmentation, own illustration adapted from [Bieg23].

```
Require: $\mathbb{X}_{\text{train}}, p,$ window bounds $b$
 1: for $X \in \mathbb{X}_{\text{train}}$ do
 2:     $\mathcal{W}_j \leftarrow$ sample window from $\mathcal{U}(1, p)$
 3:     $b^{\mathcal{W}_j}_{\text{lower}}, b^{\mathcal{W}_j}_{\text{upper}} \leftarrow b[\mathcal{W}_j]$
 4:     $c_1 \leftarrow$ sample from $\mathcal{U}(b^{\mathcal{W}_j}_{\text{lower}}, b^{\mathcal{W}_j}_{\text{upper}})$
 5:     $c_2 \leftarrow$ sample from $\mathcal{U}(b^{\mathcal{W}_j}_{\text{lower}}, b^{\mathcal{W}_j}_{\text{upper}})$
 6:     Ensure $c_2 > c_1$
 7:     time_series_mean $\leftarrow \mu(X) \in \mathbb{R}^m$
 8:     $X[c_1 : c_2] \leftarrow X[c_1 : c_2] +$ time_series_mean
 9: end for
10: return $X$
```

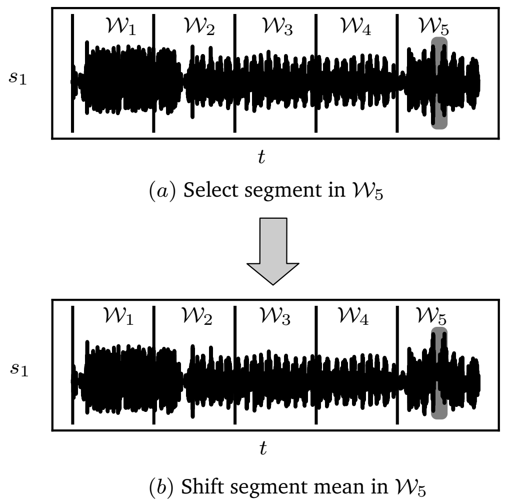


Figure 5.4.: *MeanShift* augmentation with $p = 5$ windows. (a) Select a randomly sized segment from a randomly chosen window $\mathcal{W}_j \in \{\mathcal{W}_1, \cdots, \mathcal{W}_p\}$. (b) Compute the mean of the entire time series for each of the m sensors and add the result to the selected segment, own illustration adapted from [Bieg23].

MissingSignal

The final augmentation that is proposed in the context of SSMSPC is the *MissingSignal* augmentation. This specific augmentation function aims to resemble a missing sensor signal. The routine for the *MissingSignal* augmentation is equivalent to the *MeanShift* augmentation, with one exception. Instead of computing the mean and adding it to the selected segment, the *MissingSignal* augmentation replaces the entire segment with a constant value that represents the first value of the original segment. Algorithm 4 displays the corresponding pseudocode for the preceding statements. Figure 5.5 visualizes the working details of the proposed *MissingSignal* augmentation for $p = 5$ windows.

Algorithm 4 Pseudocode for the *MissingSignal* augmentation, own illustration adapted from [Bieg23].

```
Require: $\mathbb{X}_{\text{train}}, p,$ window bounds $b$
 1: for $X \in \mathbb{X}_{\text{train}}$ do
 2:     $\mathcal{W}_j \leftarrow$ sample window from $\mathcal{U}(1, p)$
 3:     $b^{\mathcal{W}_j}_{\text{lower}}, b^{\mathcal{W}_j}_{\text{upper}} \leftarrow b[\mathcal{W}_j]$
 4:     $c_1 \leftarrow$ sample from $\mathcal{U}(b^{\mathcal{W}_j}_{\text{lower}}, b^{\mathcal{W}_j}_{\text{upper}})$
 5:     $c_2 \leftarrow$ sample from $\mathcal{U}(b^{\mathcal{W}_j}_{\text{lower}}, b^{\mathcal{W}_j}_{\text{upper}})$
 6:     Ensure $c_2 > c_1$
 7:     constant $\leftarrow X[c_1]$
 8:     $X[c_1 : c_2] \leftarrow$ constant
 9: end for
10: return $X$
```

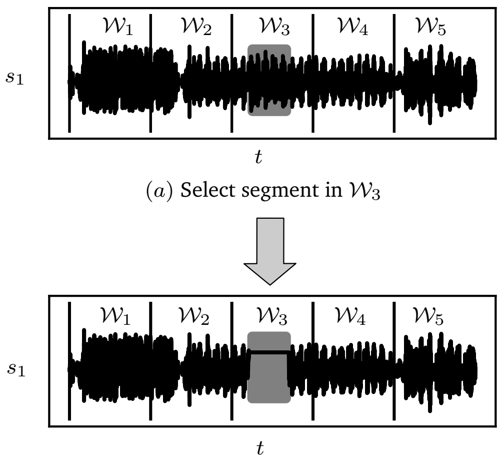


Figure 5.5.: *MissingSignal* augmentation with $p = 5$ windows. (a) Select a randomly sized segment from a randomly chosen window $\mathcal{W}_j \in \{\mathcal{W}_1, \cdots, \mathcal{W}_p\}$. (b) Replace the signal in the segment by a constant value, i.e., the first value of the original segment, own illustration adapted from [Bieg23].

Hyperparameters of the data augmentation process

The $k = 4$ data augmentation functions introduced above represent the default setting for the augmentation process in SSMSPC and are recommended for practical applications. As will be shown in Chapter 6, applying these augmentation functions results in effective representations that are useful to detect and locate anomalies in discrete manufacturing processes. Technically, the number of applied data augmentations k and the windows p in which these augmentations can be applied, represent hyperparameters of the data augmentation process. As such, even though SSMSPC comes with a default setting of $k = 4$ and $p = 5$, it is possible to embed novel augmentation functions and vary the number of windows in order to increase or decrease the difficulty of the Location + Transformation prediction pretext task.

5.2.3. Location + Transformation prediction

Location + Transformation prediction represents a multitask multiclass classification. It is composed of a transformation prediction task and a location prediction task in which the augmentation $\mathcal{T}_i$ and the window $\mathcal{W}_j$ of an augmented time series input $\mathbf{X} \in \mathbb{X}_{\text{pretext}}$ are to be classified, respectively. For this purpose, SSMSPC relies on a LeNet-type encoder network f as its default feature extractor and two projection heads g_{trans} and g_{loc}. Figure 5.6 provides an illustration of the Location + Transformation prediction objective.

LeNet-type encoder network

The LeNet-type encoder network consists of three convolutional modules and a two-layer MLP. Each convolutional module consists of a convolutional layer using $8 \times (7 \times 7)$, $16 \times (7 \times 7)$ and $32 \times (7 \times 7)$ kernels in the first, second, and third modules, respectively. The convolutional layers are followed by a batch normalization layer, a leaky ReLU activation, and a subsequent max-pooling layer. The two-layer MLP contains 32 and 16 units that are followed by a batch normalization layer and leaky

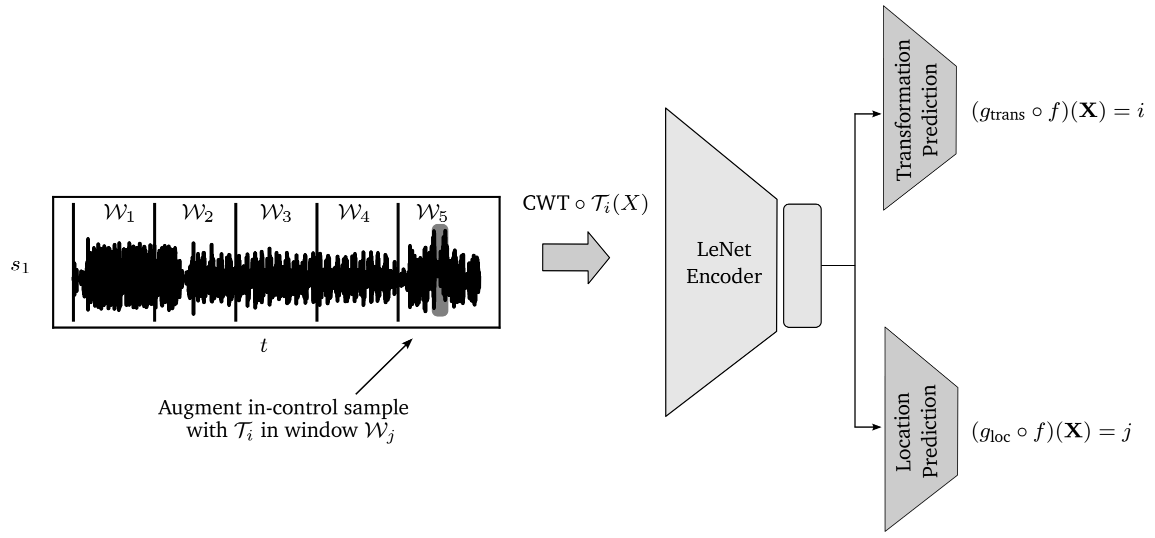

$\mathcal{W}_1$
$\mathcal{W}_2$
$\mathcal{W}_3$
$\mathcal{W}_4$
$\mathcal{W}_5$
s_1
t
Augment in-control sample with $\mathcal{T}_i$ in window $\mathcal{W}_j$
CWT $\circ\, \mathcal{T}_i(X)$
LeNet Encoder
Transformation Prediction
$(g_{\text{trans}} \circ f)(\mathbf{X}) = i$
Location Prediction
$(g_{\text{loc}} \circ f)(\mathbf{X}) = j$

Figure 5.6.: Location + Transformation prediction objective. For a given $\mathbf{X} \in \mathbb{X}_{\text{pretext}}$ that has been augmented by $\mathcal{T}_i$ in window $\mathcal{W}_j$, the task of g_{trans} is to classify i, while the task of g_{loc} is to classify j, own illustration.

ReLU activations, respectively. This choice is inspired by the works of [Ruff18; Lizn21; Ruff21b] and leverages two properties of the original LeNet, as presented in Section 2.1.3: (1) its simplicity and (2) its low capacity. As has been stated in Section 3.3, due to limiting factors in discrete manufacturing processes, such as excessive processing times, the collection of large quantities of in-control data can become challenging. Thus, compared to other domains, the expected amount of available data for training tends to be lower, which leads to an increased risk of overfitting and thus requires a model with lower capacity. In addition, the simplicity in terms of computational steps leads to a shorter inference time, which is beneficial for in-process monitoring. Nevertheless, it is worth highlighting that SSMSPC is flexible in terms of the employed feature extractor. In fact, any CNN architecture that allows to process image data in the previously specified shape is applicable in this context.

Transformation prediction loss

The objective in transformation prediction is to predict the applied augmentation $\mathcal{T}_i \in \mathbb{T}$. In order to accomplish this, a projection head g_{trans} is employed that is represented by a simple three-layer MLP. The first and second layers are equipped with 16 and 8 units, respectively, that are followed by a batch normalization layer and leaky ReLU activation. The output layer consists of k units and a subsequent softmax activation. In terms of the loss function, a simple CE loss function is employed with Equation 5.1

$$\mathcal{L}_{\text{trans}} = -\sum_{i \in \{1, \cdots, k\}} y_i \log((g_{\text{trans}} \circ f)(\mathbf{X})), \quad \mathbf{X} \in \mathbb{X}_{\text{pretext}}. \tag{5.1}$$

Location prediction loss

The objective in location prediction is to predict the window $\mathcal{W}_j \in \{\mathcal{W}_1, \cdots, \mathcal{W}_p\}$ in which the augmentation $\mathcal{T}_i$ has been applied. For this purpose, the projection head g_{loc} is employed. As stated above, the architecture of g_{loc} is equivalent to g_{trans}, differing only in the last layer, which is equipped with $p+1$ units. Similar to transformation prediction, the employed loss function for the location prediction task is the CE loss, as shown in Equation 5.2

$$\mathcal{L}_{\text{loc}} = -\sum_{i \in \{1, \cdots, p+1\}} y_i \log((g_{\text{loc}} \circ f)(\mathbf{X})), \quad \mathbf{X} \in \mathbb{X}_{\text{pretext}}. \tag{5.2}$$

Location + Transformation prediction loss

Combining the transformation prediction loss and the location prediction loss in a linear combination leads to the overall loss function for the Location + Transformation prediction pretext task, which is presented in Equation 5.3

$$\mathcal{L}_{\text{loc+trans}} = \lambda_1 \mathcal{L}_{\text{trans}} + \lambda_2 \mathcal{L}_{\text{loc}}. \tag{5.3}$$

Here, λ_1, λ_2 represent scaling parameters that allow to vary the influence of either task on the overall objective. The default setting is $\lambda_1, \lambda_2 = 1$. Note that the above procedure of combining multiple loss functions into a single overall loss function is common practice as shown, e.g., in [Tack20].

5.3. Downstream task

Having discussed the Location + Transformation prediction pretext task, this section presents the AD downstream task, which marks the second component of SSMSPC. As stated at the beginning of the chapter, this stage involves computing the Hotelling's T statistic as the anomaly score based on the learned representations. Figure 5.7 depicts the individual steps in the proposed AD downstream task of SSMSPC.

Instead of augmenting the in-control data with a set of predefined augmentation functions, the downstream task relies solely on the raw in-control data, i.e., $\mathbb{X}_{\text{train}}$. In order to extract the learned representations, the in-control data $\mathbb{X}_{\text{train}}$ are first transformed into their respective scalogram representation with the help of CWT. Here, CWT is applied with the same settings as in the pretext task, i.e., the number of scales v remains unchanged, and the scalograms are computed for each of the m sensors separately and stacked on top of each other. Thus, for each $X \in \mathbb{X}_{\text{train}}$, $\tilde{\mathbf{X}} = \text{CWT}(X) \in \mathbb{R}^{v \times n \times m}$, represents the corresponding scalogram representation. The set of in-control scalogram representations will be denoted as $\tilde{\mathbb{X}}_{\text{train}}$.

Following this, the next step involves passing the scalogram representations through the pretrained LeNet-type encoder network f. The learned representations for a given $\tilde{\mathbf{X}} \in \tilde{\mathbb{X}}_{\text{train}}$ are consequently represented by $f(\tilde{\mathbf{X}}) \in \mathbb{R}^{r \times 1}$, where r corresponds to the number of units in the last layer of the LeNet-type encoder network. Recall that the projection heads g_{trans} and g_{loc} are discarded at this stage.

Once the learned representations are extracted, the anomaly score for SSMSPC, i.e., h_{SSMSPC} is given by the Hotelling's T^2 statistic. Concretely, in a first step, the mean $\hat{\mu} \in \mathbb{R}^{r \times 1}$ and the inverse of the covariance matrix $\hat{\Sigma}^{-1} \in \mathbb{R}^{r \times r}$ are estimated based on $f(\tilde{\mathbb{X}}_{\text{train}})$. Next, the Hotelling's T^2 statistic will be computed for every $\tilde{\mathbf{X}} \in \tilde{\mathbb{X}}_{\text{train}}$, according to Equation 5.4

$$h_{\text{SSMSPC}}(X) := T^2 = (f(\tilde{\mathbf{X}}) - \hat{\mu})^T \hat{\Sigma}^{-1} (f(\tilde{\mathbf{X}}) - \hat{\mu}) \in \mathbb{R}. \tag{5.4}$$

Recall from Section 4.3 that aside from the anomaly scoring function h_{SSMSPC}, it is necessary to determine the threshold mapping b_{UCL}, which provides a binary output based on an UCL, stating whether the process data $X \in \mathbb{D}$ of the currently produced part correspond to an in-control process. For this purpose, the procedure presented in Section 2.2.1 is followed. Specifically, as is shown in

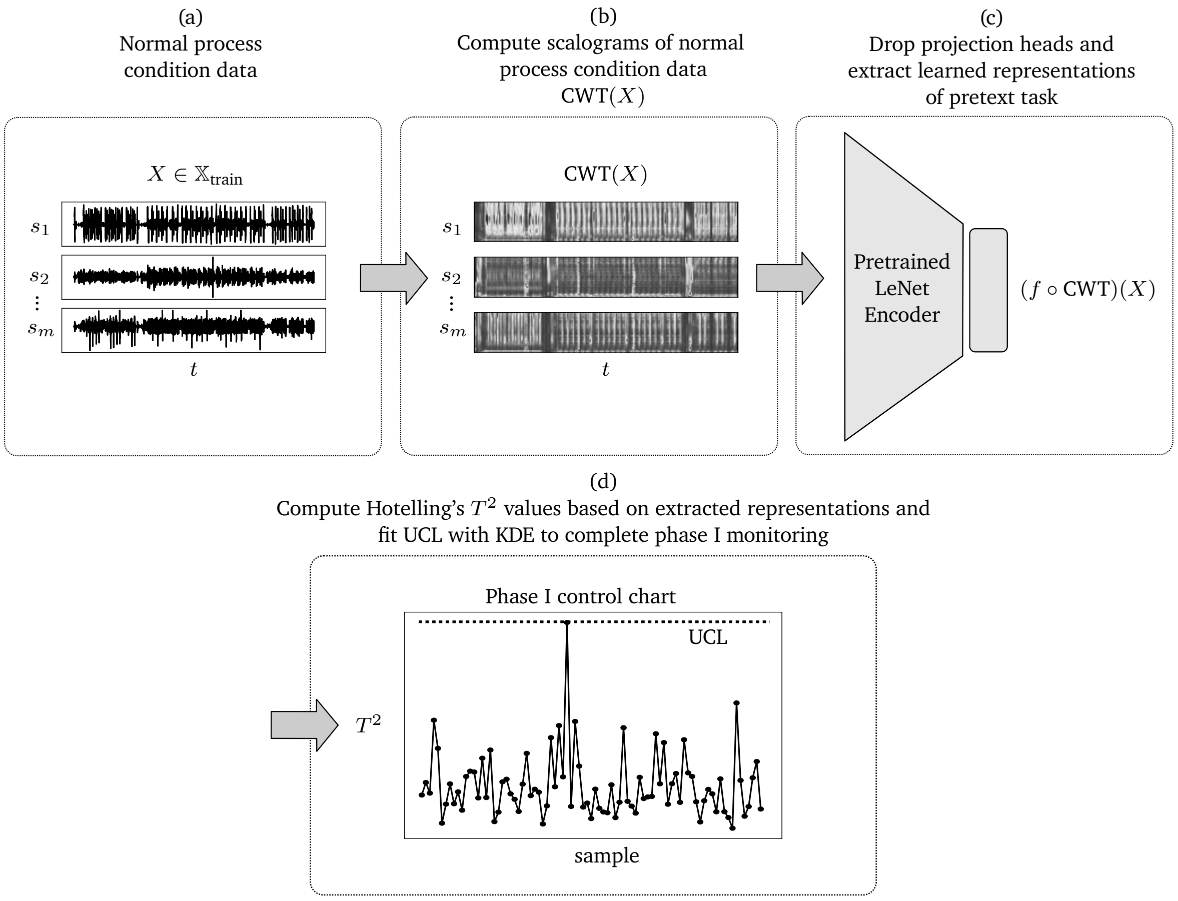

(a)
Normal process condition data
$X \in \mathbb{X}_{\text{train}}$
s_1
s_2
s_m
t
(b)
Compute scalograms of normal process condition data CWT(X)
CWT(X)
s_1
s_2
s_m
t
(c)
Drop projection heads and extract learned representations of pretext task
Pretrained LeNet Encoder
$(f \circ \text{CWT})(X)$
(d)
Compute Hotelling's T^2 values based on extracted representations and fit UCL with KDE to complete phase I monitoring
Phase I control chart
UCL
T^2
sample

Figure 5.7.: AD downstream task proposed in SSMSPC. (a) In-control dataset $\mathbb{X}_{\text{train}}$. (b) Compute scalograms of in-control data $\tilde{\mathbf{X}} = \text{CWT}(X)$. (c) Pass scalograms through the pretrained LeNet-type encoder network f from the pretext task to obtain learned representations of in-control data. (d) Compute Hotelling's T^2 statistic as the anomaly score based on $f(\tilde{\mathbf{X}})$ and estimate the corresponding PDF via KDE to fit control limits, own illustration adapted from [Bieg23].

Equation 5.5, the true PDF θ_{SSMSPC} of the corresponding anomaly scores is approximated with KDE using a Gaussian kernel

$$\hat{\theta}_{h_{\text{SSMSPC}}}(h_{\text{SSMSPC}}(X)) = \frac{1}{l\nu}\sum_{i=1}^{l} k\left(\frac{h_{\text{SSMSPC}}(X) - h_{\text{SSMSPC}}(X_i)}{\nu}\right). \tag{5.5}$$

Based on the approximated PDF $\hat{\theta}_{h_{\text{SSMSPC}}}$, the quantile function $\hat{\Theta}^{-1}_{\text{SSMSPC}}$ will be determined to obtain the UCL for a given significance level α, following Equation 5.6

$$\text{UCL} = \hat{\Theta}^{-1}_{\text{SSMSPC}}(\xi) = \hat{\Theta}^{-1}_{\text{SSMSPC}}(1-\alpha). \tag{5.6}$$

The determination of the UCL defines the threshold mapping b_{UCL}, which completes the AD downstream task.

5.4. Control chart extension

This section introduces the control chart extension that represents the third component of SSMSPC. Section 5.4.1 provides a general overview of the proposed component in order to clarify the underlying idea. Following this, Section 5.4.2 gives insights into the anomaly localization process using Grad-CAM based on the last convolutional layer of the LeNet-type encoder network.

5.4.1. Overview

As stated before, the control chart extension is integrated into the actual process monitoring phase, i.e., phase II, and is used in conjunction with the Hotelling's T^2 chart obtained from phase I. Specifically, once an anomalous process condition has been detected, the control chart extension provides an additional visualization in which the anomalous regions in the process data are highlighted. As such, it addresses the interpretability requirement that was derived in Section 4.2.2.
In addition to highlighting anomalies in the process data, SSMSPC suggests the incorporation of metadata that are collected during the processing cycle to further enhance the support for the root-cause analysis. A machining process typically consists of a set of q process modes $\{\mathcal{M}_1, \cdots, \mathcal{M}_q\}$. Each mode $\mathcal{M}_i$, corresponds to a specific processing step in the processing cycle of a part. More precisely, each processing step is characterized by a different tool that is used to machine the part. Collecting metadata such as the current Numerical Control (NC) line of the machine program, allows to segment the raw time series data into the individual process modes. This supports, e.g., a machine operator to directly pinpoint the process step in which the anomaly occurred. Thus, augmenting the visualization of the anomalous regions in the process data with metadata provides a simple measure that can greatly enhance the interpretability of the model output. Figure 5.8 provides an illustration of the proposed control chart extension.

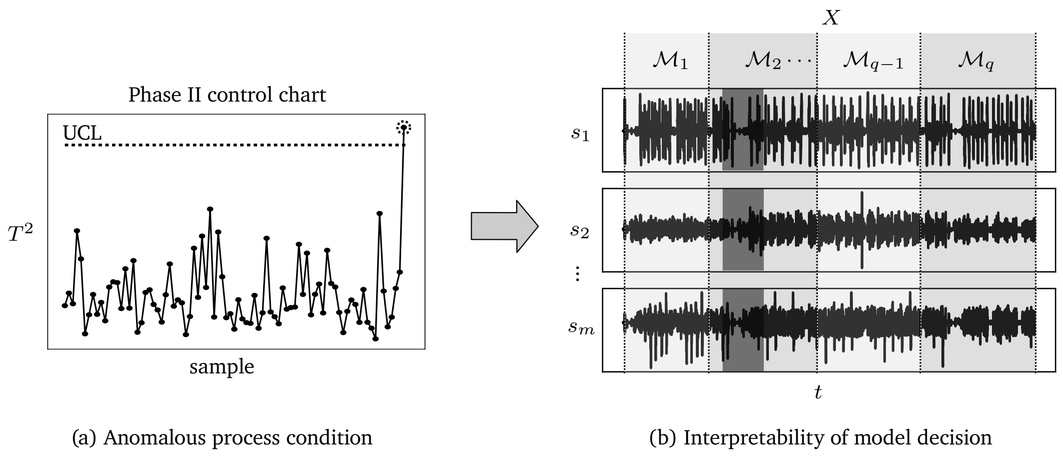


Figure 5.8.: Proposed control chart extension in SSMSPC. (a) An assignable cause has affected the process and the control chart raises an alarm. (b) An additional view is presented in which the multivariate time series data X are segmented into the individual process modes $\{\mathcal{M}_1, \cdots, \mathcal{M}_q\}$ with the help of metadata and the anomalous regions are highlighted, own illustration adapted from [Bieg23].

5.4.2. Highlighting anomalies

In order to highlight anomalies in the process data, SSMSPC builds on the Grad-CAM approach that is presented in Section 2.1.3. The reason for this is mainly motivated by the fact that it represents a simple, and at the same time, powerful technique to visually interpret CNN decisions.
Let $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times 32}$ denote the feature maps of the last convolutional layer of the LeNet-type encoder network f. Assume further that a given process sample $X \in \mathbb{D}$ is considered anomalous, i.e., $h_{\text{SSMSPC}}(X) > \text{UCL}$. Recall that the overall objective of Grad-CAM is to obtain the Grad-CAM heatmap G_{SSMSPC} for a specific CNN decision. To accomplish this with respect to SSMSPC, the first step consists of computing the importance weights $\zeta_1, \cdots, \zeta_{32}$ for $\mathbf{A}$. This is achieved by computing the gradients of $h_{\text{SSMSPC}}(X)$ with respect to $\mathbf{A}$ and applying the global average-pooling operation on each feature map, respectively, as displayed in Equation 5.7

$$\zeta_i = \frac{1}{n_1 n_2} \sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \frac{\partial h_{\text{SSMSPC}}(X)}{\partial a_{u,v,i}} \ \forall i \in \{1, \cdots, 32\}. \tag{5.7}$$

Following this, the Grad-CAM heatmap is given by the linear combination between the importance weights and the feature maps that is passed to a ReLU activation, according to Equation 5.8

$$G_{\text{SSMSPC}} = \text{ReLU}\left(\sum_{i=1}^{32} \zeta_i A_i \right). \tag{5.8}$$

Having obtained G_{SSMSPC}, SSMSPC determines the anomalous time steps in three consecutive steps. First, $G_{\text{SSMSPC}} \in \mathbb{R}^{n_1 \times n_2}$ is resized to $\widetilde{G}_{\text{SSMSPC}} \in \mathbb{R}^{v \times n}$, which represents the original height and width of the scalogram representation of X, i.e., v corresponds to the number of scales of the mother wavelet and n represents the number of time steps. Second, a threshold mapping $b_\delta : \mathbb{R}^{v \times n} \rightarrow \mathbb{R}^{v \times n}$ is defined with Equation 5.9

$$b_\delta(\widetilde{G}_{\text{SSMSPC}}) = \begin{cases} g_{i,j}, & g_{i,j} > \delta \\ 0, & \text{otherwise} \end{cases} \ \forall i \in \{1, \cdots, v\},\ \forall j \in \{1, \cdots, n\}. \tag{5.9}$$

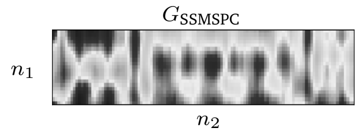
For this purpose, the KDE-based threshold selection scheme with a Gaussian kernel is employed again. Specifically, the threshold δ is determined by evaluating the quantile function $\hat{\Theta}_{\widetilde{G}_{\text{SSMSPC}}}$ at a given ξ-quantile, with the default choice being $\xi = 0.99$. Third, the columns of $b_\delta(\widetilde{G}_{\text{SSMSPC}})$ are summed up, and all nonzero values are set to one, as shown in Equation 5.10

$$\mathbb{1}_{\sum_i g_{i,j} > 0} \left(\sum_i g_{i,j} \right) \ \forall j \in \{1, \cdots, n\}. \tag{5.10}$$

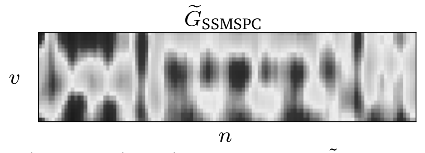
This results in a binary value for each of the n time steps, indicating whether it is considered normal or anomalous. Figure 5.9 visualizes the preceding statements.

5.5. Summary

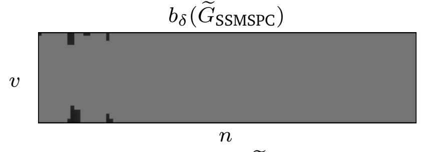
Chapter 5 introduces SSMSPC, a novel approach for MSPC based on SSL, which represents the main scientific contribution of this thesis. Section 5.1 provides an overview of the three components that SSMSPC consists of: (1) Location + Transformation prediction pretext task, (2) AD downstream task,



(a) Grad-CAM Heatmap $G_{\text{SSMSPC}} \in \mathbb{R}^{n_1 \times n_2}$



(b) Resized Grad-CAM Heatmap $\widetilde{G}_{\text{SSMSPC}} \in \mathbb{R}^{v \times n}$



(c) Thresholding $\widetilde{G}_{\text{SSMSPC}}$ via b_δ

t_1	t_2	t_3	t_4	t_5	t_6	$\cdots$	t_n
0	0	0	1	1	1	$\cdots$	0

(d) Anomalous time steps

Figure 5.9.: Highlighting anomalous regions in process data based on Grad-CAM. (a) Compute Grad-CAM heatmap G_{SSMSPC}. (b) Resize G_{SSMSPC} to the original scalogram input dimension $\widetilde{G}_{\text{SSMSPC}}$. (c) Define b_δ and apply thresholding $b_\delta(\widetilde{G}_{\text{SSMSPC}})$. (d) Sum up columns of $b_\delta(\widetilde{G}_{\text{SSMSPC}})$ and set nonzero values to one to obtain a binary value for each time step indicating whether it is normal or anomalous, own illustration adapted from [Bieg23].

and (3) control chart extension. Both the Location + Transformation prediction pretext task and the AD downstream task are embedded in phase I, while the control chart extension is incorporated into phase II of the general SPC framework. Next, it is shown that SSMSPC follows the two-stage framework by [Sohn21], as it provides the means to effectively combine SSL with MSPC by decoupling the representation learning phase from the AD downstream task.
Section 5.2 provides details regarding the proposed Location + Transformation prediction pretext task. The underlying objective of this pretext task is to classify the augmentation $\mathcal{T}_i$ and the window $\mathcal{W}_j$ based on a time series sample that has been artificially augmented by one of k augmentation functions in one of p windows. In this context, $k = 4$ augmentation functions are proposed: (1) *Identity*, (2) *CutPaste*, (3) *MeanShift*, and (4) *MissingSignal*, which represent the default setting for SSMSPC. It is shown how these augmentations are used in conjunction with the CWT to generate the pretext dataset $\mathbb{X}_{\text{pretext}}$ and the pseudo labels $\mathbb{Y}_{\text{pretext}}$ from an in-control dataset $\mathbb{X}_{\text{train}}$, increasing the number of training samples for the pretext task by a factor of $k = 4$. The number of augmentation functions k and the number of windows p represent two important hyperparameters. These allow to include additional augmentation functions and to change the number of windows to vary the difficulty of the pretext task. Next, emphasis is placed on the proposed default model architecture, which consists of a LeNet-type encoder network f and two projection heads g_{trans} and g_{loc} to solve the Location + Transformation prediction pretext task. The LeNet-type encoder network is employed as a feature

extractor, while the projection heads g_{trans} and g_{loc} are used to classify the augmentation $\mathcal{T}_i$ and the window $\mathcal{W}_j$, respectively. It is emphasized that SSMSPC is flexible in terms of its encoder choice, allowing the use of any CNN architecture that can process the specified shape of the input data. In order to train the proposed default architecture, the overall Location + Transformation prediction loss function corresponds to the linear combination of the individual CE losses obtained from the transformation prediction and location prediction tasks using scaling parameters λ_1, λ_2, respectively. Section 5.3 gives insights into the specifics of the AD downstream task. Based on the scalogram representation of the in-control dataset $\tilde{\mathbb{X}}_{\text{train}}$, the pretrained LeNet-type encoder network f is used to extract the learned representations, while the projection heads g_{trans} and g_{loc} are discarded. The anomaly score h_{SSMSPC} corresponds to the Hotelling's T^2 statistic that is computed using the learned representations $f(\tilde{\mathbb{X}}_{train})$. In order to determine the UCL and eventually derive the threshold mapping b_{UCL}, SSMSPC relies on the KDE-based threshold selection scheme applied to the anomaly scores.
Section 5.4 focuses on the control chart extension. This feature provides an additional view to the control chart, in which the anomalous regions in the process data are highlighted. In order to further enhance the interpretability of this additional view, SSMSPC proposes to use metadata that are collected during the process such as the current NC line of the machine program, to segment the time series data into the individual process modes. Doing so points the responsible personnel directly to the process steps that are affected by an assignable cause. Following this, the underlying algorithmic procedure that enables SSMSPC to identify the anomalous time steps is presented in more detail. Specifically, based on the resized Grad-CAM heatmap $\widetilde{G}_{\text{SSMSPC}}$, a threshold mapping b_δ is defined with the help of KDE that sets all entries below the threshold to zero. Next, the individual columns of $b_\delta(\widetilde{G}_{\text{SSMSPC}})$ are summed up and all nonzero entries are set to one, resulting in a binary value for each time step that indicates whether it is considered normal or anomalous.

6. Validation

The following chapter presents the validation of SSMSPC and provides the means to answer the second and third research questions. In Section 6.1, the performance of SSMSPC is evaluated and compared to state-of-the-art AD baselines using two real-world discrete manufacturing datasets, namely the Bosch CNC milling dataset, and the Center for Industrial Productivity Discrete Manufacturing Dataset (CiP-DMD). Section 6.2 compares the usability of SSMSPC as a deployed monitoring system with the conventional univariate post-process SPC using a real-world CNC milling process with human machine operators. The chapter concludes with a summary of the main findings. Note that parts of this chapter are adapted from [Bieg23; Niek23].

6.1. Validation I - performance comparison

This section presents the performance comparison between SSMSPC and state-of-the-art AD baselines. Section 6.1.1 addresses the evaluation on the Bosch CNC milling dataset, while Section 6.1.2 focuses on the CiP-DMD. The subsequent experiments are conducted on an AMD Ryzen 9 3900X processor with 3.8 GHz, 12 cores, 24 threads, and a GeForce RTX 2080 Ti GPU. For implementation purposes, the programming language Python is used in combination with the TensorFlow and scikit-learn libraries.

6.1.1. Bosch CNC milling dataset

The Bosch CNC milling dataset by [Tnan22] is a recently published discrete manufacturing benchmark dataset. It features vibration data collected with a triaxial accelerometer at a sampling rate of 2kHz from three different CNC milling machines over a three-year period. These CNC machines process different aluminum parts using a total of 15 tool operations, which run on all three machines and are named OP00 to OP14. The OPs represent process modes that correspond to distinct tools operating with unique process parameters. For each machine and each OP, the respective vibration data are labelled "good" or "bad", depending on whether they represent a normal or anomalous process condition. Approximately 96% of the available data account for the normal class, highlighting the typical imbalance of normal and anomalous samples encountered in AD problems. The overall structure of the dataset is displayed in Table 6.1.

In order to be consistent with the problem statement formulated in Section 4.3, the dataset, as described above, requires some modifications. First, as shown in the last column of Table 6.1, the data from the individual machines are merged OP-wise. Thus, the data are treated as if originating from a single machine. The purpose of this is to increase the dataset size for the subsequent analysis. Second, the available data within a given OP class are cropped to have the same length. To do so, the sample with the minimum length in an OP class serves as reference. Specifically, for each sample in a given OP class, the data points exceeding the minimum length are simply discarded. Third, the data from OP01, OP02, and OP07 are concatenated horizontally in order to emulate a process with three different modes. The rationale for selecting these OPs is that they result in the largest possible dataset for a process with three modes. Since the number of samples within the selected OPs is not

Table 6.1.: Structure of the Bosch CNC milling dataset, adapted from [Tnan22; Bieg23].

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

equal (#OP01 = 143, #OP02 = 152, #OP07 = 158), the OP with the minimum number of samples, i.e., OP01, is used as the reference, and the samples of OP02 and OP07 that exceed this number are discarded. Note that only samples corresponding to the normal class are discarded in this step. This results in a novel dataset $\mathbb{D}_{\text{Bosch}} \subset \mathbb{R}^{136084\times3}$, with $|\mathbb{D}_{\text{Bosch}}| = 143$. Fourth, the individual samples of $\mathbb{D}_{\text{Bosch}}$ are relabelled. Specifically, for $X \in \mathbb{D}_{\text{Bosch}}$, the label will be assigned to $y = 1$ whenever the original label of one of the three modes in X is anomalous. Conversely, the label of X will be assigned to $y = 0$ when the original labels of the modes correspond exclusively to the normal class. Following this labelling step, $\mathbb{D}_{\text{Bosch}}$ is partitioned into a train, validation, and test set using $60\%, 10\%$, and 30% of the data, respectively. Note that the training dataset contains only samples of the normal class. Table 6.2 presents the number of samples within each split of $\mathbb{D}_{\text{Bosch}}$. Completing this step marks the starting point for the subsequent performance comparison between SSMSPC and state-of-the-art AD baselines.

Table 6.2.: Train, validation, and test split of $\mathbb{D}_{\text{Bosch}}$ including the number of normal and anomalous samples, adapted from [Bieg23].

Split	Normal	Anomalous	Total
$\mathbb{X}^{\text{Bosch}}_{\text{train}}$	85	–	85
$\mathbb{X}^{\text{Bosch}}_{\text{val}}$	10	5	15
$\mathbb{X}^{\text{Bosch}}_{\text{test}}$	30	13	43

SSMSPC - pretext task

The first step in the application of SSMSPC encompasses the Location + Transformation prediction pretext task, as described in Section 5.2. For this purpose, the in-control dataset $\mathbb{X}^{\text{Bosch}}_{\text{train}}$ is augmented

with the proposed $k = 4$ data augmentations: *Identity*, *CutPaste*, *MeanShift*, and *MissingSignal* using $p = 5$ windows. Next, CWT is applied to the augmented time series data with $v = 128$ scales using the Morlet wavelet to retrieve the required scalogram representation. After this, the scalograms are resized down to a size of 128×512 to reduce computational efforts, and min-max scaling is applied. Conducting the aforementioned steps results in the pretext dataset $\mathbb{X}^{\text{Bosch}}_{\text{pretext}} \subset \mathbb{R}^{128 \times 512 \times 3}$, where $|\mathbb{X}^{\text{Bosch}}_{\text{pretext}}| = 4 \cdot |\mathbb{X}^{\text{Bosch}}_{\text{train}}| = 340$.
Next, the proposed model architecture, i.e., the LeNet-type encoder network f in combination with the two projection heads g_{trans} and g_{loc}, as described in Section 5.2.3, is trained on the Location + Transformation prediction objective. Table 6.3 displays the individual layers and the number of parameters of the proposed model architecture for the given input shape $128 \times 512 \times 3$.

Table 6.3.: SSMSPC model architecture used for the Location + Transformation prediction pretext task, adapted from [Bieg23].

	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

The model is trained for 20 epochs using a batch size of eight. The leaky ReLU activations use a leakiness of $\beta = 10^{-1}$. In addition to that, a cosine learning rate schedule is applied with an initial learning rate of $\gamma = 10^{-4}$. Regarding gradient-based optimization, the ADAM optimizer is employed and equipped with a weight decay of $\lambda = 10^{-3}$ for regularization purposes.

SSMSPC - AD downstream task

Having completed the representation learning phase, the second step in the application of SSMSPC embodies the AD downstream task, see Section 5.3. In the beginning, the scalogram representation of $\mathbb{D}_{\text{Bosch}}$ is obtained by applying the CWT using the settings from the pretext task, i.e., $v = 128$ scales and Morlet wavelet, on the individual splits. This dataset is denoted as $\tilde{\mathbb{D}}_{\text{Bosch}}$. The next step involves resizing the scalograms to 128×512 pixels and applying a min-max feature scaling based on the train split $\tilde{\mathbb{X}}_{\text{train}}^{\text{Bosch}}$.
The scalograms are then fed through the pretrained LeNet-type encoder network f in order to extract the learned representations $f(\tilde{\mathbb{X}}_{\text{train}}^{\text{Bosch}})$. Based on the learned representations, the required parameters $\hat{\mu} \in \mathbb{R}^{16\times 1}$ and $\hat{\Sigma}^{-1} \in \mathbb{R}^{16\times 16}$ are estimated in order to compute the anomaly score h_{SSMSPC}, which is represented by the Hotelling's T^2 statistic. Table 6.4 displays the model architecture for the AD downstream task.

Table 6.4.: SSMSPC model architecture used for the AD downstream task, adapted from [Bieg23].

	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
h_{SSMSPC}	Hotelling's T^2	(None, 1,1)	0

The threshold mapping b_{UCL} is obtained via KDE using the default significance value of $\alpha = 0.01$ to ensure a low false alarm rate. All hyperparameters are selected according to their performance on $\tilde{\mathbb{X}}_{\text{val}}^{\text{Bosch}}$ with the grid search procedure. This completes phase I of the general SPC framework.

State-of-the-art baselines

The performance of SSMSPC is compared to both the shallow and deep AD baselines that have been presented in Section 2.2.2 as well as relevant self-supervised AD baselines introduced in Section

3.1. In this context, the scalogram representation of the process data $\tilde{\mathbb{D}}_{\text{Bosch}}$, and a representation based on statistical features are used for evaluation purposes to ensure a fair and comprehensive comparison with approaches that were originally designed for tabular data such as DAGMM. Regarding the comparison with statistical features, the following eight statistical measures are computed for each individual sample X and each of its m features based on the individual splits of $\mathbb{D}_{\text{Bosch}}$: root-mean-square, peak-to-peak, interquartile range, mean, standard deviation, kurtosis, skewness, and median absolute deviation. The resulting dataset will be referred to as $\mathbb{D}_{S,\text{Bosch}} \subset \mathbb{R}^{24}$, with $|\mathbb{D}_{S,\text{Bosch}}| = 143$. Note that the employed statistical features represent a common choice in practical applications, see, e.g., [Bieg22b]. Next, the data are z-score scaled based on the corresponding train split of $\mathbb{D}_{S,\text{Bosch}}$, i.e., $\mathbb{X}_{\text{train}}^{S,\text{Bosch}}$.
The shallow AD baselines OC-SVM, IF, PCA, and KPCA are evaluated on both datasets. In terms of $\mathbb{D}_{S,\text{Bosch}}$, each baseline is trained directly on $\mathbb{X}_{\text{train}}^{S,\text{Bosch}}$. With respect to $\tilde{\mathbb{D}}_{\text{Bosch}}$, a LeNet-type AE is trained on $\tilde{\mathbb{X}}_{\text{train}}^{\text{Bosch}}$ prior to applying the individual baselines. The encoder of the LeNet-type AE uses the same structure as the LeNet-type encoder network f in SSMSPC. The corresponding decoder is constructed symmetrically, replacing convolutions with deconvolutions and max-pooling with upsampling. After training the LeNet-type AE, the baselines are trained on the respective encoded features. Note that the procedure described above is common practice for applying shallow baselines to image data, see, e.g., [Gola18]. Table C.1 in Appendix C.1.1 displays the selected hyperparameters for the shallow baselines that are obtained via a grid search based on the performance of the respective validation sets.
In terms of deep AD baselines, Deep-SVDD, LeNet-type AE, and PatchCore are evaluated on $\tilde{\mathbb{D}}_{\text{Bosch}}$, while DAGMM is evaluated on $\mathbb{D}_{S,\text{Bosch}}$. The LeNet-type AE is the same that has been used to encode the features for the shallow AD baselines. Table C.2 in Appendix C.1.1 visualizes the hyperparameters for the deep baselines. As before, the hyperparameters are obtained via grid search based on the performance on the respective validation sets. For Deep-SVDD, DAGMM, and PatchCore, the official implementation provided with the respective publication is used.
Regarding the self-supervised AD baselines, four approaches are considered and evaluated on $\tilde{\mathbb{D}}_{\text{Bosch}}$. The first two approaches represent NeuTraL AD and GeoTrans. The remaining two approaches rely on the RotNet presented by [Gida18]. Concretely, in the third approach, the learned representations of RotNet are used to fit the shallow AD baselines OC-SVM, IF, PCA, and KPCA. This corresponds to an application of the two-stage framework by [Sohn21]. The fourth approach follows a scheme described by [Hend19] and relies on the softmax output to determine if a sample is normal or anomalous. Table C.3 in Appendix C.1.1 illustrates the corresponding hyperparameter settings for the self-supervised AD baselines. Again, all hyperparameters are selected via grid search based on their performance on the validation set. For all self-supervised AD baselines, the official implementations as published by the respective authors are used.

Results

The results of SSMSPC and the respective baselines on the Bosch CNC milling dataset are displayed in Table 6.5. The mean and standard deviation of the AUROC computed over ten different random seeds are used for evaluation purposes to obtain a holistic picture of the individual model performances across all possible trade-off decisions of the corresponding test sets.
Observing Table 6.5, it stands out that SSMSPC, the LeNet-type AE, and PatchCore achieve a perfect overall score, closely followed by Deep-SVDD with an AUROC of 99.9%. Focusing on the shallow AD baselines, it can be noticed that the models based on statistical features outperform the models that rely on the scalogram representation. Especially, the OC-SVM and PCA achieve a high AUROC score of

Table 6.5.: Results on the Bosch CNC milling dataset. The mean and standard deviation of the AUROC are reported over 10 different random seeds, adapted from [Bieg23].

Category	Datasets	Method	AUROC
Shallow	$\mathbb{D}_{S,\text{Bosch}}$	OC-SVM	91.5 ± 4.4
		PCA	89.7 ± 4.7
		KPCA	88.0 ± 4.8
		IF	79.1 ± 8.7
	$\tilde{\mathbb{D}}_{\text{Bosch}}$	LeNet + PCA	75.9 ± 10.3
		LeNet + KPCA	78.3 ± 8.7
		LeNet + OC-SVM	81.3 ± 9.8
		LeNet + IF	54.3 ± 10.8
Deep	$\mathbb{D}_{S,\text{Bosch}}$	DAGMM	92.8 ± 5.5
	$\tilde{\mathbb{D}}_{\text{Bosch}}$	Deep-SVDD	99.9 ± 0.2
		LeNet-AE	$\mathbf{100.0} \pm \mathbf{0.0}$
		PatchCore	$\mathbf{100.0} \pm \mathbf{0.0}$
Self-supervised	$\tilde{\mathbb{D}}_{\text{Bosch}}$	NeuTraL AD	97.9 ± 3.6
		GeoTrans	94.5 ± 6.7
		RotNet	56.0 ± 9.9
		RotNet + PCA	94.6 ± 5.6
		RotNet + KPCA	84.7 ± 12.0
		RotNet + OC-SVM	98.4 ± 1.6
		RotNet + IF	97.6 ± 3.1
		SSMSPC	$\mathbf{100.0} \pm \mathbf{0.0}$

91.5% and 89.7%, respectively. As stated above, most of the deep AD baselines achieve a performance that is on par with SSMSPC. DAGMM represents the best-performing model that is being evaluated based on statistical features. However, with an AUROC of 92.8%, its performance lags behind the other deep AD models. The self-supervised AD baselines show an overall strong performance with RotNet + IF, NeuTraL AD and RotNet + OC-SVM achieving an AUROC of 97.6%, 97.9% and 98.4%, respectively. Only the RotNet approach, relying on the softmax output lags behind, being barely able to exceed the random guessing baseline with an AUROC of 56%.
Thus, from a threshold-independent point of view, it can be stated that SSMSPC is capable of outperforming leading AD baselines on this dataset. In order to validate that SSMSPC achieves a high recall while maintaining a low false alarm rate, which represent two of the four external requirements, a threshold-dependent evaluation is conducted. In this evaluation, the threshold mapping b_{UCL} that has been fitted in the AD downstream task is employed to classify the individual samples. The performance is measured in terms of precision, recall, and f1-score computed over ten different random seeds and for varying values of α. Table 6.6 visualizes the result. It can be seen that with the default significance value $\alpha = 0.01$, SSMSPC achieves perfect recall while simultaneously maintaining high precision. Thus, the external requirements of high recall and a low false alarm rate are satisfied.

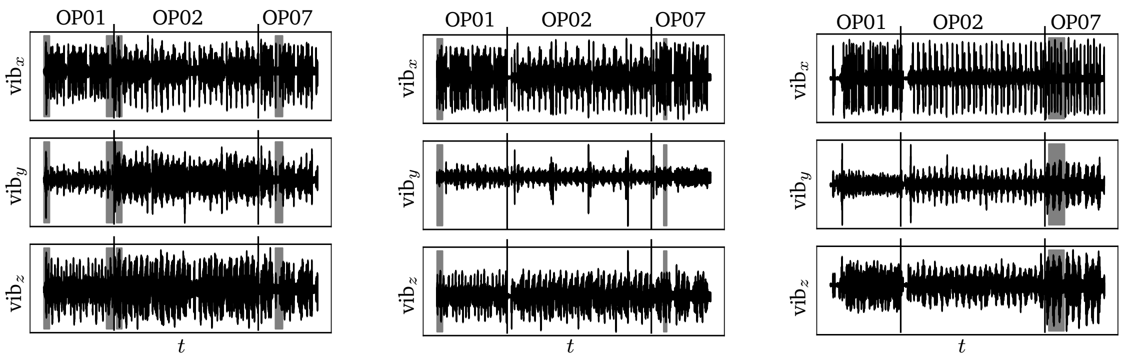
SSMSPC control chart extension

Having demonstrated that SSMSPC reliably detects anomalies, it will now be investigated if the proposed control chart extension scheme correctly locates anomalies by highlighting anomalous regions in the process data. Since the Bosch CNC milling dataset does not provide details regarding the exact location of the anomalies that occur within an OP, the subsequent investigation is of qualitative nature. Specifically, for $X \in \mathbb{D}_{\text{Bosch}}$, the original labels of OP01, OP02, and OP07 are used

Table 6.6.: Precision, recall, and f1-score of SSMSPC on the Bosch CNC milling dataset for varying levels of α. The mean and standard deviation are reported over 10 different random seeds, adapted from [Bieg23].

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

to mark the anomalous process modes. For a detected anomalous sample, it is then checked, whether SSMSPC highlights the anomalous regions only within the respective anomalous modes. Figure 6.1 visualizes the qualitative results of the proposed control chart extension using process samples from the validation set. Following the procedure described above, it can be shown that SSMSPC correctly locates the anomalous process modes.



OP01	OP02	OP07
1	1	1

OP01	OP02	OP07
1	0	1

OP01	OP02	OP07
0	0	1

Figure 6.1.: Qualitative results of the proposed control chart extension on the Bosch CNC milling dataset. Three random samples from the validation set are displayed. The tables below indicate the true label of the respective mode. SSMSPC correctly highlights the anomalous regions in the process data, own illustration adapted from [Bieg23].

Ablation study

After assessing the detection and localization capabilities of SSMSPC, the meaningfulness of individual design choices is investigated in the following. Concretely, based on the Bosch CNC milling dataset, an ablation study will be conducted that focuses on three aspects. First, the relevance of the proposed default augmentation functions will be evaluated by successively adding and removing augmentation functions. Second, the effects of reducing or increasing the number of windows p will be taken into account. Third, the impact of the Location + Transformation prediction pretext task will be analyzed by training SSMSPC separately on either the location prediction objective or the transformation prediction objective.

Regarding the first part of the ablation study, three additional augmentation functions are defined: (1) *PointOutlier*, (2) *Trend*, and (3) *Gaussian*. As the name suggests, the *PointOutlier* augmentation emulates a contextual point anomaly. For this purpose, a random window is chosen and two random points are selected to which the maximum value of the entire time series is added. The *Trend* augmentation corresponds to a gradually increasing shift in the process data. More precisely, a disturbance factor in the form of a linearly growing multiple of the standard deviation of the respective time series is added to every sample in a randomly selected window segment. With respect to the *Gaussian* augmentation, random numbers sampled from a normal distribution with a fixed mean and standard deviation are added to the values in a randomly selected window segment. The three additional augmentation functions are successively included in the data augmentation process of SSMSPC. The results are displayed in Table 6.7. It can be observed that adding the individual augmentation functions results in a deterioration of performance. Especially the Gaussian augmentation seems to decrease the effectiveness of the learned representations, since its inclusion results in a performance drop of 4%. A possible reason for the deterioration of SSMSPC in this experiment could be that the additional augmentation functions are not representative of the anomalies encountered in the underlying dataset.

Table 6.7.: Results of adding augmentations in the pretext task of SSMSPC. The mean and standard deviation of the AUROC are reported over 10 different random seeds, adapted from [Bieg23].

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

Aside from adding novel augmentation functions, Table 6.8 displays the performance of SSMSPC when the default augmentation functions are removed one after another. Interestingly, it can be observed that removing any of the proposed default augmentation functions results in declining performance. The *MissingSignal* augmentation seems to be the most important augmentation, as removing it reduces the performance to an AUROC of 98.7%. The obtained results indicate that the presented set of default augmentations for SSMSPC represents a reasonable choice for its practical application on the given dataset.

Table 6.8.: Results of removing default augmentations in the pretext task of SSMSPC. The mean and standard deviation of the AUROC are reported over 10 different random seeds, adapted from [Bieg23].

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

Next, the effects of varying the number of windows will be investigated. For this purpose, SSMSPC is trained with the default augmentation functions using different values of p. Table 6.9 displays the results. It can be seen that SSMSPC is fairly robust regarding changes in p, since both reducing and increasing the number of windows results in only a slight performance drop. This is an important feature for the hyperparameter search in practical applications. While it is difficult to determine the exact reason for the observed performance drop in the underlying example, it is likely a combination of random effects and the difficulty level of the respective pretext task. When the number of windows is reduced, the pretext task decreases in difficulty since there are fewer possible combinations to augment a time series. Specifically, for a fixed number of augmentations k, decreasing the number of windows from p to $p - 1$, reduces the number of possible combinations by k. Conversely, increasing the number of windows from p to $p + 1$, raises the number of possible combinations by k, which translates into a more difficult pretext task. Thus, it is surmised that if the difficulty level does not fit to the capabilities of the employed model, the resulting learned representations are less effective.

Table 6.9.: Results of varying the number of windows in the data augmentation process for the pretext task. The mean and standard deviation of the AUROC are reported over 10 different random seeds, adapted from [Bieg23].

Method	#Windows	AUROC
SSMSPC	3	99.5 ± 0.7
	5	$\mathbf{100.0 \pm 0.0}$
	7	98.9 ± 1.5

Lastly, the impact of the Location + Transformation prediction pretext task will be evaluated. For this purpose, SSMSPC is trained individually on the location prediction and the transformation prediction objective, by removing either g_{trans} or g_{loc} from the model architecture. The remaining settings are left unchanged. Looking at Table 6.10, it can be observed that training SSMSPC on either the location prediction or the transformation prediction objective decreases the overall performance. Thus, it can be confirmed that combining both objectives in the Location + Transformation prediction pretext task results in more effective representations for the AD downstream task.

Table 6.10.: Results of varying the pretext task objective. The mean and standard deviation of the AUROC are reported over 10 different random seeds, adapted from [Bieg23].

Method	Pretext task	AUROC
SSMSPC	Location + Transformation prediction	$\mathbf{100.0 \pm 0.0}$
	Transformation prediction	99.2 ± 1.6
	Location prediction	97.9 ± 1.7

6.1.2. CiP-DMD

The second evaluation in the performance comparison between SSMSPC and state-of-the-art AD baselines is based on the CiP-DMD [Jour23], a novel benchmark dataset featuring a multi step discrete manufacturing process in which pneumatic cylinders are produced. More precisely, the evaluation is conducted on a subset of the CiP-DMD that has been recorded by the author of this thesis in the process learning factory CiP at the Technical University of Darmstadt between November 2022 and

March 2023. This dataset corresponds to a CNC milling process that produces the steel cylinder bottoms of the pneumatic cylinders. It has explicitly been designed to be applicable to the problem statement described in Section 4.3. Note that the underlying CNC milling process serves as the basis for conducting the upcoming usability study in Section 6.2, in which SSMSPC is compared to univariate post-process SPC. In the following paragraphs, the process and the dataset are described in more detail.

Process description

The dataset under consideration has been recorded on a Deckel Maho DMC 50H horizontal CNC milling machine which is depicted in Figure 6.2.

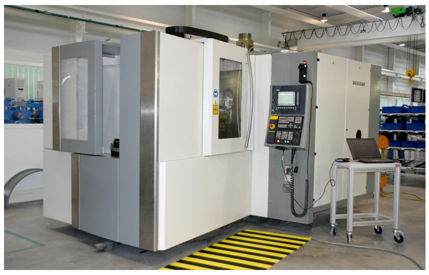


Figure 6.2.: DMC 50H machine used for dataset recording, adapted from [Bieg23].

The DMC 50H is equipped with a rotating tower that is located within the machining space. In order to produce the cylinder bottoms, the front and back sides of the tower are used. On both sides of the tower, special fixtures are mounted in which up to four different parts can be clamped. During a machining cycle, only one clamping position is occupied on either side. Figure 6.3 depicts both sides of the rotating tower and highlights the respective clamping positions.

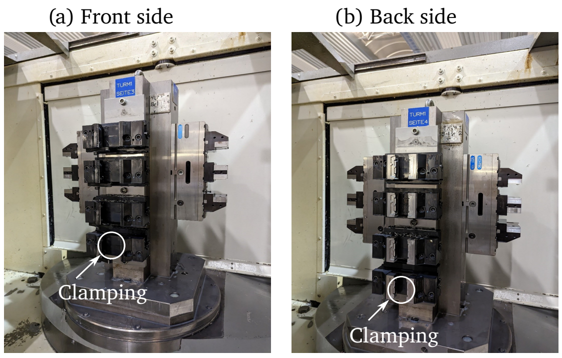


Figure 6.3.: Rotating tower of the DMC 50H CNC milling machine. (a) Front side: Used to clamp a piece of raw material. (b) Back side: Used to clamp a semi-finished part, adapted from [Bieg23].

Looking at the illustration, it can be observed that the fixtures mounted on each tower side differ in their shape. This is due to the fact that the fixtures on the front side are designed to hold pieces of raw material, while the fixtures on the back side are designed to hold semi-finished parts. Prior to starting a machining cycle, the rotating tower is loaded with two distinct parts, i.e., a piece of raw material on the front side and a semi-finished part on the back side. Both parts are then processed in sequence. In the beginning, the raw material is converted into a semi-finished part. Following this,

the tower rotates by 180 degrees, and the semi-finished part on the back side is transformed into a finished part. Thus, the output of a machining cycle is a semi-finished part and a finished part.
The processing time to convert a piece of raw material to a semi-finished part amounts to 199 seconds and encompasses nine different process modes, including face milling, drilling, and threading. Conversely, the processing time to convert a semi-finished part into a finished part requires 113 seconds and consists of four different process modes such as face milling, and deburring. Consequently, the total duration of a machining cycle amounts to approximately five minutes. For notational convenience, the raw material will henceforth be referred to as part one (P1), while the semi-finished part will be denoted as part two (P2). Figure 6.4 illustrates both P1 and P2 before and after the machining cycle. Table 6.11 displays the individual process modes.

(a) Cylinder bottom - P1 (b) Cylinder bottom - P2

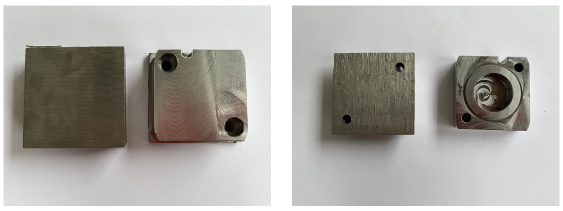


Figure 6.4.: P1 and P2 before and after processing. (a) P1: Conversion from raw material (left) to semi-finished part (right). (b) P2: Conversion from semi-finished part (left) to finished part (right), adapted from [Bieg23].

Table 6.11.: Process modes of P1 and P2, own illustration.

P1				P2			
No.	Description	Tool	∅ [mm]	No.	Description	Tool	∅ [mm]
1	Face milling	Face milling cutter	50.00	1	Face milling	Face milling cutter	50.00
2	Outer contour	End mill	10.00	2	Circular pocket	End mill	10.00
3	Lateral groove	End mill	6.00	3	Deburring	Deburrer	10.00
4	Step drilling	Step drill	6.60	4	Ring groove	Spindle tool	40.00
5	Deburring	Deburrer	10.00				
6	Lateral drilling	Drill	8.60				
7	Countersinking	Countersink	10.00				
8	Drilling	Drill	5.00				
9	Thread milling	Thread cutter	9.73				

Upon completion of the machining cycle, the finished cylinder bottom undergoes a quality control in which four quality characteristics are measured: (1) surface roughness, (2) parallelism, (3) groove depth, and (4) groove diameter. For this purpose, Mahr measurement devices are utilized that are connected to and operate in conjunction with the Hexagon/QDAS software. Figure 6.5 provides an illustration of the measurement equipment and the software.

Dataset description

Having provided a general description of the process, emphasis will now be placed on the actual dataset that is used for the evaluation. The dataset features two relevant types of data: (1) labelled process data and (2) metadata.

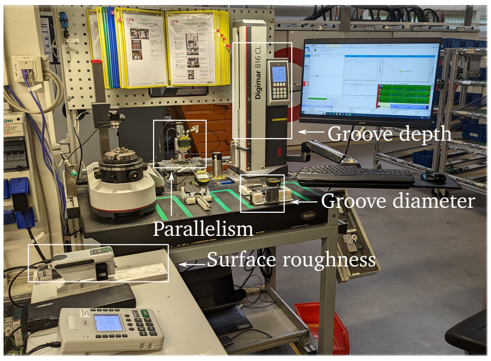


Figure 6.5.: Measuring station used for quality measurements, own illustration adapted from [Jour23].

The process data correspond to vibration data collected from a triaxial accelerometer at a sampling rate of 2.5kHz. The accelerometer is mounted on the main spindle of the machine and is connected to a data acquisition card from National Instruments. The data acquisition card is plugged into an industrial computer that is physically placed on top of the machine. Figure 6.6 provides a visualization of the sensor within the machining space of the DMC 50H. Regarding the metadata, the machine has been retrofitted with the Siemens Brownfield Connectivity Gateway to provide access to internal machine control data, e.g., the current NC line at a sampling rate of 5Hz. These metadata are utilized to segment the time series for the proposed control chart extension, see Section 6.2.

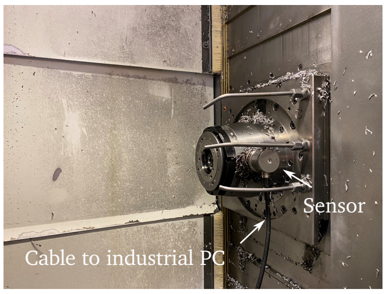


Figure 6.6.: Triaxial accelerometer mounted on the main spindle of the DMC 50H, adapted from [Bieg23].

The dataset encompasses a total of 776 completed machining cycles. In the course of these cycles, 776 samples of P1 and 775 samples of P2 are collected. The discrepancy in the number of samples between P1 and P2 results from a so-called "air cut" in the first machine cycle with respect to P2. Specifically, during this cycle, the back side of the rotating tower is empty since there is no semi-finished part yet. Thus, the corresponding sensor recordings for P2 in this cycle are excluded from the dataset.
Regarding P1, 736 samples are recorded that represent an in-control process, whereas for P2, this number amounts to 737 samples. For all remaining samples, the process is affected by one of two predefined but realistic anomalies. Realistic in this context means that these anomalies are not fictitious, but can actually occur during the production of a cylinder bottom. The first anomaly affects P1 and results from a piece of raw material that is cut off too short in the preceding sawing process. Cutting off a part too short translates into no chip removal during the face milling step of P1, which results in a semi-finished part with an unprocessed surface. This anomaly thus affects the surface roughness measurements in the ensuing quality control. The second anomaly affects P2 and corresponds to a

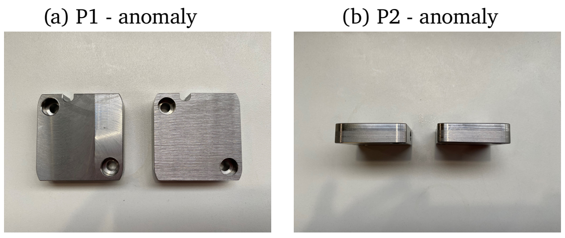


Figure 6.7.: Extent of anomalies for P1 and P2 on a completed cylinder bottom. (a) Unprocessed surface: Comparison between normal (left) and anomalous (right) cylinder bottoms. (b) Crooked part: Comparison between normal (left) and anomalous (right) cylinder bottoms, adapted from [Bieg23].

crooked part resulting from uneven clamping by the machine operator. When this clamping error occurs, the face milling step of P2 removes more chips from one side than from another. This affects the measurements of parallelism and groove depth quality characteristics. Figure 6.7 depicts the extent of the anomalies for P1 and P2 on a completed cylinder bottom.
The respective datasets for P1 and P2 are referred to as $\mathbb{D}_{\text{P1}}$ and $\mathbb{D}_{\text{P2}}$. Here, $\mathbb{D}_{\text{P1}} \subset \mathbb{R}^{497500\times3}$ with $|\mathbb{D}_{\text{P1}}| = 776$, whereas $\mathbb{D}_{\text{P2}} \subset \mathbb{R}^{282500\times3}$ with $|\mathbb{D}_{\text{P2}}| = 775$. Both datasets are split into a train, validation, and test dataset, using the split ratio from the first evaluation, i.e., 60%, 10%, and 30%, respectively. Recall that the respective train datasets encompass only in-control data. Table 6.12 displays the number of samples for each split of $\mathbb{D}_{\text{P1}}$ and $\mathbb{D}_{\text{P2}}$. Based on these datasets, the performance comparison will be conducted in the following.

Table 6.12.: Train validation and test splits of $\mathbb{D}_{\text{P1}}$, $\mathbb{D}_{\text{P2}}$ including the number of normal and anomalous samples, adapted from [Bieg23].

	Split	Normal	Anomalous	Total
$\mathbb{D}_{\text{P1}}$	$\mathbb{X}^{\text{P1}}_{\text{train}}$	465	–	465
	$\mathbb{X}^{\text{P1}}_{\text{val}}$	67	10	77
	$\mathbb{X}^{\text{P1}}_{\text{test}}$	204	30	234
$\mathbb{D}_{\text{P2}}$	$\mathbb{X}^{\text{P2}}_{\text{train}}$	465	–	465
	$\mathbb{X}^{\text{P2}}_{\text{val}}$	68	9	77
	$\mathbb{X}^{\text{P2}}_{\text{test}}$	204	29	233

SSMSPC - pretext task

Regarding the Location + Transformation prediction pretext task for SSMSPC, the same procedure is followed as that presented in the first evaluation. In the beginning, the respective train splits $\mathbb{X}^{\text{P1}}_{\text{train}}$, $\mathbb{X}^{\text{P2}}_{\text{train}}$ are augmented with the default data augmentation functions *Identity*, *CutPaste*, *MeanShift*, and *MissingSignal* using $p = 5$ windows. Following this, the scalogram representation of the augmented time series data is retrieved by applying the CWT with $v = 128$ scales using the Morlet wavelet. Next, the scalograms are resized down to 128×512 and min-max scaling is applied. Completing this procedure results in the respective pretext datasets $\mathbb{X}^{\text{P1}}_{\text{pretext}} \subset \mathbb{R}^{128\times512\times3}$, $\mathbb{X}^{\text{P2}}_{\text{pretext}} \subset \mathbb{R}^{128\times512\times3}$, where $|\mathbb{X}^{\text{P1}}_{\text{pretext}}| = |\mathbb{X}^{\text{P2}}_{\text{pretext}}| = 1860$.

Once the pretext datasets are established, the model architecture, consisting of the LeNet-type encoder network f and the projection heads g_{trans} and g_{loc}, is trained on the Location + Transformation prediction pretext task. The individual layers and number of parameters of both the encoder and the projection heads are identical to the values displayed in Table 6.3. However, as opposed to the first evaluation, the hyperparameters for the training process are slightly changed. Concretely, for both $\mathbb{X}^{\text{P1}}_{\text{pretext}}$ and $\mathbb{X}^{\text{P2}}_{\text{pretext}}$, the batch size is increased from 8 to 32, while the number of epochs is reduced from 20 to 10. The remaining hyperparameters are left unchanged.

SSMSPC - AD downstream task

In terms of the AD downstream task, there are also no deviations from the procedure followed in the first evaluation. Consequently, in the beginning, the scalogram representations $\tilde{\mathbb{D}}_{\text{P1}}$, $\tilde{\mathbb{D}}_{\text{P2}}$ of the individual splits of $\mathbb{D}_{\text{P1}}$, $\mathbb{D}_{\text{P2}}$ are obtained using $v = 128$ scales and the Morlet wavelet. After this, the scalograms are resized to 128×512 and min-max scaling is applied based on the respective train splits $\tilde{\mathbb{X}}^{\text{P1}}_{\text{train}}, \tilde{\mathbb{X}}^{\text{P1}}_{\text{train}}$.
Following the scalogram computation, the learned representations $f(\tilde{\mathbb{X}}^{\text{P1}}_{\text{train}})$ and $f(\tilde{\mathbb{X}}^{\text{P2}}_{\text{train}})$ are extracted to compute the respective anomaly scores using the Hotelling's T^2 statistic. The individual model architectures for the AD downstream task are identical to the architecture presented in Table 6.4. The threshold mapping b_{UCL} is again obtained via KDE using the default significance value of $\alpha = 0.01$.

State-of-the-art baselines

In alignment with the first evaluation, the performance of SSMSPC is compared to the same shallow, deep, and self-supervised AD baselines using both the respective scalogram representations of the process data $\tilde{\mathbb{D}}_{\text{P1}}$, $\tilde{\mathbb{D}}_{\text{P2}}$ and representations based on statistical features. The corresponding datasets $\mathbb{D}_{S,\text{P1}}, \mathbb{D}_{S,\text{P2}} \subset \mathbb{R}^{24}$ that are based on statistical features are obtained by following the description of the first evaluation. Accordingly, for the dataset splits of $\mathbb{D}_{\text{P1}}$, $\mathbb{D}_{\text{P2}}$, the statistical measures: root-mean-square, peak-to-peak, interquartile range, mean, standard deviation, kurtosis, skewness, and median absolute deviation are computed with respect to each sample and each of the m sensors. Following this, the individual splits of $\mathbb{D}_{S,\text{P1}}, \mathbb{D}_{S,\text{P2}}$ are z-score scaled based on $\mathbb{X}^{\text{P1}}_{S,\text{train}}$ and $\mathbb{X}^{\text{P2}}_{S,\text{train}}$, respectively. The hyperparameters of the shallow, deep, and self-supervised AD baselines are selected based on the performance on the validation set using grid search and are displayed in Appendix C.1.2 in Tables C.4, C.5, and C.6. Note that the hyperparameters of the deep and some of the self-supervised AD baselines are identical to the first evaluation, since they perform best on the validation set.

Results

The results of the second evaluation are visualized in Table 6.13. For SSMSPC and all state-of-the-art AD baselines, the mean and standard deviation of the AUROC are reported over five different random seeds for P1 and P2. Additionally, a "Total" column is included that summarizes the performance of the individual models on P1 and P2 in terms of a simple mean. The resulting score represents the overall performance and is used as the final evaluation measure.
Looking at Table 6.13, it can be observed that SSMSPC achieves the highest overall performance (99.6%) and surpasses all state-of-the-art baselines on P1. Surprisingly, the shallow AD baselines that are evaluated on statistical features achieve remarkable results compared to the first evaluation. Especially the overall performance of PCA stands out as it dominates leading deep and shallow state-of-the-art AD baselines such as Deep-SVDD ($+18.6\%$), PatchCore ($+0.1\%$), and NeuTraL AD ($+0.4\%$).

Table 6.13.: Results on the CiP-DMD. The mean and standard deviation of the AUROC are reported over 5 different random seeds for P1 and P2, respectively, adapted from [Bieg23].

Category	Datasets	Method	AUROC P1	AUROC P2	Total
Shallow	$\mathbb{D}_{S,\mathrm{P1}}, \mathbb{D}_{S,\mathrm{P2}}$	OC-SVM	98.3 ± 1.2	98.8 ± 0.7	98.5
		PCA	99.3 ± 0.3	$\mathbf{99.5 \pm 0.4}$	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
	$\tilde{\mathbb{D}}_{\mathrm{P1}}, \tilde{\mathbb{D}}_{\mathrm{P2}}$	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	$\mathbb{D}_{S,\mathrm{P1}}, \mathbb{D}_{S,\mathrm{P2}}$	DAGMM	99.4 ± 0.3	95.3 ± 3.0	97.4
	$\tilde{\mathbb{D}}_{\mathrm{P1}}, \tilde{\mathbb{D}}_{\mathrm{P2}}$	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	$\tilde{\mathbb{D}}_{\mathrm{P1}}, \tilde{\mathbb{D}}_{\mathrm{P2}}$	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	$\mathbf{99.8 \pm 0.25}$	99.4 ± 0.3	$\mathbf{99.6}$

Focusing only on P2, PCA even surpasses SSMSPC ($+0.1\%$). However, when the shallow AD baselines are evaluated based on the encoded scalogram representation of the LeNet-type encoder, the overall performance drops by at least 14.8% relative to SSMSPC.
Compared to the results from the first evaluation, the performance of some of the deep AD baselines such as Deep-SVDD and the LeNet-type AE strongly deteriorate. Interestingly, while achieving on-par performance with SSMSPC in the preceding evaluation, the LeNet-type AE experiences a performance drop of 32.3%. On the other hand, PatchCore and DAGMM achieve very competitive results. In fact, PatchCore has the third-strongest overall performance.
In terms of the self-supervised AD baselines, NeuTraL AD represents the strongest approach, with an overall performance of 99%. Similar to the first evaluation, the overall performance of RotNet when used as a feature extractor for OC-SVM, IF, PCA and KPCA is strong, achieving a mean score of $\geq 95\%$.
Following the threshold-independent evaluation, a threshold-dependent evaluation is conducted on P1 and P2 to investigate whether SSMSPC is capable of achieving high recall while maintaining a low false alarm rate. Aside from the default significance value of $\alpha = 0.01$, several values for α are investigated to understand how varying the UCL affects the performance. The results are displayed in Table 6.14. For both P1 and P2, the best performance in terms of recall is obtained with a significance value of $\alpha = 0.05$. Here, SSMSPC achieves a recall of 98.3% and 100.0%, respectively. However, looking at the f1-score, this value for α results in the worst overall performance. Regarding P1, the best-performing UCL corresponds to $\alpha = 0.025$, while for P2, the default significance value of $\alpha = 0.01$ is superior. For these values, the corresponding recall still amounts to 94.8% for P1 and 93.1% for P2. This again highlights that the default setting for the UCL represents a reasonable choice for practical applications.

Summarizing, the preceding performance comparison provided four major insights: First, it can be concluded that the proposed SSMSPC approach is indeed capable of learning effective representations for the detection and localization of anomalies in practical applications that are consistent with the problem statement formulated in Section 4.3. Second, it can be confirmed that SSL is a powerful and seminal concept that surpasses state-of-the-art AD baselines. Third, in certain situations and especially with respect to applications in discrete manufacturing, the performance of some deep AD baselines can fluctuate strongly and often lags behind basic shallow approaches. Incidentally, this observation was also made in other works, e.g., [Bieg22a]. Fourth, approaches like PatchCore that leverage the learned representations from a pretraining phase on large amounts of data from another domain show intriguing results and bear a non-negligible potential for future applications in discrete manufacturing.

Table 6.14.: Precision, recall, and f1-score of SSMSPC on the CiP-DMD for varying levels of α. The mean and standard deviation are reported over 5 different random seeds, adapted from [Bieg23].

	α	Precision	Recall	F1-score
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
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

6.2. Validation II - usability study

In this section, the usability of SSMSPC as a deployed monitoring system will be evaluated and compared to conventional univariate post-process SPC. For this purpose, a usability study with human machine operators is conducted that centers around the CNC milling process described in Section 6.1.2. In Section 6.2.1, the developed Graphical User Interface (GUI) of the deployed SSMSPC monitoring system is introduced. Following this, Section 6.2.2 elaborates the study design. Finally, Section 6.2.3 presents the results.

6.2.1. Graphical user interface - SSMSPC

In order to evaluate the usability of SSMSPC on the considered CNC milling process, a web-based GUI is developed using the Python library Dash. The GUI provides two visualizations: (1) the control chart view to detect anomalies and (2) the control chart extension to highlight the anomalous regions in the process data.

The first view displays the control charts for both P1 and P2, which are updated automatically during the process. The underlying SSMSPC models are trained with the same settings that were used in Section 6.1.2. However, as opposed to relying only on the in-control data from the respective train splits, the models are trained on all available in-control data, i.e., based on the in-control data contained in the corresponding train, validation, and test splits. Regarding the required UCLs, the respective control charts use the default significance value of $\alpha = 0.01$.

Once an anomaly is detected by one of the control charts, the operator can access the second view, i.e., the respective control chart extension, by clicking on the corresponding sample. In this view, the segmented raw process data are displayed and the anomalous regions are highlighted. The segmentation is conducted based on the time steps of the NC lines that contain the required information about the starting point of the individual process modes. Figure 6.8 provides an illustration of the developed GUI.

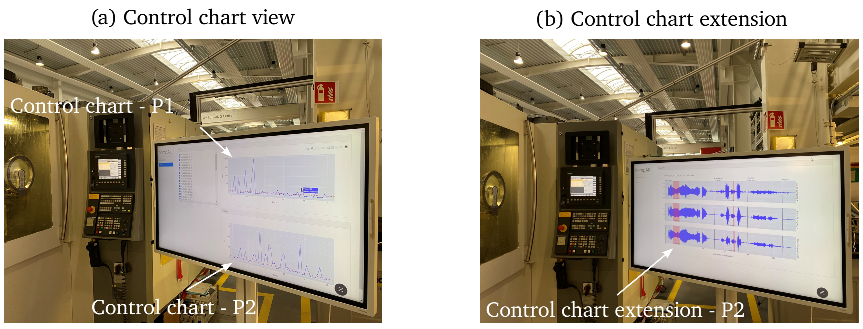


Figure 6.8.: Web-based GUI of the deployed SSMSPC system. (a) Control chart views for P1 and P2: This visualization supports the machine operator in detecting anomalies. (b) Control chart extension exemplary for P2: This visualization highlights the anomalous regions in the process data to support the root-cause analysis, own illustration adapted from [Bieg23].

6.2.2. Study design

The usability study is conducted with human machine operators that are given the assignment to monitor the CNC milling process described in Section 6.1.2 using both SSMSPC and conventional univariate post-process SPC.
Each machine operator undergoes two successive test runs. Within each test run, the task is to monitor the production of three cylinder bottoms, i.e., three machining cycles, using either SSMSPC or univariate post-process SPC. Of the three machining cycles that are conducted during a test run, one corresponds to an in-control process, while the remaining two cycles are affected by the two anomalies, i.e., an unprocessed surface and a crooked part, see Figure 6.7. The order in which the monitoring approaches are tested, as well as the respective sequence of normal and anomalous cycles within a test run, is random and varies from one operator to another.
The usability of both monitoring approaches is assessed with the help of objective and subjective evaluation measures. The objective evaluation measure corresponds to a monitoring protocol that represents an assessment sheet for the process and is filled out by the machine operator for every machining cycle. The subjective evaluation measure corresponds to the System Usability Scale (SUS) [Broo96], a standardized ten-item questionnaire. This questionnaire is filled out once for each monitoring approach at the end of a test run.
After the completion of both test runs, an unstructured interview is conducted in which the machine operator is questioned with respect to the practical applicability of SSMSPC. Figure 6.9 summarizes the preceding statements in an overview of the study design.
In the following, additional details will be provided regarding: (1) the machine operators, (2) the experimental setup, (3) the usability evaluation, (4) the interview, and (5) the experimental procedure.

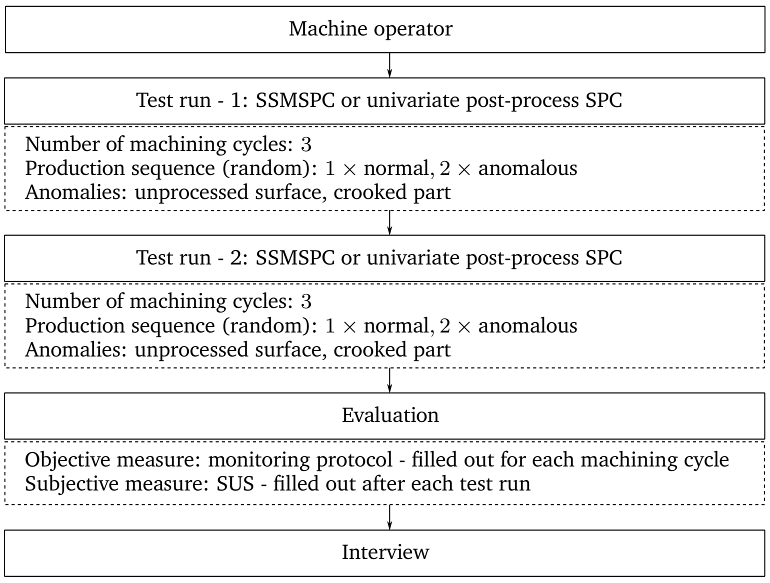

Machine operator
Test run - 1: SSMSPC or univariate post-process SPC
Number of machining cycles: 3
Production sequence (random): 1 × normal, 2 × anomalous
Anomalies: unprocessed surface, crooked part
Test run - 2: SSMSPC or univariate post-process SPC
Number of machining cycles: 3
Production sequence (random): 1 × normal, 2 × anomalous
Anomalies: unprocessed surface, crooked part
Evaluation
Objective measure: monitoring protocol - filled out for each machining cycle
Subjective measure: SUS - filled out after each test run
Interview

Figure 6.9.: Design of the usability study, own illustration.

Machine operators

A total of eight machine operators participate in the study. These machine operators represent student workers that are employed in the process learning factory CiP. Each of the participants has been employed for at least six months and has received professional training by a technician to operate the DMC 50H machine as well as the measuring station depicted in Figure 6.5. None of the participants possesses prior knowledge about the specific CNC milling process under consideration, nor are they familiar with the SSMSPC approach or the GUI. However, since all the participants are employed in the process learning factory CiP, they are familiar with the production and quality control of similar cylinder bottoms made of aluminum that share some of the process steps and most of the quality characteristics of the considered steel cylinder bottoms.

Experimental setup

The experimental setup for the test runs is depicted in Figure 6.10. As can be seen in the illustration, the space surrounding the DMC 50H machine is partitioned into a setup space and a monitoring space with the help of a privacy screen. The setup space serves the purpose of loading and unloading the machine at the beginning and end of a machining cycle. The machine setup is conducted by an independent machine operator that has prior knowledge about the production sequence but does not participate in the test runs. Next to the setup space is the monitoring space. As the name suggests, the monitoring space is used by the participants of the study to monitor the process through the machine window. Note that once the machine is set up by the independent machine operator, the participants start the machining process themselves. The rationale for physically separating the setup space from the monitoring space is to prevent the participants from inferring the process condition by observing the loading process of the machine.

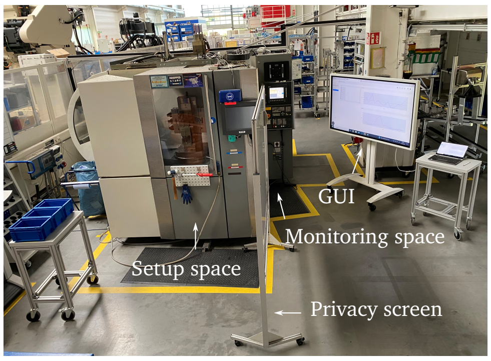


Figure 6.10.: Experimental setup for the test runs in the usability study, own illustration.

Usability evaluation

As stated before, the usability of SSMSPC and univariate post-process SPC is evaluated using the monitoring protocol as an objective measure and the SUS as a subjective measure.
The monitoring protocol is a simple assessment sheet that is filled out by the participants after each machining cycle. Concretely, with the help of the respective monitoring approach, the participants decide if the process condition is normal or anomalous, and in the case of the latter, try to identify the root cause of the anomaly. The results of the process analysis are then written down in the monitoring protocol. Tables C.7 and C.8 in Appendix C.2.1 display the monitoring protocols for SSMSPC and univariate post-process SPC. Note that in the monitoring protocol for SSMSPC, a machining cycle is subdivided into P1 and P2, while the monitoring protocol for univariate post-process SPC considers the machining cycle as a whole. The monitoring protocol corresponds to an objective evaluation measure, since the results of each participant can be compared to the underlying ground-truth production sequence. This allows to assess the performance of both monitoring approaches in terms of the detection of anomalies and the support for correctly identifying the root cause.
The SUS represents a valid and reliable standard tool for the comparison and subjective assessment of the usability of different systems [Broo13]. In fact, it is considered the most widely used tool for measuring the perception of usability [Saur11]. It embodies a questionnaire consisting of ten alternating five-level Likert items ranging from "strongly agree" to "strongly disagree" [Broo96; Broo13]. Here, the odd items encompass positive statements, while the even items are characterized by negative statements. Figure C.1 in Appendix C.2.2 provides an illustration of the SUS questionnaire that is used in the subjective usability assessment. The intriguing aspect of the SUS is that it allows to calculate the so-called SUS score, which is a single number between 0 and 100 that expresses the overall usability of the system under consideration [Broo96]. According to [Saur11], a SUS score of 68 corresponds to a system with average usability. Thus, if a system exceeds this limit, it can be considered to have above-average usability. The computation of the SUS score follows a straightforward two-step routine [Broo96]: First, the score contribution of each Likert item is summed up. For the odd-numbered items, the score contribution is given by the Likert score minus one. Conversely, for all even-numbered items, the score contribution corresponds to the Likert score minus five. Second, the sum of the score contributions is multiplied by the factor 2.5 to obtain the overall score. Equation 6.1 summarizes the preceding statements in a more formal expression:

$$\text{SUS\_score} := 2.5 \left(\sum_{i=1}^{5} (\text{SUS}[2i-1] - 1) + (\text{SUS}[2i] - 5) \right) \in [0; 100]. \tag{6.1}$$

Here, SUS$[i]$ corresponds to the selected Likert score of Likert item i in the SUS questionnaire. Incidentally, according to [Broo13], the computation routine described above is chosen to transform the actual scale of the questionnaire, which ranges from 10 to 50 to a scale that ranges from 0 to 100 for ease of interpretability. In alignment with the recommendations provided in the literature, see, e.g., [Broo96], the SUS is filled out at the end of each test run.

Interview

The unstructured interview is conducted with each participant based on the results of the individual monitoring protocols. In this context, emphasis is placed on the machining cycles that were assessed incorrectly. Given the ground-truth solution of the test runs, the participants are asked to describe the perceived advantages and disadvantages of using the system in a real-world application. If there are any perceived disadvantages, the participants are asked to provide suggestions for further improvements. The main purpose of the interview is to provide a platform for any additional remarks that are not covered in the evaluation measures and gather deeper insights regarding the practical applicability of SSMSPC.

Experimental procedure

After describing the individual components of the study design, the experimental procedure is elaborated in the following. Prior to conducting the main test, each participant undergoes a preliminary test. The preliminary test serves three purposes. First, it is used to provide the participants with the necessary background information regarding the usability study, i.e., general procedure, evaluation measures, and overall objective. Second, once the basic information have been provided and the objective of the study is understood, a single test cylinder bottom is produced that corresponds to an in-control process. During this test machining cycle, the participants have the opportunity to familiarize themselves with both monitoring approaches, i.e., use the GUI, measure the quality characteristics of the produced part, and ask any remaining questions. Third, the independent machine operator, who is responsible for the machine setup, is handed the predefined production plan that contains the machining cycle sequence for the underlying experiment, as shown in Table 6.15. The machining cycle sequence for the individual participants is displayed in Table C.9 in Appendix C.2.3.

Table 6.15.: Exemplary machining cycle sequence for a given participant, own illustration.

Participant	Monitoring approach	Process condition
1	SSMSPC	1. Anomalous - unprocessed surface 2. Anomalous - crooked part 3. Normal
	Univariate post-process SPC	1. Normal 2. Anomalous - crooked part 3. Anomalous - unprocessed surface

Once the preliminary test is terminated, the main test begins. Here, the two test runs described earlier are conducted based on the predefined production sequence. It is important to emphasize that the participants are unaware of the type and number of anomalies that are going to occur during the test runs.

When the process is monitored with SSMSPC, the participants have access to the GUI that provides

live updates about the process condition and is located in the monitoring space, as shown in Figure 6.10. Conversely, when the process is monitored with univariate post-process SPC, the participants have no access to the GUI but have to assess the process condition based on the measured quality characteristics of the completed cylinder bottoms with the help of the measuring station and the Hexagon/QDAS software, as depicted in Figure 6.5. Note that the univariate Shewhart control charts for each quality characteristic are constructed by the software using the recorded in-control quality data, see Section 6.1.2. Recall from Section 2.3.2 that the application of univariate post-process SPC involves sampling subgroups of manufactured products from a process at equidistant time intervals. Thus, the assumption of using univariate post-process SPC to measure every manufactured cylinder bottom is rather unrealistic from a practical point of view and conceals some of its major drawbacks. However, despite this fact, the procedure has been chosen to enable the comparison of both approaches in a reasonable amount of time.
After each machining cycle, the participants are given time to fill out the respective monitoring protocol. At the end of a test run, the participants fill out the SUS questionnaire.
With both test runs completed, the final interview is conducted. The overall duration of the experimental procedure for a single participant amounts to approximately three hours.

6.2.3. Results

In the following, the results of the usability study are discussed. First, the objective evaluation based on the comparison between the monitoring protocols and the ground-truth machining cycle sequence is taken into account. Following this, emphasis is placed on the subjective evaluation using the SUS. Lastly, the main insights from the interviews are presented. Note that due to the small sample size of eight participants, the subsequent analysis is of a descriptive nature.

Objective usability evaluation measure - monitoring protocols

The results of the objective usability evaluation for SSMSPC and univariate post-process SPC are displayed in Table 6.16.

Table 6.16.: Results of the objective usability evaluation (n=8 participants), own illustration adapted from [Niek23].

Monitoring approach	Detection			Localization	Root cause identification	
	Precision	Recall	F1-score		Unprocessed surface	Crooked part
SSMSPC	100%	100%	100%	100%	37.50%	25.00%
Univariate post-process SPC	100%	100%	100%	-	37.50%	87.50%

Looking at the results, it stands out that all participants correctly detected the normal and anomalous machining cycles with either monitoring approach. However, when it comes to determining the root cause of the anomalies, a different picture presents itself.
Focusing on SSMSPC, the inspection of the monitoring protocols reveals that the anomalous process condition is correctly highlighted in all cases. Thus, with the help of the control chart extension, the participants are able to correctly identify which process mode is anomalous. As such, SSMSPC satisfies the interpretability requirement formulated in Section 4.2.2. Perhaps surprisingly, even though the detected anomalies are correctly located in the process data, the participants are unable to infer the corresponding root cause in most instances. Specifically, across both anomaly classes, the root

cause is correctly identified in only about one-third of the cases (31.25%). Based on the statements written in the monitoring protocol, those participants who correctly identified the root cause did so by combining the information from the control chart extension with a visual inspection of the manufactured cylinder bottom. Taking a closer look at the individual anomaly classes, it stands out that the root cause for the unprocessed surface anomaly is more often correctly identified (37.50%) than the root cause of the crooked part anomaly (25.00%). Here, the monitoring protocols indicate that the visual feature of the unprocessed surface is more obvious than that of the crooked part. This fact helps the participants to conclude that the anomaly stems from a cylinder bottom that is cut off too short.
With respect to univariate post-process SPC, the root causes are correctly identified in about two-thirds (62.50%) of the cases. While the root cause of the unprocessed surface anomaly is identified by 37.50% of the participants, which is identical to SSMSPC, the root cause of the crooked part anomaly is correctly determined in 87.50% of the cases. According to the monitoring protocols, this surprising result stems from the additional information obtained by measuring the quality characteristics. Concretely, when the parallelism of the crooked cylinder bottoms is measured, the participants realize that the cylinder bottom is crooked. Using their domain knowledge, it is then comparatively straightforward to infer the root cause of this anomaly.
Thus, in terms of the objective usability evaluation measure, it can be concluded that SSMSPC successfully distinguishes normal from anomalous process behavior and correctly identifies the anomalous process modes. However, regarding root cause identification, the participants are more successful when the quality characteristics of the manufactured parts are measured instead of relying solely on the control chart extension. As such, univariate post-process SPC outperforms SSMSPC in this regard.

Subjective usability evaluation measure - SUS

Having presented the results of the objective evaluation, the focus will now be shifted to the subjective evaluation. Table 6.17 visualizes the SUS scores for both SSMSPC and univariate post-process SPC, based on the completed SUS questionnaires.

Table 6.17.: Results of the subjective usability evaluation (n=8 participants), own illustration adapted from [Niek23].

Participant	SUS score - SSMSPC	SUS score - univariate post-process SPC
1	75.0	60.0
2	97.5	62.5
3	72.5	55.0
4	95.0	67.5
5	62.5	62.5
6	80.0	37.5
7	40.0	85.0
8	87.5	57.5
Total	76.3 ± 17.5	60.9 ± 12.4

With a mean SUS score of 76.3, the perceived usability of SSMSPC surpasses that of univariate post-process SPC (60.9) by more than 25%. It can thus be concluded that, from a subjective usability perspective, SSMSPC fulfills the usability requirement described in Section 4.2.2. In fact, taking into account the average usability baseline of 68 that was mentioned earlier, the perceived usability of

SSMSPC can be considered above average, while the perceived usability of univariate post-process SPC is below average.

Interviews - main insights

So far, the results of the usability study indicate that SSMSPC correctly detects anomalies and identifies the anomalous process modes. In addition, according to the results of the SUS questionnaire, SSMSPC is attributed higher perceived usability than univariate post-process SPC. However, when it comes to the identification of the root cause based on the information provided by the control chart extension of SSMSPC, it is shown that the participants experience difficulties. The interviews conducted provide three major reasons for this phenomenon.
First, a lack of prior experience using SSMSPC. The majority of the participants argue that they would achieve better results in determining the root cause if they had more experience with interpreting the control chart extension. As stated before, unlike univariate post-process SPC, the participants use SSMSPC for the first time during the usability study and therefore have no experience interpreting the control chart extension.
Second, a lack of required domain knowledge. The participants state that they would require additional domain knowledge about the monitored CNC milling process, especially regarding the underlying machining program and the individual processing steps, to draw meaningful conclusions from the information provided by the control chart extension.
Third, missing insights from the quality control step. Almost unanimously, the participants emphasize that the root-cause analysis with SSMSPC could be greatly enhanced when the insights from the control chart extension are combined with the information obtained from measuring the quality characteristics of a manufactured part. All participants who suggest the inclusion of quality measurements into the root-cause analysis confirm that when SSMSPC is used in this form, it represents a suitable replacement for the conventional univariate post-process SPC.

Implications for real-world applications

The findings from the objective and subjective usability evaluation indicate that SSMSPC represents a viable approach that allows to rethink the application of SPC in discrete manufacturing processes. Instead of monitoring a process with univariate quality measurements of manufactured parts sampled in subgroups at equidistant time intervals from the process, SSMSPC essentially provides a 100% inspection by relying entirely on multivariate process data that are collected during the process. It is shown that SSMSPC reliably detects anomalies and provides helpful insights by highlighting the anomalous regions in the process data with the help of the proposed control chart extension. However, the results of the usability study emphasize that monitoring a process with the control chart view and the control chart extension alone is not sufficient to provide effective support for the root-cause analysis. Rather, it is necessary to combine these tools with the insights obtained from a quality control step of the manufactured part to exceed the usability of univariate post-process SPC both from an objective and a subjective perspective. When SSMSPC is used in this way, it has the potential to replace univariate post-process SPC.

Limitations of the usability study

The conducted usability study is subject to three major limitations that need to be taken into account when interpreting the obtained results. First, the number of participants is small. This causes the analysis of the results to be of a descriptive nature. Thus, no statistical significance can be demonstrated. Second, the study is conducted with student workers rather than actual professional machine operators. It is surmised that the results regarding the interpretability of the control chart extension might be different when analyzed with experienced machine operators. Third, for reasons of feasibility, the univariate post-process SPC is used to measure every manufactured part, which is an unrealistic assumption in practice and conceals some of its main weaknesses. It is assumed that the results could be less favorable for univariate post-process SPC when it is used the way it is in practice, especially regarding the detection of anomalies.

6.3. Summary

Chapter 6 presents the validation of SSMSPC from two different perspectives. In Section 6.1, the general performance is evaluated and compared to state-of-the-art AD baselines based on two real-world discrete manufacturing datasets. The first evaluation is conducted on the Bosch CNC milling dataset, which represents a recently published benchmark dataset for discrete manufacturing processes featuring vibration data from three different CNC milling machines. On this dataset, SSMSPC achieves a perfect score, being on par with PatchCore, LeNet-type AE, and Deep-SVDD. The effectiveness of the proposed control chart extension is demonstrated qualitatively by correctly identifying the anomalous process modes in the data. Furthermore, the performance of SSMSPC is evaluated in terms of precision, recall, and f1-score using various levels of α. It can be confirmed that, with predefined levels of the false alarm rate, SSMSPC achieves high recall. In addition to that, an ablation study is conducted in which the meaningfulness of the individual design choices of SSMSPC is assessed. It is shown that the presented default augmentation functions provide a reasonable choice for practical applications and that SSMSPC is fairly robust with respect to slight changes in the number of windows used for the augmentation process. Moreover, combining the location and transformation objectives in the Location + Transformation prediction pretext task is demonstrated to result in superior representations as opposed to using either objective alone. This substantiates the effectiveness of the proposed pretext task. The second evaluation centers around a subset of the CiP-DMD, a novel discrete manufacturing benchmark that has been recorded in the process learning factory CiP at the Technical University of Darmstadt. This dataset is in alignment with the problem statement that SSMSPC is designed to solve. It corresponds to a CNC milling process on a DMC 50H machine, in which high-frequency vibration data, as well as metadata are recorded. On this dataset, SSMSPC yields the highest overall score, outperforming all other baselines. As in the previous experiment, it is confirmed that SSMSPC achieves high recall values while maintaining low false alarm rates.
In Section 6.2, the usability of SSMSPC is evaluated and compared to univariate post-process SPC in the context of a usability study with human machine operators. In this study, the CNC milling process of the CiP-DMD is monitored with both SSMSPC and univariate post-process SPC. The usability is evaluated with the help of monitoring protocols as objective evaluation measures and the SUS as a subjective evaluation measure. The results show that the outstanding detection and localization capabilities provided by SSMSPC are not yet sufficient to effectively support the root-cause analysis. Instead, it is suggested to use these tools in conjunction with the measurements of a manufactured part obtained from a quality control step. Used in this way, SSMSPC is attributed higher usability than univariate post-process SPC, both from an objective and subjective point of view.
The validation conducted in this chapter indicates that SSMSPC satisfies all external requirements that are defined in Section 4.2.2, i.e., high recall, low false alarm rate, interpretability, and usability. In addition, it achieves superior detection performance compared to existing approaches. As such, it can be considered an approach of high practical relevance.

7. Conclusion

This chapter concludes the underlying thesis. Section 7.1 answers the research questions that are formulated in Section 1.1 and concretized in Section 4.1. Following this, Section 7.2 summarizes the scientific contributions. Lastly, Section 7.3 discusses the limitations and presents possible future research directions.

7.1. Answering the research questions

The objective of this thesis is the development of a novel approach for MSPC in discrete manufacturing processes based on SSL to detect and locate anomalies in multivariate process data. This approach is referred to as SSMSPC. Based on this research objective, three research questions are formulated that will be answered in the following.

Research question 1

How can SSL be incorporated into an MSPC system for discrete manufacturing processes?

The answer to this research question is reflected in the design of SSMSPC. As shown in Section 5.1, SSMSPC follows the two-stage framework presented by [Sohn21] in which the self-supervised representation learning phase is decoupled from the actual AD downstream task. Due to this decoupling, the two-stage framework is well suited to be embedded in phase I of the general SPC framework. This permits the application of established MSPC approaches based on the learned representations obtained from the self-supervised pretext task.
In SSMSPC, the self-supervised representation learning phase is represented by the proposed Location + Transformation prediction pretext task. This pretext task is shown to provide effective representations for the detection and localization of anomalies in high-frequency multivariate time series data as they occur in discrete manufacturing processes. Based on the learned representations, SSMSPC computes the Hotelling's T^2 statistic as the anomaly score in the AD downstream task and fits the control limits via a conventional KDE-based threshold selection scheme.

Research question 2

What are the benefits of SSL compared to existing shallow and DL approaches for MSPC in discrete manufacturing processes?

There are three major benefits to using SSL as opposed to existing shallow and deep approaches for MSPC in discrete manufacturing processes. First, the ability to incorporate domain knowledge into a handcrafted pretext task to learn domain-specific representations. With this ability, it is possible to tailor the pretext task to the needs of the downstream task. Second, employing a handcrafted pretext task that involves the classification of transformed or augmented instances of the normal class leads to an increase in the resulting training dataset size that is proportional to the number of augmentations. Since even the collection of normal data can become challenging in discrete manufacturing processes due to, e.g., excessive processing times, this feature is highly relevant. Third, the obtained performance levels. As is shown in Section 3.1, SSL approaches in general tend to outperform state-of-the-art shallow and deep AD baselines on various data types. In Section 6.1, it is demonstrated that based on the learned representations obtained from the Location + Transformation prediction pretext task, SSMSPC consistently outperforms leading shallow and deep baselines that can be used in the context of MSPC.

Research question 3

How can SSMSPC be applied in practice to replace univariate post-process SPC, which is accepted in industry and embodied in today's norms?

The results of the usability study in Section 6.2 provide a clear answer to this question. SSMSPC is shown to have superior performance when it comes to the detection and localization of anomalies. However, with respect to the root-cause analysis, it is demonstrated that these features might not yet be sufficient. In order to provide a suitable alternative to univariate post-process SPC, it is suggested to use SSMSPC in conjunction with a quality control step. Instead of sampling subgroups of manufactured parts at equidistant time intervals from the process that are then measured according to the univariate post-process SPC scheme, the process is continuously monitored with SSMSPC. Since the process data from each part are analyzed, this essentially corresponds to a 100% inspection of the process. The detection of an anomaly by SSMSPC serves as a signal to sample manufactured parts from the process that need to undergo a quality control step. Combining the insights from the control chart extension with the findings from quality control, the machine operator is well-equipped to identify the root cause of the anomaly. Using SSMSPC in this way allows rethinking the practical application of SPC in discrete manufacturing processes and provides the necessary means to complete the required paradigm shift from univariate post-process SPC to MSPC.

7.2. Summary of the scientific contributions

Having answered the research questions, the scientific contributions from each chapter are summarized in the following.
Chapter 2 presents the theoretical foundations across the three areas of research that are addressed with this thesis: (1) ML, (2) AD, and (3) SPC. In this context, the concept of SSL is identified as an emerging research field that provides the opportunity to learn effective representations from unlabelled raw data, matching or even surpassing the performance of fully supervised approaches. Incorporating SSL into AD establishes a new research field known as self-supervised AD that represents a major milestone in contemporary AD research, especially in terms of the semi-supervised AD setting. Following this, it is deduced that the general SPC framework essentially corresponds to a semi-supervised AD setting. This key insight provides the theoretical basis for considering SPC as a branch of ML research. Next, it is demonstrated that univariate post-process SPC, which is deeply rooted in ISO norms and embodies the state of the art in practical applications, neglects the emerging opportunities in modern discrete manufacturing environments brought on by digitization and Industry 4.0. In this context, the need for a paradigm shift from univariate post-process SPC to MSPC is elaborated. MSPC is shown to establish the connecting link between ML, AD, and SPC that provides the required tools to monitor discrete manufacturing processes based on multivariate process data during the process.

Chapter 3 presents the state of the art and derives the research gap. Based on the results of two separately conducted systematic literature reviews covering the fields of self-supervised AD from a fundamental ML research perspective and MSPC with respect to discrete and continuous manufacturing processes, it is demonstrated that SSL possesses enormous potential for MSPC. To this end, the application of SSL in the context of MSPC is identified as a novel research field that is still mostly unexplored and requires further attention from the research community. In contrast to the continuous manufacturing domain, in which one research article can be identified that incorporates SSL into MSPC, it is shown that there exists no such research with respect to discrete manufacturing processes yet. This key observation marks the research gap.
Chapter 4 outlines the overall objective of the thesis and formulates the external requirements that need to be satisfied by SSMSPC to be of practical relevance. It is demonstrated that in the discrete manufacturing domain, recall is generally more important than precision since missed anomalies pose a serious threat, resulting in unplanned downtimes or defective parts being passed to the customer undetected. At the same time, it is paramount to ensure a low number of false alarms to avoid unnecessary distractions, wasted time and resources. It is shown that, in order to support the root-cause analysis, the model decisions need to be interpretable by virtue of highlighting the identified anomalous regions in the process data. Based on the widening gap between academia and practice, it is concluded that the existing MSPC approaches are considered inappropriate by practitioners to replace univariate post-process SPC. To bridge this widening gap, SSMSPC must be attributed higher usability than univariate post-process SPC.
Chapter 5 introduces the three components of the proposed SSMSPC approach: (1) Location + Transformation prediction pretext task, (2) AD downstream task, and (3) control chart extension. These components represent the most important scientific contributions of this thesis. The objective of the Location + Transformation prediction pretext task is to classify the augmentation $\mathcal{T}_i$ and the window $\mathcal{W}_j$ based on a time series sample that is artificially augmented by one of k augmentation functions in one of p windows. In the AD downstream task, the learned representations from the pretext task are used to compute the anomaly score, which corresponds to the Hotelling's T^2 statistic. Based on the anomaly scores, the control limits are determined with the help of the KDE-based threshold selection scheme. In the context of the two-stage framework by [Sohn21], both the Location + Transformation prediction pretext task and the AD downstream task are embedded in phase I of the general SPC framework. The control chart extension provides an additional view to the control chart, in which the anomalous regions in the process data are highlighted. In this regard, the incorporation of metadata such as the current NC line is suggested to segment the time series data into the individual process modes. This step further enhances the interpretability of the control chart extension by pointing a machine operator directly to the process modes that are affected by an assignable cause.
Chapter 6 presents the validation of SSMSPC, both from a performance and usability perspective. In terms of the performance comparison, SSMSPC is evaluated and compared to state-of-the-art shallow, deep, and self-supervised AD baselines using two real-world discrete manufacturing datasets. The first dataset represents the Bosch CNC milling dataset, which is a recently published discrete manufacturing benchmark dataset featuring vibration data from three different CNC milling machines. SSMSPC is shown to yield a perfect score on this dataset in terms of AUROC and achieve high recall values while maintaining low levels of false alarms. The meaningfulness of the individual design choices are confirmed in an extensive ablation study. The second dataset is a subset of the CiP-DMD, a novel publicly available discrete manufacturing benchmark dataset that has been recorded at the Technical University of Darmstadt in the course of this thesis and corresponds to another CNC milling process featuring vibration data. SSMSPC yields the highest overall score in terms of AUROC and is demonstrated to achieve high recall with low false alarm rates. In the context of the usability study,

SSMSPC is evaluated and compared to univariate post-process SPC on the CNC milling process of the CiP-DMD with human machine operators. Usability is assessed both subjectively with the SUS and objectively using monitoring protocols. The results on the SUS indicate that the perceived usability of SSMSPC is above average and exceeds that of univariate post-process SPC by more than 25%. In terms of the monitoring protocols, it is demonstrated that the detection and localization performance of SSMSPC is outstanding. However, it is shown that the interpretation of the control chart extension alone might not be sufficient when it comes to providing effective support in identifying the root cause, as it requires additional domain expertise and experience to unfold its potential. To this end, it is suggested to combine the control chart extension of SSMSPC with the insights obtained from a quality control step of the manufactured part to exceed the usability of univariate post-process SPC, both objectively and subjectively. Overall, the results indicate that SSMSPC outperforms existing state-of-the-art AD approaches, which substantiates the effectiveness of its components. It fulfills the formulated external requirements and can, as such, be considered an approach of practical relevance that has the potential to replace univariate post-process SPC in discrete manufacturing processes.

7.3. Limitations and future work

SSMSPC, as presented in this thesis, is subject to three major limitations that need to be taken into account when considering its use in practical applications. These limitations are discussed below.
The first limitation concerns the proposed control chart extension. In its presented version, the control chart extension highlights the identified anomalous regions in the process data on a time step basis across all sensors at once, independent of whether the recorded process data of a given sensor are actually anomalous. Consequently, it does not explicitly provide information about which sensors detected the anomaly. This is due to the fact that the anomalous time steps are retrieved from the Grad-CAM heatmap in which the "depth" information is lost. Highlighting the anomalous regions and simultaneously discriminating between the sensors requires architectural changes.
The second limitation addresses the proposed data processing pipeline. SSMSPC retrieves the whole sequence of process data that correspond to a manufactured part as input. Thus, computational efforts rise tremendously with a growing number of sensors, higher sampling frequencies and longer processing times. For certain discrete manufacturing processes, this might become unfeasible and would require switching to a window-based approach, in which the process data are subdivided into smaller chunks that are then passed to SSMSPC.
The third limitation focuses on the presented evaluation. In terms of performance, SSMSPC is only evaluated on CNC milling processes that feature vibration data. While, from a theoretical point of view, any discrete manufacturing process that satisfies the internal requirements from Section 4.2.1 can be monitored with SSMSPC, it requires additional tests in practical applications to confirm this assumption. When interpreting the results of the usability study, three additional aspects need to be taken into account. The number of eight surveyed machine operators is small and does not permit a comprehensive statistical investigation. The machine operators are student workers who are experienced in using the DMC 50H machine but are not trained professional machine operators. Lastly, for reasons of feasibility, univariate post-process SPC is employed under unrealistic conditions that conceal many of its weaknesses, providing it with an unfair advantage over SSMSPC. It is therefore required to conduct additional and more extensive usability studies with experienced machine operators to provide a comprehensive assessment of the underlying value of the control chart extension.
In addition to the future work resulting from the presented limitations, four additional directions should be taken into account in the context of future research efforts.

First, as stated in Section 3.1, the application of contrastive learning schemes is continuously gaining momentum in self-supervised AD research. Since this form of SSL is shown to have a high impact on many real-world applications, it would be worthwhile to investigate the potential of contrastive learning for MSPC in discrete manufacturing processes.
Second, one of the internal requirements of SSMSPC is that the recorded process data are associated with the final quality characteristics of the manufactured part. According to [DIN3534-2], this requirement is essential for a control chart scheme to operate effectively. However, choosing the right process data in relation to quality characteristics requires careful analysis and strongly depends on the process at hand. From a research perspective, it would be beneficial to develop guidelines for practitioners that provide criteria for this decision.
Third, additional research efforts need to be undertaken with respect to the explainability of existing and newly developed monitoring approaches. As is shown in this thesis, reliably detecting anomalies is a necessary but insufficient condition for practical applications. In order to cooperate effectively, model decisions need to be interpretable and provide effective support in the problem-solving process. This feature is surmised to be of essential relevance for the widespread adoption of ML in manufacturing.
Fourth, one essential topic that is neglected in this thesis is that of uncertainty estimation. In light of, e.g., distribution shifts, it is crucial for deployed monitoring approaches to be able to quantify the uncertainty of their decisions in order to gain acceptance and trust of practitioners as well as to provide a signal when it is required to retrain a model.
All in all, the results presented in this thesis provide important contributions that advance the research field of MSPC in discrete manufacturing and narrow the gap between practice and academia.

Bibliography

Articles and books

[Abad15] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015.

[Abat19] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara. “Latent Space Autoregression for Novelty Detection”. In: *Conference on Computer Vision and Pattern Recognition* (2019), pp. 481–490.

[Abdu21] A. Abdulaal, Z. Liu, and T. Lancewicki. “Practical Approach to Asynchronous Multivariate Time Series Anomaly Detection and Localization”. In: *KDD* (2021), pp. 2485–2494.

[Agga17] C. C. Aggarwal. *Outlier Analysis*. Cham: Springer International Publishing, 2017.

[Ahma20] S. Ahmad, N. Enshaei, F. Naderkhani, and A. Awasthi. “Integrated Deep Learning and Statistical Process Control for Online Monitoring of Manufacturing Processes”. In: *IEEE International Conference on Prognostics and Health Management* (2020), pp. 1–6.

[Ahma21] M. R. Ahmad and S. E. Ahmed. “On the distribution of the T2 statistic, used in statistical process monitoring, for high-dimensional data”. In: *Statistics & Probability Letters* 168 (2021), p. 108919.

[Ahn22] K. Ahn, J. Zhang, and S. Sra. “Understanding the Unstable Convergence of Gradient Descent”. In: *International Conference on Machine Learning* (2022).

[Ai23] M. Ai, Y. Xie, S. X. Ding, Z. Tang, and W. Gui. “Domain Knowledge Distillation and Supervised Contrastive Learning for Industrial Process Monitoring”. In: *IEEE Transactions on Industrial Electronics* 70.9 (2023), pp. 9452–9462.

[Akba21] H. Akbari, L. Yuan, R. Qian, W.-H. Chuang, S.-F. Chang, Y. Cui, and B. Gong. “VATT: Transformers for Multimodal Self-Supervised Learning from Raw Video, Audio and Text”. In: *Conference on Neural Information Processing Systems* 34 (2021).

[Alay20] J.-B. Alayrac, A. Recasens, R. Schneider, R. Arandjelović, J. Ramapuram, J. D. Fauw, L. Smaira, S. Dieleman, and A. Zisserman. “Self-Supervised MultiModal Versatile Networks”. In: *Conference on Neural Information Processing Systems* (2020).

[Alwa20] H. Alwassel, D. Mahajan, B. Korbar, L. Torresani, B. Ghanem, and Du Tran. “Self-Supervised Learning by Cross-Modal Audio-Video Clustering”. In: *Conference on Neural Information Processing Systems* (2020).

[Ande03] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. 3rd ed. Wiley Series in Probability and Statistics. Hoboken NJ: Wiley, 2003.

[Bach19] P. Bachman, R. D. Hjelm, and W. Buchwalter. “Learning Representations by Maximizing Mutual Information Across Views”. In: *Conference on Neural Information Processing Systems* (2019), pp. 15535–15545.

[Baev20] A. Baevski, H. Zhou, A. Mohamed, and M. Auli. “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations”. In: *Conference on Neural Information Processing Systems* (2020).

[Barb12] D. Barber. *Bayesian Reasoning and Machine Learning*. New York: Cambridge University Press, 2012.

[Bard22] A. Bardes, J. Ponce, and Y. Lecun. “VICReg: Variance-Invariance-Covariance Regularization for self-supervised learning”. In: *International Conference on Learning Representations* (2022).

[Bayd18] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. “Automatic Differentiation in Machine Learning: A Survey”. In: *Journal of Machine Learning Research* (2018), pp. 1–43.

[Beck92] S. Becker and G. E. Hinton. “Self-organizing neural network that discovers surfaces in random-dot stereograms”. In: *Nature* 355 (1992), pp. 161–163.

[Beng07a] Y. Bengio and Y. Lecun. “Scaling Learning Algorithms towards AI”. English (US). In: *Large-scale kernel machines*. MIT Press, 2007.

[Beng07b] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. “Greedy Layer-Wise Training of Deep Networks”. In: *Conference on Neural Information Processing Systems* 19 (2007), pp. 153–160.

[Beng09] Y. Bengio. “Learning Deep Architectures for AI”. In: *Foundations and Trends® in Machine Learning* 2.1 (2009), pp. 1–127.

[Beng11] Y. Bengio and O. Delalleau. “On the Expressive Power of Deep Architectures”. In: *Algorithmic Learning Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 18–36.

[Beng12] Y. Bengio. “Practical Recommendations for Gradient-Based Training of Deep Architectures”. In: *Neural Networks: Tricks of the Trade*. 2012, pp. 437–478.

[Beng13a] Y. Bengio. “Deep Learning of Representations: Looking Forward”. In: *Statistical Language and Speech Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–37.

[Beng13b] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai. “Better Mixing via Deep Representations”. In: *International Conference on Machine Learning* (2013), pp. 552–560.

[Beng13c] Y. Bengio, A. Courville, and P. Vincent. “Representation learning: a review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[Berg19] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. “MVTec AD – A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection”. In: *Conference on Computer Vision and Pattern Recognition* (2019), pp. 9592–9600.

[Berg20a] L. Bergman and Y. Hoshen. “Classification-Based Anomaly Detection For General Data”. In: *International Conference on Learning Representations* (2020).

[Berg20b] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. “Uninformed Students: Student-Teacher Anomaly Detection With Discriminative Latent Embeddings”. In: *Conference on Computer Vision and Pattern Recognition* (2020), pp. 4183–4192.

[Bers07] S. Bersimis, S. Psarakis, and J. Panaretos. “Multivariate Statistical Process Control Charts: An Overview”. In: *Quality AND Reliability Engineering International* 23.5 (2007), pp. 517–543.

[Bhun21] A. K. Bhunia, P. N. Chowdhury, Y. Yang, T. M. Hospedales, T. Xiang, and Y.-Z. Song. “Vectorization and Rasterization: Self-Supervised Learning for Sketch and Handwriting”. In: *International Conference on Computer Vision and Pattern Recognition* (2021), pp. 5672–5681.

[Bieg22a] T. Biegel, N. Jourdan, C. Hernandez, A. Cviko, and J. Metternich. “Deep learning for multivariate statistical in-process control in discrete manufacturing: A case study in a sheet-metal forming process”. In: *Procedia CIRP* (2022), pp. 422–427.

[Bieg22b] T. Biegel, N. Jourdan, T. Madreiter, L. Kohl, S. Fahle, F. Ansari, B. Kuhlenkötter, and J. Metternich. “Combining process monitoring with text mining for anomaly detection in discrete manufacturing”. In: *SSRN* (2022).

[Bieg23] T. Biegel, P. Helm, N. Jourdan, and J. Metternich. “SSMSPC: self-supervised multivariate statistical in-process control in discrete manufacturing processes”. In: *Journal of Intelligent Manufacturing* (2023). URL: `http://creativecommons.org/licenses/by/4.0/`.

[Bisg12] S. Bisgaard. “The Future of Quality Technology: From a Manufacturing to a Knowledge Economy & From Defects to Innovations”. In: *Quality Engineering* 24.1 (2012), pp. 30–36.

[Bish06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information science and statistics. New York NY: Springer, 2006.

[Bish94] C. M. Bishop. “Neural networks and their applications”. In: *Review of Scientific Instruments* 65.6 (1994), pp. 1803–1832.

[Bish96] C. M. Bishop. *Neural Networks for Pattern Recognition*. USA: Oxford University Press, 1996.

[Bjor18] J. Bjorck, C. Gomes, S. Bart, and K. Q. Weinberger. “Understanding Batch Normalization”. In: *Conference on Neural Information Processing Systems* (2018).

[Bláz21] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano. “A review on outlier/anomaly detection in time series data”. In: *ACM Computing Surveys* (2021).

[Blit15] J. K. Blitzstein and J. Hwang. *Introduction to Probability*. CRC Press, 2015.

[Bott12] L. Bottou. “Stochastic Gradient Descent Tricks”. In: *Neural Networks: Tricks of the Trade*. 2012, pp. 421–436.

[Bour10a] Y.-L. Boureau, F. Bach, Y. Lecun, and J. Ponce. “Learning Mid-Level Features for Recognition”. In: *Conference on Computer Vision and Pattern Recognition* (2010).

[Bour10b] Y.-L. Boureau, J. Ponce, and Y. Lecun. “A Theoretical Analysis of Feature Pooling in Visual Recognition”. In: *International Conference on Machine Learning* (2010).

[Boyd04] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge UK and New York: Cambridge University Press, 2004.

[Brad97] A. P. Bradley. “The use of the area under the ROC Curve in the evaluation of machine learning algorithms”. In: *Pattern Recognition* (1997), pp. 1145–1159.

[Breu00] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. “LOF: Identyfiying Denstiy-Based Local Outliers”. In: *ACM SIGMOD International Conference on Management of Data* (2000).

[Broo13] J. Brooke. “SUS: A Retrospective”. In: *Journal of Usability Studies* (2013), pp. 29–40.

[Broo96] J. Brooke. “SUS: A Quick and Dirty Usability Scale”. In: *Usability Evaluation in Industry*. 1996.

[Bukc21] G. Bukchin, E. Schwartz, K. Saenko, O. Shahar, R. Feris, R. Giryes, and L. Karlinsky. “Fine-Grained Angular Contrastive Learning With Coarse Labels”. In: *International Conference on Computer Vision and Pattern Recognition* (2021), pp. 8730–8740.

[Camp16] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle. “On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study”. In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 891–927.

[Caro18] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. “Deep Clustering for Unsupervised Learning of Visual Features”. In: *European Conference on Computer Vision* (2018).

[Caro21] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. “Emerging Properties in Self-Supervised Vision Transformers”. In: *International Conference on Computer Vision* (2021), pp. 9650–9660.

[Carr20] F. Carrara, G. Amato, L. Brombin, F. Falchi, and C. Gennaro. “Combining GANs and AutoEncoders for Efficient Anomaly Detection”. In: *International Conference on Pattern Recognition* (2020), pp. 3939–3946.

[Chal19] R. Chalapathy and S. Chawla. *Deep Learning for Anomaly Detection: A Survey*. 2019. arXiv: `1901.03407v2`.

[Chan09] V. Chandola, A. Banerjee, and V. Kumar. “Anomaly Detection: A Survey”. In: *ACM Computing Surveys* 41.3 (2009), pp. 1–58.

[Chan10] S. I. Chang and S. Yadama. “Statistical process control for monitoring non-linear profiles using wavelet filtering and B-Spline approximation”. In: *International Journal of Production Research* 48.4 (2010), pp. 1049–1068.

[Chat95] C. Chatfield. *The Analysis of Time Series an Introduction*. 5th ed. Chapman and Hall, 1995.

[Chen19] F. Cheng, Q. P. He, and J. Zhao. “A novel process monitoring approach based on variational recurrent autoencoder”. In: *Computers & Chemical Engineering* 129 (2019), p. 106515.

[Chen20] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *International Conference on Machine Learning* (2020).

[Chen21a] C. Chen, Y. Xie, S. Lin, R. Qiao, J. Zhou, X. Tan, Y. Zhang, and L. Ma. “Novelty Detection via Contrastive Learning with Negative Data Augmentation”. In: *International Joint Conference on Artificial Intelligence* 1 (2021), pp. 606–614.

[Chen21b] T. Chen, J. Frankle, S. Chang, S. Liu, Y. Zhang, M. Carbin, and Z. Wang. “The Lottery Tickets Hypothesis for Supervised and Self-Supervised Pre-Training in Computer Vision Models”. In: *International Conference on Computer Vision and Pattern Recognition* (2021), pp. 16306–16316.

[Chen21c] T. Chen, C. Luo, and L. Li. “Intriguing Properties of Contrastive Losses”. In: *Conference on Neural Information Processing Systems* 34 (2021).

[Chen21d] X. Chen and K. He. “Exploring Simple Siamese Representation Learning”. In: *International Conference on Computer Vision and Pattern Recognition* (2021).

[Chen21e] X. Chen, S. Xie, and K. He. “An Empirical Study of Training Self-Supervised Vision Transformers”. In: *International Conference on Computer Vision* (2021), pp. 9640–9649.

[Chen21f] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng. “Learning Graph Structures with Transformer for Multivariate Time Series Anomaly Detection in IoT”. In: *IEEE Internet of Things Journal* (2021), pp. 1–11.

[Chen94] B. Cheng and D. M. Titterington. “Neural Networks: A Review from a Statistical Perspective”. In: *Statistical Science* (1994), pp. 2–30.

[Chia01] L. H. Chiang, E. L. Russell, and R. D. Braatz. *Fault Detection and Diagnosis in Industrial Systems*. Advanced Textbooks in Control and Signal Processing. London: Springer London, 2001.

[Cho21] H. Cho, J. Seol, and S.-g. Lee. “Masked Contrastive Learning for Anomaly Detection”. In: *International Joint Conference on Artificial Intelligence* (2021), pp. 1434–1441.

[Chol17] F. Chollet. *Deep learning with Python*. Manning Publications Co., 2017.

[Chui22] M. Chui, B. Hall, H. Mayhew, A. Singla, and A. Sukharevsky. *The state of AI in 2022 - and half a decade in review, McKinsey & Company*. Technical report. Dec. 2022.

[Chui23a] M. Chui, I. Mena, R. Roberts, and L. Yee. *Technology Trends Outlook 2023, McKinsey & Company*. Technical report. July 2023.

[Chui23b] M. Chui, L. Yee, B. Hall, A. Singla, and A. Sukharevsky. *The state of AI in 2023: Generative AI’s breakout year, McKinsey & Company*. Technical report. Aug. 2023.

[Cogs16] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra. “Reducing Overfitting in Deep Networks by Decorrelating Representations”. In: *International Conference on Learning Representations* (2016).

[Cort95] C. Cortes and V. Vapnik. “Support-Vector Networks”. In: *Machine Learning* (1995), pp. 273–297.

[Cove67] T. M. Cover and P. E. Hart. “Nearest Neighbor Pattern Classification”. In: *IEEE Transactions on information theory* (1967), pp. 21–27.

[Cybe89] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals and Systems* (1989), pp. 303–314.

[Dai22] E. Dai and J. Chen. “Graph-Augmented Normalizing Flows for Anomaly Detection of Multiple Time Series”. In: *International Conference on Learning Representations* (2022).

[Dark92] C. Darken, J. Chang, and J. Moody. “Learning rate schedules for faster stochastic gradient search”. In: *Neural Networks for Signal Processing* (1992), pp. 3–12.

[Deec19] L. Deecke, R. A. Vandermeulen, L. Ruff, S. Mandt, and M. Kloft. “Image Anomaly Detection with Generative Adverserial Networks”. In: *Machine Learning and Knowledge Discovery in Databases* (2019).

[Deec21] L. Deecke, L. Ruff, R. A. Vandermeulen, and H. Bilen. “Transfer-Based Semantic Anomaly Detection”. In: *International Conference on Machine Learning* (2021), pp. 2546–2558.

[Deha20] D. Dehaene, O. Frigo, S. Combrexelle, and P. Eline. “Iterative energy-based projection on a normal data manifold for anomaly localization”. In: *International Conference on Learning Representations* (2020).

[Deis20] M. P. Deisenroth, A. A. Faisal, and C. S. Ong. *Mathematics for Machine Learning*. Cambridge: Cambridge University Press, 2020.

[Deng21] A. Deng and B. Hooi. “Graph Neural Network-Based Anomaly Detection in Multivariate Time Series”. In: *AAAI Conference on Artifical Intelligence* (2021).

[Di M16] M. Di Mauro, M. F. Maggioni, M. Grasso, and B. M. Colosimo. “Design performance analysis of a Self-Organizing Map for statistical monitoring of distribution-free data streams”. In: *Procedia CIRP* 41 (2016), pp. 448–453.

[Diet91] B. L. Dietrich. “A Taxonomy of Discrete Manufacturing Systems”. In: *Operations Research* (1991), pp. 886–902.

[Ding14] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire. “An experimental evaluation of novelty detection methods”. In: *Neurocomputing* 135 (2014), pp. 313–327.

[Doer15] C. Doersch, A. Gupta, and A. Efros. “Unsupervised Visual Representation Learning by Context Prediction”. In: *International Conference on Computer Vision* (2015), pp. 1422–1430.

[Doer17] C. Doersch and A. Zissermann. “Multi-task Self-Supervised Visual Learning”. In: *International Conference on Computer Vision* (2017), pp. 2070–2079.

[Down93] J. Downs and E. Vogel. “A plant-wide industrial process control problem”. In: *Computers & Chemical Engineering* 17.3 (1993), pp. 245–255.

[Duch11] J. Duchi, E. Hazan, and Y. Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* (2011), pp. 2121–2159.

[Dwib21] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. “With a Little Help From My Friends: Nearest-Neighbor Contrastive Learning of Visual Representations”. In: *International Conference on Computer Vision* (2021), pp. 9588–9597.

[Edge87] F. Y. Edgeworth. “On discordant observations”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* (1887), pp. 364–375.

[Ermo21] A. Ermolov, A. Siarohin, E. Sangineto, and N. Sebe. “Whitening for Self-Supervised Representation Learning”. In: *International Conference on Machine Learning* (2021), pp. 3015–3024.

[Fan21] L. Fan, S. Liu, P.-Y. Chen, G. Zhang, and C. Gan. “When does Contrastive Learning Preserve Adversarial Robustness from Pretraining to Finetuning?” In: *Conference on Neural Information Processing Systems* 34 (2021).

[Fath21] S. Fathizadan, F. Ju, and Y. Lu. “Deep representation learning for process variation management in laser powder bed fusion”. In: *Additive Manufacturing* 42 (2021), p. 101961.

[Fawc06] T. Fawcett. “An introduction to ROC analysis”. In: *Pattern Recognition Letters* 27.8 (2006), pp. 861–874.

[Ferr07] A. Ferrer. “Multivariate Statistical Process Control Based on Principal Component Analysis (MSPC-PCA): Some Reflections and a Case Study in an Autobody Assembly Process”. In: *Quality Engineering* 19.4 (2007), pp. 311–325.

[Ferr14] A. Ferrer. “Latent Structures-Based Multivariate Statistical Process Control: A Paradigm Shift”. In: *Quality Engineering* 26.1 (2014), pp. 72–91.

[Fix51] E. Fix and J. L. Hodges. *Discriminatory analysis, nonparametric discrimination*. Ed. by USAF School of Aviation Medicine, Randolph Field, Tex., Project 21-49-004, Rept. 4, Contract AF41(128)-31. 1951.

[Flac05] P. A. Flach and S. Wu. “Repairing Concavities in ROC Curves”. In: *International Joint Conference on Artificial Intelligence* (2005), pp. 702–705.

[Fu22] Y. Fu and F. Xue. “MAD: Self-Supervised Masked Anomaly Detection Task for Multivariate Time Series”. In: *International Joint Conference on Neural Networks* (2022).

[Fuku80] K. Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics* (1980), pp. 193–202.

[Funa89] K. Funahashi. “On the Approximate Realization of Continuous Mappings by Neural Networks”. In: *Neural Networks* (1989), pp. 183–192.

[Gao11] R. X. Gao and R. Yan. “Continuous Wavelet Transform”. In: *Wavelets: Theory and Applications for Manufacturing*. Boston, MA: Springer US, 2011.

[Ge13] Z. Ge and Z. Song. *Multivariate Statistical Process Control: Process Monitoring Methods and Applications*. Advances in Industrial Control. London: Springer London, 2013.

[Geor21] M.-I. Georgescu, A. Barbalau, R. T. Ionescu, F. S. Khan, M. Popescu, and M. Shah. “Anomaly Detection in Video via Self-Supervised and Multi-Task Learning”. In: *Conference on Computer Vision and Pattern Recognition* (2021), pp. 12742–12752.

[Gero19] A. Geron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2nd ed. Sebastopol CA: O’Reilly Media Inc, 2019.

[Gida18] S. Gidaris, P. Singh, and N. Komodakis. “Unsupervised Representation Learning by Predicting Image Rotations”. In: *International Conference on Learning Representations* (2018).

[Giro95] F. Girosi, M. Jones, and T. Poggio. “Regularization Theory and Neural Networks Architectures”. In: *Neural computation* 7.2 (1995), pp. 219–269.

[Glor11] X. Glorot, A. Bordes, and Y. Bengio. “Deep Sparse Rectifier Neural Network”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings* 15.315-323 (2011).

[Gola18] I. Golan and R. El-Yaniv. “Deep Anomaly Detection Using Geometric Transformations”. In: *Conference on Neural Information Processing Systems* (2018).

[Good14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative Adversarial Networks”. In: *Conference on Neural Information Processing Systems* (2014).

[Good16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Cambridge MA: MIT Press, 2016.

[Good20] A. Goodge, B. Hooi, S. K. Ng, and W. S. Ng. “Robustness of Autoencoders for Anomaly Detection Under Adversarial Impact”. In: *International Joint Conference on Artificial Intelligence* 2 (2020), pp. 1244–1250.

[Gopa19] P. Gopalan, V. Sharan, and U. Wieder. “PIDForest: Anomaly Detection via Partial Identification”. In: *Conference on Neural Information Processing Systems* 32 (2019).

[Goup84] P. Goupillaud, A. Grossmann, and J. Morlet. “Cycle-octave and related transforms in seismic signal analysis”. In: *Geoexploration* (1984), pp. 85–102.

[Goya20] S. Goyal, A. Raghunathan, M. Jain, H. V. Simhadri, and P. Jain. “DROCC: Deep Robust One-Class Classification”. In: *International Conference on Machine Learning* (2020).

[Gras14] M. Grasso, B. M. Colosimo, and M. Pacella. “Profile monitoring via sensor fusion: the use of PCA methods for multi-channel data”. In: *International Journal of Production Research* 52.20 (2014), pp. 6110–6135.

[Gras15a] M. Grasso. “Profile Monitoring of Multi-Stream Sensor Data”. PhD thesis. Mailand: Politecnico di Milano, 2015.

[Gras15b] M. Grasso, B. M. Colosimo, and F. Tsung. “Quality Monitoring of Multimode Processes via Signal Data”. In: *AITeM Conference* (2015).

[Gril20] J.-B. Grill et al. “Bootstrap your own latent: A new approach to self-supervised Learning”. In: *Conference on Neural Information Processing Systems* (2020), pp. 21271–21284.

[Grin97] C. M. Grinstead and J. L. Snell. *Introduction to Probability*. Providence RI: American Mathematical Society, 1997.

[Gros84] A. Grossmann and J. Morlet. “Decomposition of Hardy Functions into Square Integrable Wavelets of Constant Shape”. In: *SIAM Journal on Mathematical Analysis* 15.4 (1984), pp. 723–736.

[Gu18] J. Gu et al. “Recent Advances in Convolutional Neural Networks”. In: *Pattern Recognition* (2018), pp. 354–377.

[Gu19] X. Gu, L. Akoglu, and A. Rinaldo. “Statistical Analysis of Nearest Neighbor Methods for Anomaly Detection”. In: *Conference on Neural Information Processing Systems* 32 (2019).

[Guh07] R.-S. Guh. “On-line Identification and Quantification of Mean Shifts in Bivariate Processes using a Neural Network-based Approach”. In: *Quality and Reliability Engineering International* 23.3 (2007), pp. 367–385.

[Gupt14] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han. “Outlier Detection for Temporal Data: A Survey”. In: *IEEE Transactions on Knowledge and Data Engineering* (2014), pp. 2250–2267.

[Hads06] R. Hadsell, S. Chopra, and Y. Lecun. “Dimensionaltiy Reduction by Learning an Invariant Mapping”. In: *International Conference on Computer Vision and Pattern Recognition* (2006), pp. 1735–1742.

[Hami94] J. D. Hamilton. *Time-series analysis*. Princeton University Press, 1994.

[Hanl82] J. A. Hanley and B. J. McNeis. “The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve”. In: *Radiology* (1982), pp. 29–36.

[Hans88] S. J. Hanson and L. Y. Pratt. “Comparing Biases for Minimal Network Construction with Back-Propagation”. In: *Conference on Neural Information Processing Systems* (1988).

[Hard16] M. Hardt, B. Recht, and Y. Singer. “Train faster, generalize better: Stability of stochastic gradient descent”. In: *International Conference on Machine Learning* (2016).

[Harr20] F. Harrou, Y. Sun, A. S. Hering, M. Madakyaru, and A. Dairi. *Statistical Process Monitoring using Advanced Data-Driven and Deep Learning Approaches: Theory and Practical Applications*. Amsterdam: Elsevier, 2020.

[Hast08] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics. New York: Springer, 2008.

[He15] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *International Conference on Computer Vision and Pattern Recognition* (2015), pp. 770–778.

[He18] X. He, Y. Zhou, Z. Zhou, S. Bai, and X. Bai. “Triplet-Center Loss for Multi-View 3D Object Retrieval”. In: *Conference on Computer Vision and Pattern Recognition* (2018).

[Hend19] D. Hendrycks, M. Mazeika, and T. Dietterich. “Deep Anomaly Detection with Outlier Exposure”. In: *International Conference on Learning Representations* (2019).

[Hint06] G. E. Hinton, S. Osindero, and Y.-W. Teh. “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[Hint12] G. Hinton et al. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97.

[Horn89] K. Hornik, M. Stinchcombe, and H. White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* (1989), pp. 359–366.

[Hote47] H. Hotelling. “Multivariate quality control, illustrated by the air testing of sample bombsights”. In: *Techniques of statistical analysis* (1947), pp. 111–184.

[Hou21] J. Hou, Y. Zhang, Q. Zhong, Di Xie, S. Pu, and H. Zhou. “Divide-and-Assemble: Learning Block-Wise Memory for Unsupervised Anomaly Detection”. In: *International Conference on Computer Vision* (2021), pp. 8791–8800.

[Hu20] W. Hu, M. Wang, Q. Qin, J. Ma, and B. Liu. “HRN: A Holistic Approach to One Class Learning”. In: *Conference on Neural Information Processing Systems* (2020).

[Huan23] L. Huang, J. Qin, Y. Zhou, F. Zhu, L. Liu, and L. Shao. “Normalization Techniques in Training DNNs: Methodology, Analysis and Application”. In: *IEEE transactions on pattern analysis and machine intelligence* (2023).

[Hube62] D. H. Hubel and T. N. Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *Journal of Physiology* (1962), pp. 106–154.

[Hübn20] H. B. Hübner, M. A. V. Duarte, and R. B. Da Silva. “Automatic grinding burn recognition based on time-frequency analysis and convolutional neural networks”. In: *The International Journal of Advanced Manufacturing Technology* 110.7-8 (2020), pp. 1833–1849.

[Ibra23] M. Ibrahim and R. Balestriero. *The self-supervised learning cookbook*. Meta AI. Apr. 2023. URL: `https://ai.meta.com/blog/self-supervised-learning-practical-guide/` (visited on 08/28/2023).

[Ioff15] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *International Conference on Machine Learning* (2015), pp. 448–456.

[Jack85] J. E. Jackson. “Multivariate quality control”. In: *Communications in Statistics - Theory and Methods* 14.11 (1985), pp. 2657–2688.

[Jack91] J. E. Jackson. *A User’s Guide to Principal Components*. New York: John Wiley, 1991.

[Jame13] G. James, D. Witten, T. Hastie, and T. Robert. *An Introduction to Statistical Learning*. 1st ed. Springer Texts in Statistics. New York: Springer, 2013.

[Jarr09] K. Jarret, K. Kavukcuoglu, M. A. Ranzato, and Y. Lecun. “What is the Best Multi-Stage Architecture for Object Recognition?” In: *International Conference on Computer Vision* (2009), pp. 2146–2153.

[Jian20] Z. Jiang, T. Chen, T. Chen, and Z. Wang. “Robust Pre-Training by Adversarial Contrastive Learning”. In: *Conference on Neural Information Processing Systems* (2020).

[Jian21] Z. Jiang, T. Chen, T. Chen, and Z. Wang. “Improving Contrastive Learning on Imbalanced Seed Data via Open-World Sampling”. In: *Conference on Neural Information Processing Systems* (2021).

[Jin01] J. Jin and J. Shi. “Automatic feature extraction of waveform signals for in-process diagnostic performance improvement”. In: *Journal of Intelligent Manufacturing* 12 (2001), pp. 257–268.

[Jing20] L. Jing and Y. Tian. “Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey”. In: *IEEE transactions on pattern analysis and machine intelligence* (2020), pp. 4037–4058.

[Jone14] L. A. Jones-Farmer, W. H. Woodall, S. H. Steiner, and C. W. Champ. “An Overview of Phase I Analysis for Process Improvement and Monitoring”. In: *Journal of Quality Technology* (2014), pp. 265–280.

[Jone21] C. L. Jones, A.-S. G. Abdel-Salam, and D. Mays. “Practitioners guide on parametric, nonparametric, and semiparametric profile monitoring”. In: *Quality and Reliability Engineering International* 37.3 (2021), pp. 857–881.

[Jour21a] N. Jourdan, L. Longard, T. Biegel, and J. Metternich. “Machine Learning for intelligent maintenance and quality control: A review of existing datasets and corresponding use cases”. In: *Conference on Production Systems and Logistics* (2021).

[Jour21b] N. Jourdan, S. Sen, E. J. Husom, E. Garcia-Ceja, T. Biegel, and J. Metternich. “On the reliability of machine learning applications in manufacturing environments”. In: *Conference on Neural Information Processing Systems, Workshop on Distribution Shifts* (2021).

[Jour23] N. Jourdan, T. Biegel, B. B. Cassoli, and J. Metternich. “A new benchmark dataset for machine learning applications in discrete manufacturing: CiP-DMD [Manuscript accepted for publication]”. 2023.

[Kete15] B. de Ketelaere, M. Hubert, and E. Schmitt. “Overview of PCA-based statistical process monitoring methods for time-dependent, high-dimensional data”. In: *Journal of Quality Technology* 47.4 (2015), pp. 318–335.

[Khan14] S. S. Khan and M. G. Madden. “One-Class Classification: Taxonomy of Study and Review of Techniques”. In: *The Knowledge Engineering Review* 29.3 (2014), pp. 345–374.

[Kim20] K. Kim, S. Shim, Y. Lim, J. Jeon, J. Choi, B. Kim, and A. S. Yoon. “RaPP: Novelty Detection with Reconstruction along Projection Pathway”. In: *International Conference on Learning Representations* (2020).

[King14] D. P. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations* (2014).

[King15] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (2015).

[Kole19] A. Kolesnikov, X. Zhai, and L. Beyer. “Revisiting Self-Supervised Visual Representation Learning”. In: *International Conference on Computer Vision and Pattern Recognition* (2019), pp. 1920–1929.

[Kong20] D. Kong and X. Yan. “Industrial process deep feature representation by regularization strategy autoencoders for process monitoring”. In: *Measurement Science and Technology* 31.2 (2020).

[Kour02] T. Kourti. “Process analysis and abnormal situation detection: from theory to practice”. In: *IEEE Control Systems* 22.5 (2002), pp. 10–25.

[Kour95] T. Kourti and J. F. MacGregor. “Process analysis, monitoring and diagnosis, using multivariate projection methods”. In: *Chemometrics and Intelligent Laboratory Systems* 28.1 (1995), pp. 3–21.

[Kour96] T. Kourti and J. F. MacGregor. “Multivariate SPC Methods for Process and Product Monitoring”. In: *Journal of Quality Technology* 28.4 (1996), pp. 409–428.

[Kriz12] A. Krizhevsky, I. Sutskever, and G. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Conference on Neural Information Processing Systems* 25 (2012), pp. 1097–1105.

[Krog91] A. Krogh and J. A. Hertz. “A Simple Weight Decay Can Improve Generalization”. In: *Conference on Neural Information Processing Systems* (1991).

[Kuhn13] M. Kuhn and K. Johnson. *Applied predictive modeling*. New York NY: Springer, 2013.

[Kuhn19] M. Kuhn and K. Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Data Science Series. Boca Raton FL: Taylor & Francis Group, 2019.

[Kuma19] A. Kumagai, T. Iwata, and Y. Fujiwara. “Transfer Anomaly Detection by Inferring Latent Domain Representations”. In: *Conference on Neural Information Processing Systems* 32 (2019).

[Lai20] C.-H. Lai, D. Zou, and G. Lerman. “Robust Subspace Recovery Layer for Unsupervised Anomaly Detection”. In: *International Conference on Learning Representations* (2020).

[Lawr00] S. Lawrence and L. C. Giles. “Overfitting and Neural Networks: Conjugate Gradient and Backpropagation”. In: *International Joint Conference on Neural Networks* (2000), pp. 114–119.

[Lecu10] Y. Lecun, K. Kavukcuoglu, and C. Farabet. “Convolutional Networks and Applications in Vision”. In: *IEEE International Symposium on Circuits and Systems* (2010), pp. 253–256.

[Lecu15] Y. Lecun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[Lecu21] Y. Lecun and I. Misra. *Self-supervised learning: The dark matter of intelligence*. Meta AI. Mar. 2021. URL: `https://ai.meta.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/` (visited on 08/29/2023).

[Lecu89] Y. Lecun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. “Handwritten Digit Recognition with a Back-Propagation Network”. In: *Conference on Neural Information Processing Systems* 2 (1989), pp. 396–404.

[Lecu98a] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[Lecu98b] Y. Lecun, L. Bottou, G. B. Orr, and K.-R. Müller. “Efficient BackProp”. In: *Neural Networks: Tricks of the Trade* (1998).

[Lee19] S. Lee, M. Kwak, K.-L. Tsui, and S. B. Kim. “Process monitoring using variational autoencoder for high-dimensional nonlinear processes”. In: *Engineering Applications of Artificial Intelligence* 83 (2019), pp. 13–27.

[Lee21] C.-K. Lee, Y.-J. Cheon, and W.-Y. Hwang. “Studies on the GAN-Based Anomaly Detection Methods for the Time Series Data”. In: *IEEE Access* 9 (2021), pp. 73201–73215.

[Lejo18] E. Lejon, P. Kyösti, and J. Lindström. “Machine learning for detection of anomalies in press-hardening: Selection of efficient methods”. In: *Procedia CIRP* 72 (2018), pp. 1079–1083.

[Li20] Z. Li and S. Arora. “An Exponential Learning Rate Schedule For Deep Learning”. In: *International Conference on Learning Representations* (2020).

[Li21a] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister. “CutPaste: Self-Supervised Learning for Anomaly Detection and Localization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2021).

[Li21b] Z. Li, Y. Zhao, J. Han, Y. Su, R. Jiao, X. Wen, and D. Pei. “Multivariate Time Series Anomaly Detection and Interpretation using Hierarchical Inter-Metric and Temporal Embedding”. In: *KDD* (2021), pp. 3220–3230.

[Li22] S. Li, J. Luo, and Y. Hu. “Toward Interpretable Process Monitoring: Slow Feature Analysis-Aided Autoencoder for Spatiotemporal Process Feature Learning”. In: *IEEE Transactions on Instrumentation and Measurement)* 71 (2022), pp. 1–11.

[Li23] D. Li, J. Lu, T. Zhang, and J. Ding. “Self-Supervised Learning and Multisource Heterogeneous Information Fusion Based Quality Anomaly Detection for Heavy-Plate Shape”. In: *IEEE Transactions on Automation Science and Engineering* (2023), pp. 1–12.

[Liao21] Y. Liao, I. Ragai, Z. Huang, and S. Kerner. “Manufacturing process monitoring using time-frequency representation and transfer learning of deep neural networks”. In: *Journal of Manufacturing Processes* 68 (2021), pp. 231–248.

[Lind19] B. Lindemann, F. Fesenmayr, N. Jazdi, and M. Weyrich. “Anomaly detection in discrete manufacturing using self-learning approaches”. In: *Procedia CIRP* 79 (2019), pp. 313–318.

[Lind20] B. Lindemann, N. Jazdi, and M. Weyrich. “Anomaly detection and prediction in discrete manufacturing based on cooperative LSTM networks”. In: *International Conference on Automation Science and Engineering* (2020), pp. 1003–1010.

[Lipt16] Z. C. Lipton. “The Mythos of Model Interpretability”. In: *International Conference on Machine Learning, Workshop on Human Interpretability in Machine Learning* (2016).

[Liu08] F. T. Liu, K. M. Ting, and Z.-H. Zhou. “Isolation Forest”. In: *IEEE International Conference on Data Mining* (2008), pp. 413–422.

[Liu16] W. Liu, Y. Wen, Z. Yu, and M. Yang. “Large-Margin Softmax Loss for Convolutional Neural Networks”. In: *International Conference on Machine Learning* (2016).

[Liu18] J. Liu, J. Guo, P. Orlik, M. Shibata, D. Nakahara, S. Mii, and M. Takac. “Anomaly Detection in Manufacturing Systems Using Structured Neural Networks”. In: *Congress on Intelligent Control and Automation* (2018), pp. 175–180.

[Liu20] Y. Liu, Y. Gao, and W. Yin. “An Improved Analysis of Stochastic Gradient Descent with Momentum”. In: *Conference on Neural Information Processing Systems* (2020).

[Liu21] B. Liu, D. Wang, K. Lin, P.-N. Tan, and J. Zhou. “RCA: A Deep Collaborative Autoencoder Approach for Anomaly Detection”. In: *International Joint Conference on Artificial Intelligence* 2 (2021), pp. 1505–1511.

[Lizn21] P. Liznerski, L. Ruff, R. Vandermeulen A., B. Joe Franks, M. Kloft, and K.-R. Müller. “Explainable Deep One-Class Classification”. In: *International Conference on Learning Representations* (2021).

[Losh17] I. Loshchilov and F. Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: *International Conference on Learning Representations* (2017).

[Losh19] I. Loshchilov and F. Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations* (2019).

[Lu23] S. Lu, H. Dong, and H. Yu. “Abnormal Condition Detection Method of Industrial Processes Based on Cascaded Bagging-PCA and CNN Classification Network”. In: *IEEE Transactions on Industrial Informatics* (2023), pp. 1–11.

[Ma19] J. Ma and D. Yarats. “Quasi-Hyperbolic Momentum and Adam for Deep Learning”. In: *International Conference on Learning Representations* (2019).

[MacG94] J. F. MacGregor, C. Jaeckle, C. Kiparissides, and M. Koutoudi. “Process monitoring and diagnosis by multiblock PLS methods”. In: *AIChE Journal* 40.5 (1994), pp. 826–838.

[MacG95] J. F. MacGregor and T. Kourti. “Statistical Process Control of Multivariate Processes”. In: *Control Engineering Practice* 3.3 (1995), pp. 402–414.

[MacG97] J. F. MacGregor. “Using On-Line Process Data to Improve Quality: Challenges for Statisticians”. In: *International Statistical Review* 65.3 (1997), pp. 309–323.

[Magg14] M. Maggioni, E. Marzorati, M. Grasso, B. M. Colosimo, and P. Parenti. “In-Process Quality Characterization of Grinding Processes: A Sensor-Fusion Based Approach”. In: *ASME Conference on Engineering System* (2014).

[Mahe16] A. Mahendran and A. Vedaldi. “Salient Deconvolutional Networks”. In: *European Conference on Computer Vision* (2016), pp. 120–135.

[Mall09] S. G. Mallat. *A wavelet tour of signal processing: The sparse way*. 3rd ed. Amsterdam and Boston: Elsevier/Academic Press, 2009.

[Maso02] R. L. Mason and J. C. Young. *Multivariate Statistical Process Control with Industrial Applications*. ASA-SIAM series on statistics and applied probability. Philadelphia PA and Alexandria VA: Society for Industrial and Applied Mathematics and ASA, 2002.

[McCu43] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *Bulletin of Mathematical Biophysics* (1943), pp. 115–133.

[Mitc97] T. M. Mitchell. *Machine Learning*. McGraw-Hill series in computer science. New York: McGraw-Hill, 1997.

[Mont09] D. C. Montgomery. *Introduction to Statistical Quality Control*. 6th ed. Hoboken NJ: Wiley, 2009.

[Murp12] K. P. Murphy. *Machine learning: A probabilistic perspective*. Adaptive computation and machine learning series. Cambridge MA: MIT Press, 2012.

[Nair10] V. Nair and G. E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *International Conference on Machine Learning* (2010).

[Nali19] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. “Do Deep Generative Models Know What They Don’t Know?” In: *International Conference on Learning Representations* (2019).

[Nest04] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Massachusetts, USA: Kluwer Academic Publishers, 2004.

[Nguy19] D. T. Nguyen, Z. Lou, M. Klar, and T. Brox. “Anomaly Detection With Multiple-Hypotheses Predictions”. In: *International Conference on Machine Learning* (2019), pp. 4800–4809.

[Niek23] A. Niekrawietz, C. Litzinger, D. Jeckel, H. Lee, N. Bode, and T. Schadt. *Evaluation of a collaborative AI to support decision-making by workforce members in quality control*. Student report in the course of the KompAKI research project, supervised by: N. Theobald, T. Steinebach, P. Joisten and T.Biegel. 2023.

[Niel15] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[Noor11] R. Noorassana, A. Saghaei, and A. Amiri. *Statistical Analysis of Profile Monitoring*. Wiley Series in Probability and Statistics. Hoboken NJ: Wiley, 2011.

[Noro16] M. Noroozi and P. Favaro. “Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles”. In: *European Conference on Computer Vision* (2016), pp. 69–84.

[Noro18] M. Noroozi, A. Vinjimoor, P. Favaro, and H. Pirsiavash. “Boosting Self-Supervised Learning via Knowledge Transfer”. In: *International Conference on Computer Vision and Pattern Recognition* (2018), pp. 9359–9367.

[Oakl03] J. S. Oakland. *Statistical Process Control*. 5th ed. Butterworth-Heinemann, 2003.

[Oliv18] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow. “Realistic Evaluation of Deep Semi-Supervised Learning Algorithms”. In: *Conference on Neural Information Processing Systems* 32 (2018).

[Oord18] A. v. d. Oord, Y. Li, and O. Vinyals. *Representation Learning with Contrastive Predictive Coding*. 2018. arXiv: 1807.03748v2.

[Open22a] OpenAI. *ChatGPT: Optimizing Language Models for Dialogue*. OpenAI. Nov. 2022. URL: https://openai.com/blog/chatgpt (visited on 08/28/2023).

[Open22b] OpenAI. *DALL.E 2*. OpenAI. Apr. 2022. URL: https://openai.com/dall-e-2 (visited on 08/28/2023).

[Oshi23] T. Oshida, T. Murakoshi, L. Zhou, H. Ojima, K. Kaneko, T. Onuki, and J. Shimizu. "Development and implementation of real-time anomaly detection on tool wear based on stacked LSTM encoder-decoder model". In: *International Journal of Advanced Manufacturing Technology* (2023).

[Pang21] G. Pang, C. Shen, L. Cao, and A. van den Hengel. "Deep Learning for Anomaly Detection: A review". In: *ACM Computing Surveys* 54.2 (2021), pp. 1–38.

[Park20] H. Park, J. Noh, and B. Ham. "Learning Memory-Guided Normality for Anomaly Detection". In: *Conference on Computer Vision and Pattern Recognition* (2020), pp. 14372–14381.

[Parz62] E. Parzen. "On estimation of a probability density function and mode". In: *The Annals of Mathematical Statistics* (1962), pp. 1065–1076.

[Path16] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. "Context Encoders: Feature Learning by Inpainting". In: *International Conference on Computer Vision and Pattern Recognition* (2016).

[Pedr11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Pidh18] S. Pidhorskyi, R. Almohsen, and G. Doretto. "Generative Probabilistic Novelty Detection with Adversarial Autoencoders". In: *Conference in Neural Information Processing Systems* (2018).

[Piri21] S. Piri, A.-S. G. Abdel-Salam, and E. L. Boone. "A wavelet approach for profile monitoring of Poisson distribution with application". In: *Communications in Statistics - Simulation and Computation* 50.2 (2021), pp. 525–536.

[Pogg85] T. Poggio, V. Torre, and C. Koch. "Computational vision and regularization theory". In: *Nature* 317.6035 (1985), pp. 314–319.

[Prot20] A. Proteau, R. Zemouri, A. Tahan, and M. Thomas. "Dimension reduction and 2D-visualization for early change of state detection in a machining process with a variational autoencoder approach". In: *International Journal of Advanced Manufacturing Technology* 111.11-12 (2020), pp. 3597–3611.

[Qiu21] C. Qiu, T. Pfrommer, M. Kloft, S. Mandt, and M. Rudolph. "Neural Transformation Learning for Deep Anomaly Detection Beyond Images". In: *International Conference on Machine Learning* (2021), pp. 8703–8714.

[Ranz07] M. A. Ranzato, F.-J. Huang, Y.-L. Boureau, and Y. Lecun. "Unsupervised learning of Invariant Feature Hierarchies with Applications to Object Recognition". In: *Conference on Computer Vision and Pattern Recognition* (2007).

[Rao17] A. S. Rao and G. Verweij. *Sizing the price - What's the real value of AI for your business and how can you capitalise?, PwC*. Technical report. 2017.

[Rebj21] Q. Rebjock, B. Kurt, T. Januschowski, and L. Callot. “Online false discovery rate control for anomaly detection in time series”. In: *Conference on Neural Information Processing Systems* (2021).

[Ren19] J. Ren, P. J. Liu, E. Fertig, J. Snoek, R. Poplin, M. Depristo, J. Dillon, and B. Lakshminarayanan. “Likelihood Ratios for Out-of-Distribution Detection”. In: *Conference on Neural Information Processing Systems* 32 (2019).

[Riou91] O. Rioul and M. Vetterli. “Wavelets and signal processing”. In: *IEEE Signal Processing Magazine* 8.4 (1991), pp. 14–38.

[Riou92] O. Rioul and P. Duhamel. “Fast algorithms for discrete and continuous wavelet transforms”. In: *IEEE Transactions on information theory* 38.2 (1992), pp. 569–586.

[Rose56] M. Rosenblatt. “Remarks on some nonparametric estimates of a density function”. In: *The Annals of Mathematical Statistics* (1956), pp. 832–837.

[Rose58] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain”. In: *Psychological review* (1958).

[Roth22] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler. “Towards Total Recall in Industrial Anomaly Detection”. In: *Conference on Computer Vision and Pattern Recognition* (2022).

[Rude16] S. Ruder. *An overview of gradient descent optimization algorithms*. 2016. arXiv: 1609.04747v2.

[Ruff18] L. Ruff, R. A. Vandermeulen, N. Görnitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. “Deep One-Class Classification”. In: *International Conference on Machine Learning* (2018).

[Ruff21a] L. Ruff. “Deep One-Class Learning: A Deep Learning Approach to Anomaly Detection”. PhD Thesis. Technical University Berlin, 2021.

[Ruff21b] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. “A Unifying Review of Deep and Shallow Anomaly Detection”. In: *Proceedings of the IEEE* 109.5 (2021), pp. 756–795.

[Rume86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: (1986), pp. 533–536.

[Russ16] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Upper Saddle River NJ: Pearson Education Inc, 2016.

[Sale21] M. Salehi, N. Sadjadi, S. Baselizadeh, M. H. Rohban, and H. R. Rabiee. “Multiresolution Knowledge Distillation for Anomaly Detection”. In: *Conference on Computer Vision and Pattern Recognition* (2021), pp. 14902–14912.

[Samu16] R. T. Samuel and Y. Cao. “Nonlinear process fault detection and identification using kernel PCA and kernel density estimation”. In: *Systems Science & Control Engineering* 4.1 (2016), pp. 165–174.

[Sant18] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. “How Does Batch Normalization Help Optimization?” In: *Conference on Neural Information Processing Systems* (2018).

[Saur11] J. Sauro. *A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. Measuring Usability LLC, 2011.

[Schö97] B. Schölkopf, A. Smola, and K.-R. Müller. “Kernel Principal Component Analysis”. In: *International Conference on Artificial Neural Networks* (1997), pp. 583–588.

[Schö99] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. “Support Vector Method for Novelty Detection”. In: *Conference on Neural Information Processing Systems* (1999), pp. 582–588.

[Sehw21] V. Sehwag, M. Chiang, and P. Mittal. “SSD: A Unified Framework for Self-Supervised Outlier Detection”. In: *International Conference on Learning Representations* (2021).

[Selv19] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”. In: *International Journal of Computer Vision* 128.2 (2019), pp. 336–359.

[Serg21] N. D. Sergin and H. Yan. “Toward a better monitoring statistic for profile monitoring via variational autoencoders”. In: *Journal of Quality Technology* (2021), pp. 1–46.

[Serr05] T. Serre, L. Wolf, and T. Poggio. “Object Recognition with Features Inspired by Visual Cortex”. In: *International Conference on Computer Vision and Pattern Recognition* (2005), pp. 994–1000.

[Serr07] T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio. “A quantitative theory of immediate visual recognition”. In: *Progress in brain research* 165 (2007), pp. 33–56.

[Shal14] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. New York NY USA: Cambridge University Press, 2014.

[Shao13] C. Shao, K. Paynabar, T. H. Kim, J. Jin, S. J. Hu, J. P. Spicer, H. Wang, and J. A. Abell. “Feature selection for manufacturing process monitoring using cross-validation”. In: *Journal of Manufacturing Systems* 32.4 (2013), pp. 550–555.

[Shen20] L. Shen, Z. Li, and J. T. Kwok. “Timeseries Anomaly Detection using Temporal Hierarchichal One-Class Network”. In: *Conference on Neural Information Processing Systems* 33 (2020), pp. 13016–13026.

[Shen21] L. Shen, Z. Yu, Q. Ma, and J. T. Kwok. “Time Series Anomaly Detection with Multiresolution Ensemble Decoding”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.11 (2021), pp. 9567–9575.

[Shen22] T. Shenkar and L. Wolf. “Anomaly detection for tabular data with internal contrastive learning”. In: *International Conference on Learning Representations* (2022).

[Shew31] W. A. Shewhart. *Economic Control of Quality of Manufactured Product*. New York NY: D. Van Nostrand Company, 1931.

[Shi23] H. Shi, W. Mao, Y. Zhang, and X. Liang. “Unsupervised Deep Tensor Multi-task Anomaly Detection with Rule Adaptation for Online Early Fault Evaluation”. In: *IEEE Sensors Journal* (2023), p. 1.

[Shin20] H.-K. Shin, W. Lee, J.-H. Yun, and H.-C. Kim. “HAI 1.0: HIL-based Augmented ICS Security Dataset”. In: *13th USENIX Workshop on Cyber Security Experimentation and Test (CSET 20)* (2020).

[Sinh05] S. Sinha, P. S. Routh, P. D. Anno, and J. P. Castagna. “Spectral decomposition of seismic data with continuous-wavelet transform”. In: *Geophysics* 70.6 (2005), pp. 19–25.

[Smit18] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le. “Don’t Decay the Learning Rate, Increase the Batch Size”. In: *International Conference on Learning Representations* (2018).

[Sohn16] K. Sohn. “Improved Deep Metric Learning with Multi-class N-pair Loss Objective”. In: *Conference on Neural Information Processing Systems* (2016), pp. 1857–1865.

[Sohn21] K. Sohn, C.-L. Li, J. Yoon, M. Jin, and T. Pfister. “Learning and Evaluating Representations for Deep One-class Classification”. In: *International Conference on Learning Representations* (2021).

[Song07] X. Song, M. Wu, C. Jermaine, and S. Ranka. “Conditional Anomaly Detection”. In: *IEEE Transactions on Knowledge and Data Engineering* 19.5 (2007), pp. 631–645.

[Sord21] A. Sordoni, N. Dziri, H. Schulz, G. Gordong, P. Bachman, and R. Tachet. “Decomposed Mutual Information Estimation for Contrastive Representation Learning”. In: *International Conference on Machine Learning* (2021), pp. 9859–9869.

[Spie00] S. Spiewak, R. Duggirala, and K. Barnett. “Predictive monitoring and control of the cold extrusion process”. In: *CIRP Annals* (2000), pp. 383–386.

[Stou00] Z. G. Stoumbos, M. R. Reynolds, T. P. Ryan, and W. H. Woodall. “The State of Statistical Process Control as We Proceed into the 21st Century”. In: *Journal of the American Statistical Association* (2000), pp. 992–998.

[Stra16] G. Strang. *Introduction to linear algebra*. Fifth edition. Wellesley, MA: Wellesley-Cambridge Press, 2016.

[Stra19] G. Strang. *Linear Algebra and Learning from Data*. Wellesley-Cambridge Press, 2019.

[Sun03] R. Sun and F. Tsung. “A kernel-distance-based multivariate control chart using support vector methods”. In: *International Journal of Production Research* 41.13 (2003), pp. 2975–2989.

[Sun23] S. Sun, Y. Liu, X. Hu, and W. Zhang. “A semisupervised autoencoder-based method for anomaly detection in cutting tools”. In: *Journal of Manufacturing Processes* 93 (2023), pp. 315–327.

[Sutt18] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[Tack20] J. Tack, S. Mo, J. Jeong, and J. Shin. “CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances”. In: *Conference on Neural Information Processing Systems* 33 (2020), pp. 11839–11852.

[Take06] I. Takeuchi, Q. V. Le, T. D. Sears, and A. J. Smola. “Nonparametric Quantile Estimation”. In: *Journal of Machine Learning Research* (2006), pp. 1231–1264.

[Tang20] P. Tang, K. Peng, J. Dong, K. Zhang, and S. Zhao. “Monitoring of Nonlinear Processes With Multiple Operating Modes Through a Novel Gaussian Mixture Variational Autoencoder Model”. In: *IEEE Access* 8 (2020), pp. 114487–114500.

[Tax01] D. M. Tax. “One-class classification: Concept-learning in the absence of counter-examples”. PhD Thesis. Delft University of Technology, 2001.

[Tax04] D. M. Tax and R. P. Duin. “Support Vector Data Description”. In: *Machine Learning* 54.1 (2004), pp. 45–66.

[Tax99] D. M. Tax and R. P. Duin. “Support vector domain description”. In: *Pattern Recognition Letters* (1999), pp. 1191–1199.

[Tian20] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. “What Makes for Good Views for Contrastive Learning?” In: *Conference on Neural Information Processing Systems* (2020).

[Tiel12] T. Tieleman and G. Hinton. *Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning*. Technical report. 2012.

[Tnan22] M.-A. Tnani, M. Feil, and K. Diepold. “Smart Data Collection System for Brownfield CNC Milling Machines: A New Benchmark Dataset for Data-Driven Machine Monitoring”. In: *Procedia CIRP* 107 (2022), pp. 131–136.

[Tong17] C. Tong and X. Yan. “A Novel Decentralized Process Monitoring Scheme Using a Modified Multiblock PCA Algorithm”. In: *IEEE Transactions on Automation Science and Engineering* 14.2 (2017), pp. 1129–1138.

[Touv23] H. Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: `2307.09288`.

[Tran20a] M.-Q. Tran, M.-K. Liu, and Q.-V. Tran. “Milling chatter detection using scalogram and deep convolutional neural network”. In: *The International Journal of Advanced Manufacturing Technology* 107.3-4 (2020), pp. 1505–1516.

[Tran20b] T. Tran and J. Lundgren. “Drill Fault Diagnosis Based on the Scalogram and Mel Spectrogram of Sound Signals Using Artificial Intelligence”. In: *IEEE Access* 8 (2020).

[Vapn95] V. Vapnik. *The Nature Of Statistical Learning Theory*. 1st ed. New York: Springer, 1995.

[Wang18] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu. “Deep learning for smart manufacturing: Methods and applications”. In: *Journal of Manufacturing Systems* 48 (2018). Special Issue on Smart Manufacturing, pp. 144–156.

[Wang19] S. Wang, Y. Zeng, E. Zhu, J. Yin, C. Xu, and M. Kloft. “Effective End-to-end Unsupervised Outlier Detection via Inlier Priority of Discriminative Network”. In: *Conference on Neural Information Processing Systems* (2019).

[Wang21] X. Wang, S. Zhang, Z. Qing, Y. Shao, C. Gao, and N. Sang. “Self-Supervised Learning for Semi-Supervised Temporal Action Proposal”. In: *International Conference on Computer Vision and Pattern Recognition* (2021), pp. 1905–1914.

[Wang23] R. Wang, C. Liu, X. Mou, K. Gao, X. Guo, P. Liu, T. Wo, and X. Liu. “Deep Contrastive One-Class Time Series Anomaly Detection”. In: *SIAM International Conference on Data Mining* (2023), pp. 694–702.

[Wees16] M. Weese, W. Martinez, F. M. Megahed, and L. A. Jones-Farmer. “Statistical Learning Methods Applied to Process Monitoring: An Overview and Perspective”. In: *Journal of Quality Technology* 48.1 (2016), pp. 4–24.

[Wei16] Q. Wei, W. Huang, W. Jiang, and W. Zhao. “Real-time process monitoring using kernel distances”. In: *International Journal of Production Research* 54.21 (2016), pp. 6563–6578.

[Wei21] F. Wei, Y. Gao, Z. Wu, H. Hu, and S. Lin. “Aligning Pretraining for Detection via Object-Level Contrastive Learning”. In: *Conference on Neural Information Processing Systems* 34 (2021).

[Wies11] S. Wiesler, R. Schlüter, and H. Ney. “A Convergence Analysis of Log-Linear Training and its Application to Speech Recognition”. In: *IEEE Workshop on Automatic Speech Recognition and Understanding* (2011), pp. 1–6.

[Wolp97] D. H. Wolpert and W. G. Macready. “No Free Lunch Theorems For Optimization”. In: *IEEE Transactions on Evolutionary Computation* (1997), pp. 67–82.

[Wood00] W. H. Woodall. “Controversies and Contradictions in Statistical Process Control”. In: *Journal of Quality Technology* 32.4 (2000), pp. 341–350.

[Wood04] W. H. Woodall, D. J. Spitzner, D. C. Montgomery, and S. Gupta. “Using Control Charts to Monitor Process and Product Quality Profiles”. In: *Journal of Quality Technology* 36.3 (2004), pp. 309–320.

[Wood14] W. H. Woodall and D. C. Montgomery. “Some Current Directions in the Theory and Application of Statistical Process Monitoring”. In: *Journal of Quality Technology* 46.1 (2014), pp. 78–94.

[Wood17] W. H. Woodall. “Bridging the Gap between Theory and Practice in Basic Statistical Process Monitoring”. In: *Quality Engineering* 29.1 (2017), pp. 2–15.

[Wood99] W. H. Woodall and D. C. Montgomery. “Research Issues and Ideas in Statistical Process Control”. In: *Journal of Quality Technology* 31.4 (1999), pp. 376–386.

[Wu18] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. “Unsupervised Feature Learning via Non-Parametric Instance Discrimination”. In: *International Conference on Computer Vision and Pattern Recognition* (2018).

[Wu21] J.-C. Wu, D.-J. Chen, C.-S. Fuh, and T.-L. Liu. “Learning Unsupervised Metaformer for Anomaly Detection”. In: *International Conference on Computer Vision* (2021), pp. 4369–4378.

[Xie22] X. Xie and Q. Peihua. “Machine Learning Control Charts for Monitoring Serially Correlated Data”. In: *Control Charts and Machine Learning for Anomaly Detection in Manufacturing*. Springer Cham, 2022.

[Xu15] B. Xu, N. Wang, T. Chen, and M. Li. *Empirical Evaluation of Rectified Activations in Convolution Network*. 2015. arXiv: `1505.00853v2`.

[Xu22] J. Xu, H. Wu, J. Wang, and M. Long. “Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy”. In: *International Conference on Learning Representations* (2022).

[Yan20] S. Yan and X. Yan. “Quality-Driven Autoencoder for Nonlinear Quality-Related and Process-Related Fault Detection Based on Least-Squares Regularization and Enhanced Statistics”. In: *Industrial & Engineering Chemistry Research* 59.26 (2020), pp. 12136–12143.

[Yang21] C. Yang, Z. Wu, B. Zhou, and S. Lin. “Instance Localization for Self-Supervised Detection Pretraining”. In: *International Conference on Computer Vision and Pattern Recognition* (2021), pp. 3987–3996.

[Ye21] Z. Ye, Y. Chen, and H. Zheng. “Understanding the Effect of Bias in Deep Anomaly Detection”. In: *International Joint Conference on Artificial Intelligence* 3 (2021), pp. 3314–3320.

[Yu11] J. Yu. “Fault Detection Using Principal Components-Based Gaussian Mixture Model for Semiconductor Manufacturing Processes”. In: *IEEE Transactions on Semiconductor Manufacturing* 24.3 (2011), pp. 432–444.

[Yu18] J. Yu and X. Yan. “Layer-by-Layer Enhancement Strategy of Favorable Features of the Deep Belief Network for Industrial Process Monitoring”. In: *Industrial & Engineering Chemistry Research* 57.45 (2018), pp. 15479–15490.

[Yu20] J. Yu and C. Zhang. “Manifold regularized stacked autoencoders-based feature learning for fault detection in industrial processes”. In: *Journal of Process Control* 92 (2020), pp. 119–136.

[Yu21] J. Yu, X. Liu, and L. Ye. “Convolutional Long Short-Term Memory Autoencoder-Based Feature Learning for Fault Detection in Industrial Processes”. In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–15.

[Yu23] J. Yu and Y. Zhang. “Challenges and opportunities of deep learning-based process fault detection and diagnosis: a review”. In: *Neural Computing and Applications* 35.1 (2023), pp. 211–252.

[Yuan20] X. Yuan, D. Wang, Y. Wang, and W. Shao. “A Time Window based Two-Dimensional PCA for Process Monitoring and Its Application to Tennessee Eastman Process”. In: *IEEE Data Driven Control and Learning Systems Conference* (2020), pp. 1364–1369.

[Yue21] X. Yue, Z. Zheng, S. Zhang, Y. Gao, T. Darrell, K. Keutzer, and A. S. Vincentelli. “Prototypical Cross-Domain Self-Supervised Learning for Few-Shot Unsupervised Domain Adaptation”. In: *International Conference on Computer Vision and Pattern Recognition* (2021), pp. 13834–13844.

[Zbon21] J. Zbontar, L. Jing, I. Misra, Y. Lecun, and S. Deny. “Barlow Twins: Self-Supervised Learning via Redundancy Reduction”. In: *International Conference on Machine Learning* (2021), pp. 12310–12320.

[Zeil13] M. D. Zeiler and R. Fergus. “Stochastic Pooling for Regularization of Deep Convolutional Neural Networks”. In: *International Conference on Learning Representations* (2013).

[Zeil14] M. D. Zeiler and R. Fergus. “Visualizing and Understanding Convolutional Networks”. In: *European Conference on Computer Vision* (2014), pp. 818–833.

[Zhan16] R. Zhang, P. Isola, and A. A. Efros. “Colorful Image Colorization”. In: *European Conference on Computer Vision* (2016), pp. 649–666.

[Zhan19a] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, C. Wei, J. Ni, B. Zong, H. Chen, and N. V. Chawla. “A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data”. In: *AAAI Conference on Artifical Intelligence* (2019), pp. 1409–1416.

[Zhan19b] G. Zhang, C. Wang, B. Xu, and R. Grosse. “Three Mechanisms of Weight Decay Regularization”. In: *International Conference on Learning Representations* (2019).

[Zhan20] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into Deep Learning*. 2020.

[Zhan22] D. Zhang et al. *The AI Index 2022 Annual Report*. Technical report. AI Index Steering Committee, Stanford Institute for Human-Centered AI, Stanford University, Mar. 2022.

[Zhen18] A. Zheng and A. Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. Sebastopol CA: O'Reilly Media Inc, 2018.

[Zhen21] M. Zheng, S. You, F. Wang, C. Qian, C. Zhang, X. Wang, and C. Xu. "ReSSL: Relational Self-Supervised Learning with Weak Augmentation". In: *Conference on Neural Information Processing Systems* 34 (2021).

[Zhou16] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. "Learning Deep Features for Discriminative Localization". In: *Conference on Computer Vision and Pattern Recognition* (2016), pp. 2921–2929.

[Zhou18] Z.-H. Zhou. "A brief introduction to weakly supervised learning". In: *National Science Review* 5.1 (2018), pp. 44–53.

[Zong18] B. Zong, Q. Song, M. Renqiang Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. "Deep Autoencoding Gaussian Mixture Model For Unsupervised Anomaly Detection". In: *International Conference on Learning Representations* (2018).

[Zwet21] I. M. Zwetsloot and W. H. Woodall. "A Review of Some Sampling and Aggregation Strategies for Basic Statistical Process Monitoring". In: *Journal of Quality Technology* 53.1 (2021), pp. 1–16.

Norms

[DIN3534-2] DIN ISO 3534-2:2013-12. *Statistics - Vocabulary and symbols - Part 2: Applied statistics*

[DIN9000] DIN EN ISO 9000:2015:2015-11. *Quality management systems – Fundamentals and vocabulary*

[ISO7870-1] ISO 7870-1:2019-11. *Control charts - Part 1: General guidelines*

[ISO7870-2] ISO 7870-2:2013-4. *Control charts - Part 2: Shewhart control charts*

[ISO7870-4] ISO 7870-4:2021-9. *Control charts - Part 4: Cumulative sum charts*

[ISO7870-6] ISO 7870-6:2016-2. *Control charts - Part 6: EWMA control charts*

[ISO7870-7] ISO 7870-7:2020-2. *Control charts - Part 7: Multivariate control charts*

[ISO9241-11] ISO 9241-11:2018-3. *Ergonomics of human-system interaction - Part 11: Usability: Definitions and concepts*

A. Supplementary material for Chapter 2

A.1. Machine learning

A.1.1. Data preprocessing - example

To make the necessity of feature scaling more apparent, consider Table A.1, in which a small toy dataset is depicted with ten samples and two features with two different scales each.

Table A.1.: Toy dataset to demonstrate feature scaling, own illustration adapted from [Jame13].

No.	Salary[$\frac{€}{\text{hour}}$]	Salary[$\frac{€}{\text{day}}$]	Age[year]	Age[month]	Class
1	10	80	20	240	b
2	20	160	25	300	b
3	30	240	26	312	a
4	5	40	30	360	b
5	50	200	40	480	a
6	40	160	50	600	a
7	28	224	32	384	b
8	14	112	16	192	b
9	33	264	29	348	a
10	38	304	59	708	a

The first feature x_1 represents the income of a person and is displayed both on an hourly and a daily basis. Conversely, the second feature x_2 corresponds to the age of a person and is given both in years and months. Each person belongs to one of two job classes: a and b.

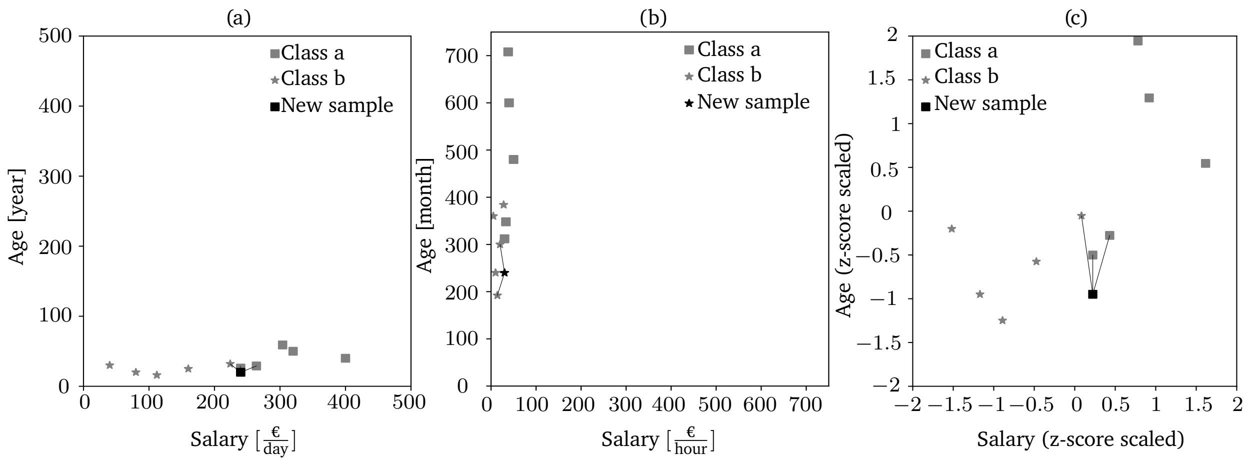


Figure A.1.: Impact of feature scaling in ML, own illustration adapted from [Jame13].

Now consider, for instance, the k-Nearest Neighbor (KNN) algorithm [Fix51; Cove67] that assigns a new sample to the most frequently occurring class among its k closest neighbors, and assume that $k = 3$. In this example, the closeness between two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^2$ is measured using the Euclidean distance, i.e., $d(\mathbf{x}_1, \mathbf{x}_2) = ||\mathbf{x}_1 - \mathbf{x}_2||_2$.
Figure A.1 (a) and (b) demonstrate that the model yields different classification results depending on the scale of the features. However, when the data are scaled with, e.g., z-score scaling as shown in Figure A.1 (c), the scale dependency can be removed.
It is important to emphasize that the respective feature scaling parameters are computed prior to training based on the training set and are subsequently used to transform both the training set and new data points that are not part of the training set such as the validation and test sets [Bish96; Chol17; Gero19].

A.1.2. Gradient-based optimization - ADAM

ADAM combines the advantages of two other SGD-based optimization algorithms, namely Root Mean Square Propagation (RMSProp) [Tiel12] and Adaptive Gradient (AdaGrad) [Duch11]. Specifically, the idea is to continuously update exponential moving averages of the first ($\mathbf{m}_j$) and second ($\mathbf{v}_j$) moment estimates of the gradient using the hyperparameters $\beta_1, \beta_2 \in [0; 1)$ that control the exponential decay rate [King15]. The moment estimates $\mathbf{m}_j$ and $\mathbf{v}_j$ are typically initialized as $\mathbf{m}_j = \mathbf{v}_j = \mathbf{0}$, whereas the default settings for β_1, β_2 are $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [Zhan20]. Algorithm 5 provides the pseudocode for the ADAM algorithm.

Algorithm 5 ADAM algorithm for gradient-based optimization. All operations on vectors are element-wise, own illustraton adapted from [King15].

```
Require: $\gamma$: learning rate
Require: $\epsilon$: small constant to stabilize division by small numbers
Require: $\beta_1, \beta_2 \in [0; 1)$: exponential decay rates for the moment estimates
Require: $\mathcal{L}(\phi(\cdot, \mathbf{w}))$: loss function with parameters $\mathbf{w}$
Require: $\mathbf{w}_0$: initial parameter vector
  $\mathbf{m}_0 \leftarrow \mathbf{0}$                                  ▷ initialize 1st moment vector
  $\mathbf{v}_0 \leftarrow \mathbf{0}$                                  ▷ initialize 2nd moment vector
  $j \leftarrow 0$                                  ▷ initialize time step
  while $\mathbf{w}_j$ not converged do
      $j \leftarrow j + 1$
      $\mathbb{B} \leftarrow \{\mathbf{x}_1, \cdots, \mathbf{x}_{|\mathbb{B}|}\} \subset \mathbb{X}_{\text{train}}$      ▷ sample random minibatch of size $|\mathbb{B}|$ including labels $\mathbf{y}_i$
      $\mathbf{g}_j \leftarrow \frac{1}{|\mathbb{B}|} \sum_{i=1}^{|\mathbb{B}|} \nabla_{\mathbf{w}} \mathcal{L}(\phi(\mathbf{x}_i; \mathbf{w}_j), \mathbf{y}_i)$      ▷ compute gradients with respect to weights $\mathbf{w}_j$
      $\mathbf{m}_j \leftarrow \beta_1 m_{j-1} + (1 - \beta_1)\mathbf{g}_j$      ▷ update biased first moment estimate
      $\mathbf{v}_j \leftarrow \beta_2 v_{j-1} + (1 - \beta_2)\mathbf{g}_j^2$      ▷ update biased second raw moment estimate
      $\hat{\mathbf{m}}_j \leftarrow \mathbf{m}_j / (1 - \beta_1^j)$      ▷ compute bias-corrected first moment estimate
      $\hat{\mathbf{v}}_j \leftarrow \mathbf{v}_j / (1 - \beta_2^j)$      ▷ compute bias-corrected second raw moment estimate
      $\mathbf{w}_j \leftarrow \mathbf{w}_{j-1} - \gamma \hat{\mathbf{m}}_j / (\sqrt{\hat{\mathbf{v}}_j} + \epsilon)$      ▷ update parameters
  end while
  return $\mathbf{w}_j$
```

A.1.3. Backpropagation - example

To make the idea of backpropagation more tangible, consider a simple MLP with one input unit, a single hidden layer with one hidden unit, and an output layer with one output unit. Suppose that the activation function in the hidden layer ρ_1 is a ReLU, the activation of the output layer ρ_2 is a sigmoid, and the loss function is binary CE loss, i.e., $\mathcal{L}(\hat{y}, y) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$. Assume further a training input $x = 1$ and a corresponding label $y = 1$. Both weights of the presented MLP are randomly initialized as $w^{(1)} = w^{(2)} = 0.1$. Bias terms are omitted for simplicity reasons.
Figure A.2 provides a visualization of both the forward pass and the backward pass for the considered MLP.

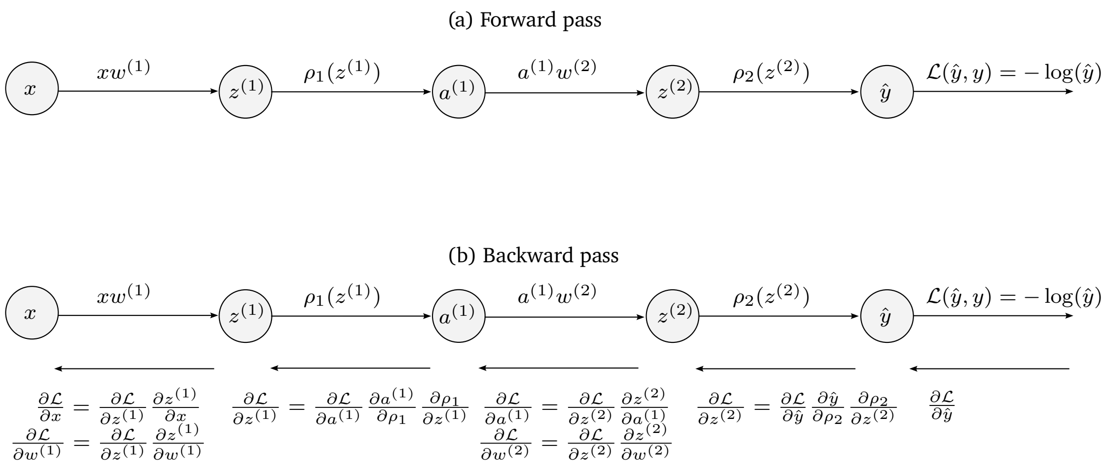

(a) Forward pass
x
$xw^{(1)}$
$z^{(1)}$
$\rho_1(z^{(1)})$
$a^{(1)}$
$a^{(1)}w^{(2)}$
$z^{(2)}$
$\rho_2(z^{(2)})$
$\hat{y}$
$\mathcal{L}(\hat{y}, y) = -\log(\hat{y})$
(b) Backward pass
x
$xw^{(1)}$
$z^{(1)}$
$\rho_1(z^{(1)})$
$a^{(1)}$
$a^{(1)}w^{(2)}$
$z^{(2)}$
$\rho_2(z^{(2)})$
$\hat{y}$
$\mathcal{L}(\hat{y}, y) = -\log(\hat{y})$
$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial x}$
$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial w^{(1)}}$
$\frac{\partial \mathcal{L}}{\partial z^{(1)}} = \frac{\partial \mathcal{L}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial \rho_1} \frac{\partial \rho_1}{\partial z^{(1)}}$
$\frac{\partial \mathcal{L}}{\partial a^{(1)}} = \frac{\partial \mathcal{L}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}}$
$\frac{\partial \mathcal{L}}{\partial w^{(2)}} = \frac{\partial \mathcal{L}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial w^{(2)}}$
$\frac{\partial \mathcal{L}}{\partial z^{(2)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \rho_2} \frac{\partial \rho_2}{\partial z^{(2)}}$
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$

Figure A.2.: Backpropagation. (a) Forward pass: The output of the network and the corresponding loss are computed by following the computation chain from input layer to output layer. (b) Backward pass: The gradients are computed via the chain rule starting from the output layer back to the input layer, own illustration adapted from [Good16; Deis20].

With the described settings of the MLP, the forward pass can be computed as:

$$
\begin{aligned}
\mathcal{L}(y, \hat{y}) &= -y\log(\hat{y}) - (1-y)\log(1-\hat{y}) && \text{(A.1)}\\
&= -\log(\rho_2(w^{(2)}(\rho_1(w^{(1)}x)))) && \text{(A.2)}\\
&= -\log\left(\frac{1}{1+e^{-w^{(2)}\max(w^{(1)}x,0)}}\right) && \text{(A.3)}\\
&= -\log\left(\frac{1}{1+e^{-0.1\max(0.1,0)}}\right) && \text{(A.4)}\\
&= -\log(0.5) && \text{(A.5)}\\
&= 0.69 && \text{(A.6)}
\end{aligned}
$$

The backward pass then uses the precomputed values from the forward pass to obtain the gradients with respect to $w^{(1)}$ and $w^{(2)}$.

$$\frac{\partial \mathcal{L}}{\partial w^{(2)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \rho_2} \frac{\partial \rho_2}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial w^{(2)}} \tag{A.7}$$

$$= -\frac{1}{\hat{y}} \cdot \frac{1}{1+e^{-z^{(2)}}} \cdot \left(1 - \frac{1}{1+e^{-z^{(2)}}}\right) \cdot a^{(1)} \tag{A.8}$$

$$= -2 \cdot 0.5 \cdot (1 - 0.5) \cdot 0.1 \tag{A.9}$$

$$= -0.05 \tag{A.10}$$

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \rho_2} \frac{\partial \rho_2}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial \rho_1} \frac{\partial \rho_1}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial w^{(1)}} \tag{A.11}$$

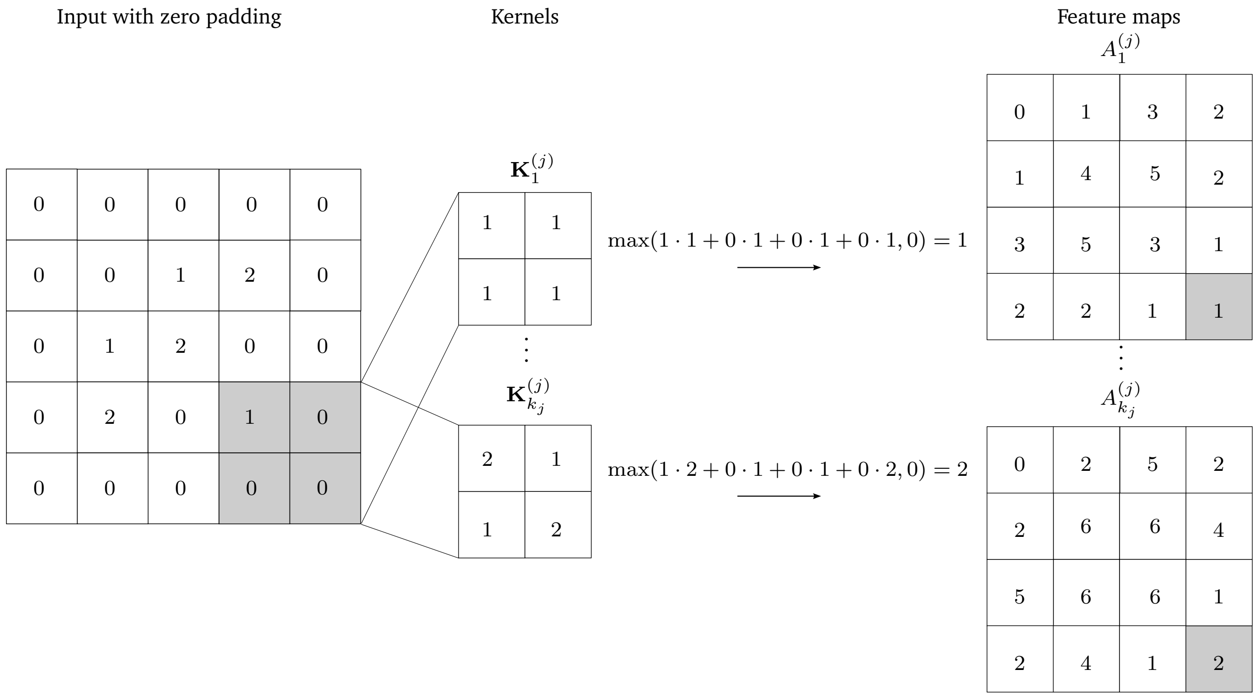
$$= -\frac{1}{\hat{y}} \cdot \frac{1}{1+e^{-z^{(2)}}} \cdot \left(1 - \frac{1}{1+e^{-z^{(2)}}}\right) \cdot a^{(1)} \cdot z^{(1)} \cdot x \tag{A.12}$$

$$= -2 \cdot 0.5 \cdot (1 - 0.5) \cdot 0.1 \cdot 1 \cdot 1 \cdot 1 \tag{A.13}$$

$$= -0.05 \tag{A.14}$$

A.1.4. Zero padding

Figure A.3 visualizes the effects of padding on the output size. By adding extra zero units to the input perimeter, the output size of the feature maps increases from 2×2 to 4×4.


Input with zero padding
Kernels
Feature maps
$A_1^{(j)}$
$\mathbf{K}_1^{(j)}$
$\max(1 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 1, 0) = 1$
$\mathbf{K}_{k_j}^{(j)}$
$A_{k_j}^{(j)}$
$\max(1 \cdot 2 + 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 2, 0) = 2$

0	0	0	0	0
0	0	1	2	0
0	1	2	0	0
0	2	0	1	0
0	0	0	0	0

$\mathbf{K}_1^{(j)}$

1	1
1	1

$\mathbf{K}_{k_j}^{(j)}$

2	1
1	2

$A_1^{(j)}$

0	1	3	2
1	4	5	2
3	5	3	1
2	2	1	1

$A_{k_j}^{(j)}$

0	2	5	2
2	6	6	4
5	6	6	1
2	4	1	2

Figure A.3.: Zero padding of the input to prevent information loss. Note that the respective bias terms $w_{0,1}^{(j)}, \cdots, w_{0,k_j}^{(j)}$ are assumed to be zero, own illustration adapted from [Zhan20].

A.1.5. Max-pooling

Figure A.4 provides a visualization of the max-pooling operation.

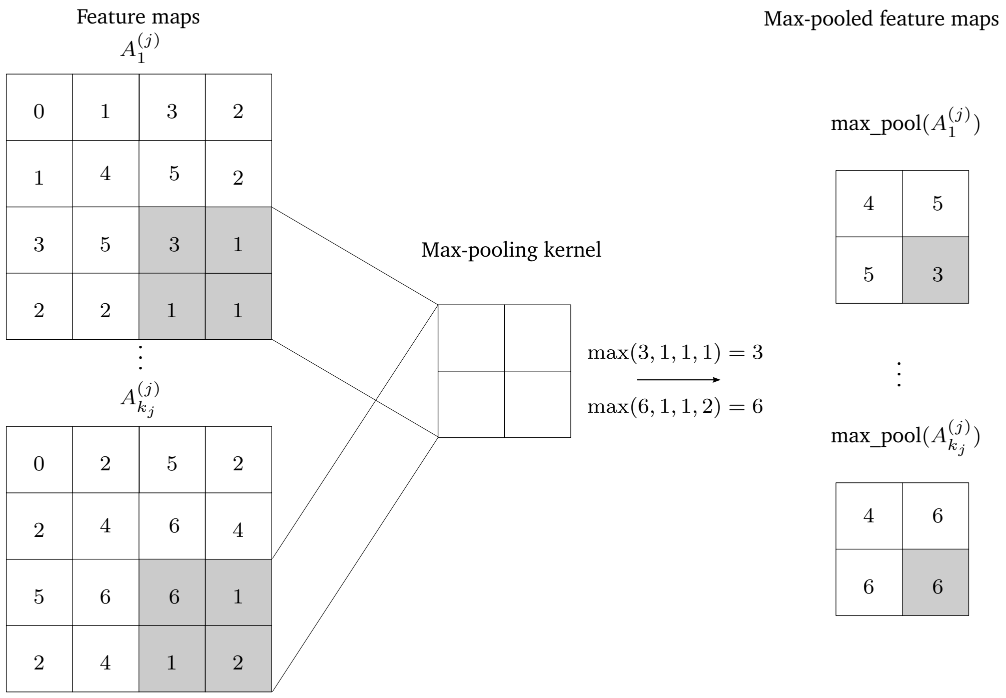


Figure A.4.: Max-pooling operation in a pooling layer of a CNN. The value of a specific unit in the pooled feature map is obtained by selecting the maximum unit inside the respective local receptive field. Horizontal and vertical stride $s_h = s_v = 2$, own illustration adapted from [Zhan20].

B. Supplementary material for Chapter 3

B.1. Systematic literature review I

Table B.1.: Details for systematic literature review I, own illustration.

Category	Details
Search string	(("self-supervised" OR "deep" OR "contrastive") AND ("anomaly detection" OR "outlier detection" OR "novelty detection" OR "out-of-distribution" OR "one-class" OR "classification"))
Databases	Web of science, IEEE Xplore, JSTOR, SCOPUS, ScienceDirect
Eligibility criteria	Articles published in Q1/Q2 journals Articles published in leading ML conferences, e.g., ICLR, ICML, AAAI, NIPS Articles published between 2018 and 2023 Articles addressing fundamental ML research
Screening process	Title screening Abstract screening Full text review

B.2. Systematic literature review II

Table B.2.: Details for systematic literature review II, own illustration.

Category	Details
Search string	(("manufacturing" OR "machining") AND ("quality assurance" OR "quality control" OR "quality monitoring" OR "process control" OR "process monitoring" OR "anomaly detection" OR "fault detection" OR "sensor signal monitoring" OR "profile monitoring")) AND ("deep learning" OR "neural network*" OR "autoencoder")
Databases	Web of science, IEEE Xplore, JSTOR, SCOPUS, ScienceDirect
Eligibility criteria	Articles published in Q1/Q2 journals Articles published in leading manufacturing conferences, e.g., CMS, CPSL, CLF, ICME Articles published between 2018 and 2023 In-process monitoring approaches focusing on continuous or discrete manufacturing processes Semi-supervised AD setting in which multivariate process observations are processed
Screening process	Title screening Abstract screening Full text review

C. Supplementary material for Chapter 6

C.1. Validation I

C.1.1. Bosch CNC milling dataset - hyperparameters for state-of-the-art baselines

Table C.1.: Hyperparameters for shallow AD baselines, own illustration adapted from [Bieg23].

	Hyperparameter	$\tilde{\mathbb{D}}_{\text{Bosch}}$	$\mathbb{D}_{S,\text{Bosch}}$
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
KPCA	n_components	15	13
	kernel	cosine	cosine

Table C.2.: Hyperparameters for deep AD baselines, own illustration adapted from [Bieg23].

	Hyperparameter	$\tilde{\mathbb{D}}_{\text{Bosch}}$	$\mathbb{D}_{S,\text{Bosch}}$
DAGMM	batch_size	-	8
	learning_rate	-	10^{-4}
	epochs	-	1000
Deep-SVDD	batch_size	8	-
	learning_rate	10^{-3}	-
	epochs	50	-
	weight_decay	10^{-6}	-
	ν	0.005	-
PatchCore	batch_size	2	-
	patch_size	3	-
LeNet-type AE	batch_size	8	-
	learning_rate	10^{-4}	-
	epochs	100	-

Table C.3.: Hyperparameters for self-supervised AD baselines, own illustration adapted from [Bieg23].

	Hyperparameter	$\tilde{\mathbb{D}}_{\text{Bosch}}$
RotNet	batch_size	8
	learning_rate	10^{-1}
	epochs	100
	weight_decay	$5 \cdot 10^{-4}$
	input_size	128×128
RotNet+PCA	n_components	5
RotNet+KPCA	n_components	3
	kernel	cosine
RotNet+OC-SVM	kernel	RBF
	γ	0.5
	ν	0.005
RotNet+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

C.1.2. CiP-DMD - hyperparameters for state-of-the-art baselines

Table C.4.: Hyperparameters for shallow AD baselines, own illustration adapted from [Bieg23].

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

Table C.5.: Hyperparameters for deep AD baselines, own illustration adapted from [Bieg23].

	Hyperparameter	$\tilde{\mathbb{D}}_{\text{P1}}$	$\mathbb{D}_{S,\text{P1}}$	$\tilde{\mathbb{D}}_{\text{P2}}$	$\mathbb{D}_{S,\text{P2}}$
DAGMM	batch_size	-	8	-	8
	learning_rate	-	10^{-4}	-	10^{-4}
	epochs	-	1000	-	1000
Deep-SVDD	batch_size	8	-	8	-
	learning_rate	10^{-3}	-	10^{-3}	-
	epochs	50	-	50	-
	weight_decay	10^{-6}	-	10^{-6}	-
	ν	0.005	-	0.005	-
PatchCore	batch_size	2	-	2	-
	patch_size	3	-	3	-
LeNet-type AE	batch_size	8	-	8	-
	learning_rate	10^{-4}	-	10^{-4}	-
	epochs	100	-	100	-

Table C.6.: Hyperparameters for self-supervised AD baselines, own illustration adapted from [Bieg23].

	Hyperparameter	$\tilde{\mathbb{D}}_{\text{P1}}$	$\tilde{\mathbb{D}}_{\text{P2}}$
RotNet	batch_size	8	8
	learning_rate	10^{-1}	10^{-1}
	epochs	100	100
	weight_decay	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
	input_size	128×128	128×128
RotNet+PCA	n_components	5	64
RotNet+KPCA	n_components	9	10
	kernel	cosine	cosine
RotNet+OC-SVM	kernel	RBF	RBF
	γ	2^{-30}	0.125
	ν	0.005	0.005
RotNet+IF	n_estimators	50	50
	max_samples	64	16
NeuTraL AD	batch_size	8	8
	learning_rate	10^{-3}	10^{-3}
	epochs	20	20
	transformations	15	15
GeoTrans	batch_size	8	8
	learning_rate	10^{-3}	10^{-3}
	epochs	3	3
	input_size	128×128	128×128

C.2. Validation II

C.2.1. Monitoring protocols

Table C.7.: Monitoring protocol for SSMSPC, own illustration.

Machine cycle	Machine sub cycle	Process condition	Decision criteria	Identified root cause
1	P1			
	P2			
2	P1			
	P2			
3	P1			
	P2			

Table C.8.: Monitoring protocol for univariate post-process SPC, own illustration.

Machine cycle	Process condition	Decision criteria	Identified root cause
1			
2			
3			

C.2.2. SUS questionnaire

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
2. I found the system unnecessary complex.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
3. I thought the system was easy to use.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
4. I think that I would need the support of a technical person to be able to use this system.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
5. I found the various functions in this system were well integrated.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
6. I thought there was too much inconsistency in this system.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
7. I would imagine that most people would learn to use this system very quickly.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
8. I found the system very cumbersome to use.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
9. I felt very confident using the system.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
10. I needed to learn a lot of things before I could get going with this system.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5

Figure C.1.: SUS questionnaire, own illustration adapted from [Broo96].

C.2.3. Machining cycle sequence for test runs

Table C.9.: Machining cycle sequence for test runs, own illustration adapted from [Niek23].

Participant	Monitoring approach	Process condition
1	Univariate post-process SPC	1. Anomalous - crooked part 2. Anomalous - unprocessed surface 3. Normal
	SSMSPC	1. Normal 2. Anomalous - unprocessed surface 3. Anomalous - crooked part
2	SSMSPC	1. Anomalous - unprocessed surface 2. Anomalous - crooked part 3. Normal
	Univariate post-process SPC	1. Normal 2. Anomalous - crooked part 3. Anomalous - unprocessed surface
3	Univariate post-process SPC	1. Anomalous - unprocessed surface 2. Anomalous - crooked part 3. Normal
	SSMSPC	1. Normal 2. Anomalous - crooked part 3. Anomalous - unprocessed surface
4	SSMSPC	1. Anomalous - crooked part 2. Normal 3. Anomalous - unprocessed surface
	Univariate post-process SPC	1. Anomalous - unprocessed surface 2. Normal 3. Anomalous - crooked part
5	Univariate post-process SPC	1. Anomalous - unprocessed surface 2. Normal 3. Anomalous - crooked part
	SSMSPC	1. Anomalous - crooked part 2. Normal 3. Anomalous - unprocessed surface
6	SSMSPC	1. Anomalous - unprocessed surface 2. Normal 3. Anomalous - crooked part
	Univariate post-process SPC	1. Anomalous - crooked part 2. Normal 3. Anomalous - unprocessed surface
7	Univariate post-process SPC	1. Anomalous - crooked part 2. Normal 3. Anomalous - unprocessed surface
	SSMSPC	1. Anomalous - unprocessed surface 2. Normal 3. Anomalous - crooked part
8	SSMSPC	1. Normal 2. Anomalous - unprocessed surface 3. Anomalous - crooked part
	Univariate post-process SPC	1. Anomalous - crooked part 2. Normal 3. Anomalous - unprocessed surface