Efficient Deep Feature Learning for Noisy Industrial Time-Series Data

Mohamed Ali Tnani

TUM School of Computation, Information and Technology
Technische Universität München

Efficient Deep Feature Learning for Noisy Industrial Time-Series Data

Mohamed Ali Tnani

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 26.06.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 29.01.2024 angenommen.

Mohamed Ali Tnani. *Efficient Deep Feature Learning for Noisy Industrial Time-Series Data .* Dissertation, Technische Universität München, Munich, Germany, 2023.

Chair of Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de/`.

Acknowledgments

With this manuscript, I am bringing to a close one of the most valuable life experiences I have had, which has been a unique experience, full of emotions, challenges, and joys. For this reason, I want to dedicate this section to sincerely thank all those who made this experience possible and who contributed to the completion of this work.

First and foremost, I would like to thank Prof. Dr.-Ing. Klaus Diepold for giving me the opportunity to complete a doctoral thesis at our data processing chair. I am very grateful for his inspiring guidance and support during the doctoral period, which goes beyond the scope of this dissertation and pushes the self-development. It was a huge learning experience and a great pleasure to be part of the team.

I would like to thank my technical supervisors, Dr. Eduardo Monari and Dr. Matthias Meier, for mentoring me and pushing my limits despite the distance and telecommuting conditions. Their encouragement and invaluable advice kept me on track and able to accomplish this work.

I would also like to thank all the colleagues and doctoral students at Bosch Rexroth who supported and helped me during my research, even when it came to critical topics related to open-sourcing.

A special thanks to my fellow doctorates at the data processing chair for their warm welcome, supportive discussions, and for explaining the hidden rules of the doctorate. I am so grateful for the academic experience I shared with them at our chair that made this experience so enjoyable even though I am still waiting for the server room tour.

I would like to thank my family and friends for their support during my studies and all the energy they provide me everyday. I would like to thank my father and my brothers, who taught me to always stay strong while striving for excellence. I would like to thank my mother, who supported me every step of the way.

Finally, I would like to thank all the people who have contributed in one way or another to this journey.

Abstract

Driven by digital transformation and recent advances in the Industrial Internet of Things, industrial processes such as quality control and machine monitoring are undergoing a rapid evolution towards intelligent automation and increased productivity. Recent achievements in the field of signal processing have led to promising state-of-the-art data-driven techniques for machine process monitoring.

However, given the environmental and industrial challenges, the application in real-world production is highly complex and requires a considerable amount of expertise, from the early phase of data annotation to the extraction of meaningful features from the sensory data. In addition, data-driven models suffer from poor generalizability and robustness against data drift and high process variety.

To address manufacturing challenges, this thesis researches different deep learning methods for noisy industrial time-series data. It first investigates the various industrial and environmental hurdles faced in real-world applications and proposes a novel benchmark dataset that embeds these challenges. In the second part, an in-depth study of unsupervised deep learning methods is performed and a new deep learning architecture, characterized by a small number of parameters and a rich information extraction capability, is proposed. The unsupervised trained model is then embedded in a two-stage learning method that improves the learning mechanism using a limited amount of annotations. The architecture investigation results prove that when dealing with noisy time-series data, the width of the network architecture is an essential parameter where the performance of wide networks exceeds the deep networks. The study further confirms the superiority of supervised methods over unsupervised methods and demonstrates that the two-stage learning method is an efficient and reliable approach for anomaly detection in machine processes. It outperforms conventional supervised and unsupervised methods as well as handcrafted feature models, in terms of robustness and generalizability, using a very limited annotated dataset which reduces the labeling cost significantly.

kurzfassung

Angetrieben von der digitalen Transformation und den neuesten Fortschritten im industriellen Internet der Dinge entwickeln sich industrielle Prozesse wie Qualitätskontrolle und Maschinenüberwachung in Richtung intelligenter Automatisierung und höherer Produktivität. Aktuelle Errungenschaften auf dem Gebiet der Signalverarbeitung haben zu vielversprechenden modernen datengesteuerten Techniken für die Überwachung von Maschinenprozessen geführt.

Allerdings sind reale Produktionsanwendungen angesichts der industriellen Herausforderungen komplex und erfordern ein erhebliches Maß an Fachwissen, von den frühen Phasen der Datenbeschriftung bis hin zur Extraktion relevanter Merkmale aus Sensordaten. Darüber hinaus leiden datengesteuerte Modelle unter mangelnder Skalierbarkeit und Robustheit gegenüber Datendrift und hoher Prozessvielfalt.

Um die Herausforderungen in der Fertigung zu bewältigen, werden in dieser Arbeit verschiedene Deep-Learning-Methoden für verrauschte industrielle Zeitreihendaten untersucht. Die Forschung untersucht zunächst die verschiedenen industriellen und externen Hürden, die in realen Anwendungen auftreten, und stellt einen neuen Benchmark-Datensatz vor, der diese Herausforderungen berücksichtigt. Im zweiten Teil wird eine eingehende Studie über unüberwachte Deep-Learning-Methoden durchgeführt und eine neue Deep-Learning-Architektur vorgeschlagen, die sich durch eine geringe Anzahl von Parametern und eine umfassende Fähigkeit zur Informationsextraktion auszeichnet. Die unsupervised trainierten Modelle werden dann in eine zweistufige Lernmethode eingebettet, die den Lernmechanismus unter Verwendung eines begrenzten Satzes von Annotationen verbessert. Die Ergebnisse der Architekturstudie zeigen, dass bei verrauschten Zeitreihendaten die Breite der Netzwerkarchitektur ein essentieller Parameter ist, bei dem die Leistung von breiten Netzwerken die von tiefen Netzwerken übertrifft. Die Studie bestätigt auch die Überlegenheit supervised Methoden gegenüber unsupervised Methoden und zeigt, dass die zweistufige Lernmethode ein effizienter und zuverlässiger Ansatz zur Erkennung von Anomalien in maschinellen Prozessen ist. Es übertrifft traditionelle Supervised und Unsupervised Methoden sowie handgefertigte Merkmalsmodelle in Bezug auf Robustheit und Verallgemeinerbarkeit, wobei ein sehr begrenzter annotierter Datensatz verwendet wird, der die Kosten für das Labeling erheblich reduziert.

Contents

1 Introduction

Steered by technological progress, the industrial field has experienced a series of revolutions in recent decades with the common goal of increasing productivity, quality and automation. Being an important pillar of the industry, quality control has also benefit from this paradigm and is slowly shifting from being at the end-of-production testing station to the production cell.

Thanks to advances in the Industrial Internet-of-Things (IIoT) and the proliferation of sensors in every machine and component, quality inspection becomes possible instantaneously during production and defects that were previously undetectable are revealed [4, 22]. This reduces the likelihood of human error, and ensures consistency in the quality of the product.

This research is not to eliminate manual labour in the quality control process, but on the contrary to empower the control engineer with analytical methods to support quality control and move towards augmented intelligence where human and artificial intelligence would partner together.

1.1 Motivation

Significant developments in artificial intelligence (AI) and, in particular, signal processing methods have further advanced the idea of smart manufacturing by means of data-driven methods designed to support online and offline machine monitoring. These techniques have the potential to analyse large amounts of data from machines in real time and identify potential faults and anomalies.

However, being a critical topic, quality control tools require very high precision and reliability. In industries such as the automotive industry, where customer safety is of highest importance, there are stringent production control measures in place to ensure the quality and safety of the automotive systems. Although state-of-the-art data-driven models have shown promising results in the scientific field, they often do not meet the high expectations in real-world application [51]. The primary challenge faced by AI applications in manufacturing is data drift. Data drift refers to the change in the input data over time or between different systems, which leads to model degradation [20, 45]. Apart from the issue of data drift, one of the significant challenges faced by AI applications in the industrial sector is the requirement for expert human input and domain knowledge.

A high level of human expertise is required to annotate events in noisy time-series

(TS) data and extract meaningful patterns. Annotating refers to the process of labeling specific time intervals or sequences in the data that correspond to specific events and actions. For example, in the context of process monitoring, events could include the start and end of a specific step in the process, or the occurrence of a particular anomaly during production.

Identifying relevant information and patterns from the TS data is commonly referred to as "Feature Extraction" and is crucial for effective process and machinery monitoring, as the derived features must be relevant and representative for the specific application with minimal loss of information. In the context of noisy TS data, examples of features are statistical patterns such as standard deviation, peak-to-peak, kurtosis, ect. It is important to note that the task of feature extraction also consists of a number of data manipulation steps to ensure that the data is in a suitable format for analysis, such as data filtering, which aims to remove unwanted noise or irrelevant information, and data transformation, which converts the raw data from one domain into another, or feature selection analysis, which aims to select the most informative features, etc.

Industrial TS data, such as vibration data, is the most common type of data in manufacturing and is characterised by its noisiness and non-stationarity [7, 20]. An example of vibration data collected is illustrated in Fig. 1.1. As previously mentioned, annotating these type of data is a challenging and time-consuming task, especially when compared to the labeling of textual or visual data. It requires deep understanding of the underlying physical process, as well as expertise in data analysis.

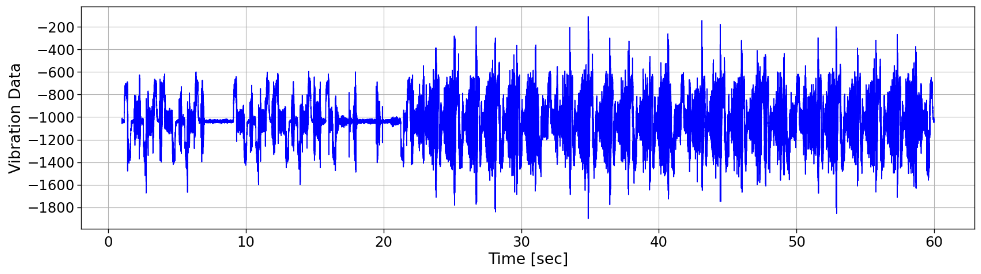


Figure 1.1: Example of noisy and non-stationary time-series data: Vibration data with a sampling rate of 2 kHZ

The TS data addressed in this thesis are collected from Computer Numerical Control (CNC) machines (see Fig. 1.2), which are characterised by their high accuracy, robustness and longevity. These characteristics add complexity to the application in several ways. Firstly, the aging of machine components over time is a common occurrence and leads to changes in the machine's behavior and thus to data drift. Secondly, high accuracy and robustness involves a low rate of failures such as chip jamming, tool breakage, or improper tool clamping, etc. [51]. This leads to a large unbalance between the normal and abnormal samples.

To address these challenges, deep learning methods are investigated in this thesis.

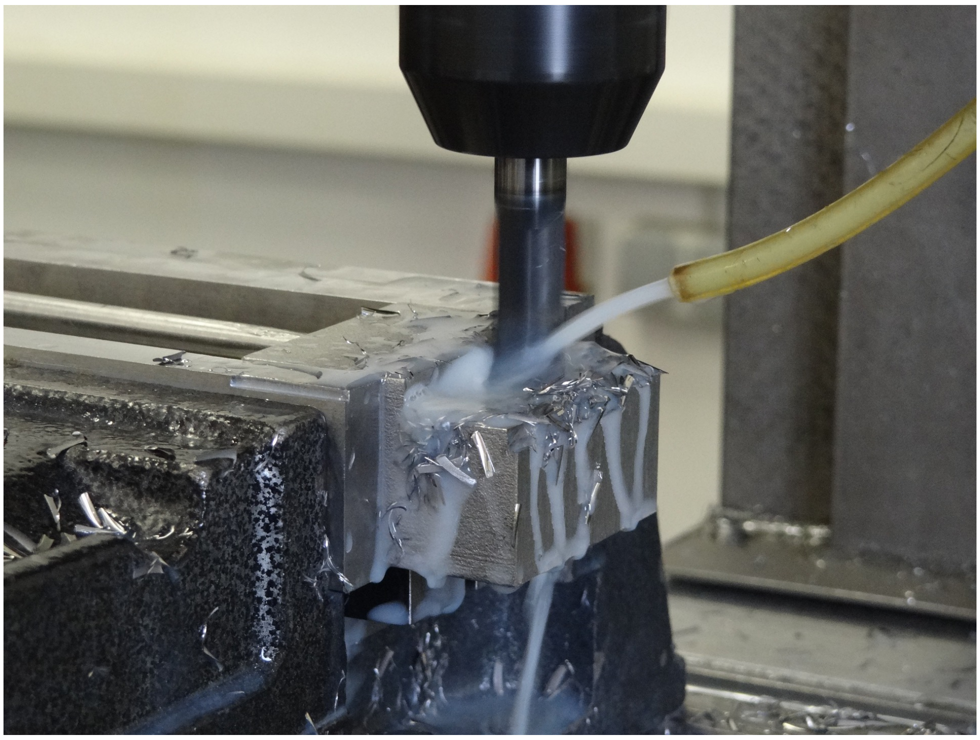


Figure 1.2: Overview of the extreme environment during high-speed manufacturing in CNC machines [56].

Deep Learning (DL) has shown promising results in computer vision, natural language processing, and speech recognition. Their ability to learn from large amounts of data and extract meaningful features can be of great benefit to industrial applications by reducing the need for manual feature engineering and improving the performance and efficiency of process monitoring techniques.

However, their application in industrial sensor data is characterized by its non-acceptance due to the "black box" nature and lack of interpretability of these methods [15, 52]. The application of DL methods to noisy TS data is further hampered by the lack of annotated data and pre-trained Feature Extractors (FEs) which can enhance the robustness and accuracy of the model. In comparison to other domains, such as computer vision where large datasets with labeled data are readily available, the manufacturing domain has fewer annotated datasets, making it challenging to train DL models. This results in the need for data collection and annotation, which can be a time-consuming and expensive process.

In this research work, we address the topic of feature extraction on noisy industrial TS data and propose methods to tackle the lack of pre-trained FEs by utilizing zero-label and limited labeled data.

1.2 Research Questions

The aim of this research is to improve the application of machine learning methods in real-world machine monitoring applications by reducing the need for domain knowledge and data labeling overheads, and investigating the generalization and robustness capabilities of deep learning techniques. Due to the lack of publicly available datasets from real-world production, this work first contributes with a novel benchmark dataset for machine monitoring which incorporates the real-world challenges such as data drift and the high variety of processes. This dataset will be used as benchmark to answer the research questions. The main research questions of this dissertation are the following:

RQ 1: What are the real-world challenges with regard to data-driven machine monitoring?
Data-driven techniques face several challenges in the field of industrial health monitoring caused by the extreme environment and complexity of the processes. The answer to this research question highlights the key challenges to consider when evaluating and validating data-driven models for industrial applications.

RQ 2: Can handcrafted features extracted from industrial time-series data be bypassed by learned features extracted using deep learning techniques?
Current state-of-the-art models rely on domain experts to extract meaningful features from raw industrial data and are therefore very application specific. DL

methods reduce domain knowledge and take the raw data as input to extract learned features. The answer to this research question delivers a more generic methodology that minimizes the required domain expertise and increases the scalability on unseen applications.

RQ 3: Do depth and width of state-of-the-art DL blocks improve the feature learning of noisy time-series data?
Advances in the field of DL offer a variety of DL blocks and layers, from conventional convolutional blocks to more complex blocks. As I investigate DL feature extraction methods for noisy TS data, this research question provides an in-depth analysis of different DL blocks and their performance, taking into account manufacturing requirements.

RQ 4: Does fine-tuning pre-trained FEs improve performance and robustness in the context of noisy time series data?
In a real-world scenario, huge amounts of TS data are available in production, and in most cases, only a limited set of data is labeled. Research question 4 addresses the fine-tuning of pre-trained models and their ability to improve based on a limited set of labels. It provides a training methodology with a limited amount of labeled data and a quantitative overview of the annotations required to achieve satisfactory performance.

1.3 Dissertation Structure

As mentioned in the previous sections, this dissertation aims to overcome existing challenges in machine monitoring applications by investigating DL methods for noisy TS data. The research has been divided into four main chapters and the structure is illustrated in Fig.1.3 . Chapter 2 discusses the latest advancements in the scientific field. Furthermore, the fundamental concepts and methods used in this thesis are presented. Chapter 3 describes the experimental setup and case study of the thesis. RQ1 is answered in this chapter through an investigation of ML challenges in manufacturing, which are then embedded in the experimental dataset. Chapter 4 introduces the research on feature learning by presenting the results of unsupervised feature learning methods. An intensive investigation of state-of-the-art DL architectures addresses RQ3 and an evaluation of the performance of unsupervised models compared to handcrafted FEs partially addresses RQ2. Chapter 5 extends the feature learning research by examining the performance of few-shot learning methods in combination with unsupervised learning methods, thus answering RQ2 and RQ4. Finally, a concluding chapter summarises the findings of this thesis and provides an overview of future work.

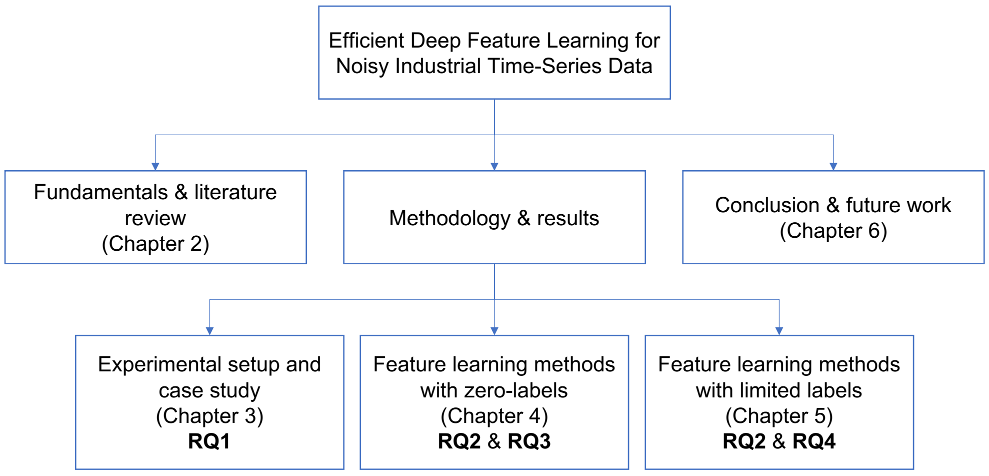

Efficient Deep Feature Learning for Noisy Industrial Time-Series Data
Fundamentals & literature review (Chapter 2)
Methodology & results
Conclusion & future work (Chapter 6)
Experimental setup and case study (Chapter 3) RQ1
Feature learning methods with zero-labels (Chapter 4) RQ2 & RQ3
Feature learning methods with limited labels (Chapter 5) RQ2 & RQ4

Figure 1.3: Overview of the dissertation structure.

1.4 Publications List

The main chapters of this dissertation are based on separately published work, including three research papers, a patent, and a dataset. These contributions represent the core of the research presented in this dissertation.

1. Smart Data Collection System for Brownfield CNC Milling Machines: A New Benchmark Dataset for Data-Driven Machine Monitoring [64].
   Mohamed-Ali Tnani, Michael Feil, and Klaus Diepold.
   Elsevier Procedia CIRP 2022.

2. Detecting When a Piece of Material is Caught between a Chuck and a Tool.
   Mohamed-Ali Tnani, Benjamin Menz, and Scott Hibbard.
   Patent Application 2021.

3. Extract, Compress and Encode: LitNet an Efficient Autoencoder for Noisy Time-Series Data [66].
   Mohamed-Ali Tnani, Paul Subarnaduti, and Klaus Diepold.
   IEEE International Conference on Industrial Technology 2022.

4. Efficient Feature Learning Approach for Raw Industrial Vibration Data using Two-Stage Learning Framework [65].
   Mohamed-Ali Tnani, Paul Subarnaduti, and Klaus Diepold.
   MDPI Sensors 2022.

The following publication is published during the doctoral program, but is not made part of this thesis:

1. Active Transfer Prototypical Network: An Efficient Labeling Algorithm for Time-Series Data [83].
   Yuqicheng Zhu, **Mohamed-Ali Tnani**, Timo Jahnz, and Klaus Diepold.
   Elsevier Procedia Computer Science 2023.

2 Fundamentals and Contemporary Developments

Encouraged by the outstanding results in various fields, Machine Learning (ML) in manufacturing has attracted great interest, with machine monitoring emerging as a prominent trend. In this chapter I present the fundamental and theoretical concepts of this thesis as well as the latest achievements of machine monitoring systems and ML in manufacturing in a broad perspective. In each main chapter, detailed related work and background information can be found.

2.1 Machine Monitoring

Highly automated machining centers are one of the main pillars of current industry thanks to their high efficiency in mass production. Nevertheless, failures during machining such as tool breakage, improper tool clamping, chatter, are inevitable and minor deviations in the final product result in significant costs and time-consuming rework and scrap [4, 51]. With the goal of automating fault detection and reducing downtime, monitoring of machining systems and processes has been intensively researched, resulting in a variety of approaches.

2.1.1 Classification of monitoring techniques

Maintenance and management of complex process equipment and manufacturing processes have become imperative in modern industry. The major objectives are to ensure the human safety and the environment, as well as to guarantee the timely production and delivery of products that meet high quality standards [2]. Additionally, the demand for cost-effective solutions grows, leading to a need to prioritize energy efficiency and environmentally friendly practices in industrial processes. This necessitates the development of systems that can optimize equipment performance and promote sustainable industrial processes [17].

In machining maintenance, three types of maintenance can be identified: reactive, preventive, and predictive [50]. Reactive maintenance is performed when a machine fails and requires immediate attention, leading to high expenses and decreased production availability. Preventive maintenance is done periodically to replace components, ensuring the machine remains in optimal condition. It increases the production availability but at higher costs. Predictive maintenance, based on machine sensory data, allows for fault diagnosis and predicting remaining useful life which induces costs reduction and improving production availability. Predictive maintenance is often referred to as "condition monitoring", which involves the identification of deviations or changes from the typical operating state of a machine.

Process condition monitoring, often referred to as "quality monitoring" or "quality control," is a distinct aspect of condition monitoring. Process monitoring focuses on monitoring the performance of the machining process to ensure that it meets the desired quality standards.

The case study presented in this thesis belongs to the category of process monitoring. However, the dataset provided in this work can also be utilized for condition monitoring research, such as examining the degradation of machining tools thanks to the wide variety of tools available and the presence of a challenging real-world data drift.

2.1.2 Time-series data in machine monitoring

State-of-the-art research on CNC machine health monitoring has explored various types of data, among them temperature, vibration, acoustic emissions, and cutting force, recorded with thermocouples, accelerometers, microphones, and dynamometers, respectively. The most commonly used type of TS data in machine monitoring is the cutting force, as it perfectly reflects the condition of the machine and has a high measurement accuracy. Cutting force has been used for machine monitoring in several researches, such as real-time chatter detection for milling processes [72], and tool condition monitoring and life cycle prediction [24, 48, 69]. However, it is measured using dynamometers, which are sensitive to the extreme environmental conditions of CNC machines and are characterised by their high cost [55].

Being the second most common type of time series data in machine monitoring, vibration data generated by accelerometers are characterised by their low cost and low power consumption [26]. It is widely used in process condition monitoring to detect process anomalies [41], breakage [9], wear [30], chatter [18, 33], surface roughness [31], etc. However, vibration data is characterized as noisy, nonstationary, and nonlinear and requires robust expertise to initially label the data. This requires a significant amount of effort and a high degree of accuracy to avoid mislabeling that leads to degradation of the subsequent learning process [57].

In addition to vibration signal and cutting force, acoustic emission (AE) signal is widely used in health monitoring [32, 49]. Nevertheless, AE sensors are considered sensitive to the production environment and require meticulous calibration to avoid undesirable noises [14]. In addition, visual data obtained from optical microscopes and machine vision technologies have yielded some promising results in research [23, 43]. However, these techniques face a major challenge when dealing with a real-world environment where disturbances from the cutting fluid and chips affect the data quality and thus the performance of the models [14].

In this thesis, vibration data obtained from accelerometers mounted in the machines is used as monitoring method. This is first because of its low cost, second because of its robustness against extreme environmental conditions and third because of its high information content compared to other data sources [41].

2.1.3 Open-source datasets

Open source datasets play a critical role in advancing research and development in a variety of fields. In the manufacturing industry, the availability of high-quality and comprehensive datasets is equally important to drive innovation and advances in ML and industrial analytics.

Although the availability of open-source datasets in the industrial TS field is relatively limited compared to other domains, several datasets have been made accessible to the research community. A good collection of industrial and manufacturing-related data, such as sensor data from machining processes, quality control measurements, and equipment performance data have been released at the University of California UCI [3]. In 2011, Randall et al. [53] released a fault diagnosis and condition monitoring dataset for rotatory machinery, comprising vibration and temperature sensor data collected from bearings. More recently, the University of Michigan published a machining condition monitoring dataset, specifically focused on detecting tool wear and clamping failures in milling machines [46]. Additionally, there is a similar milling dataset available that allows for the study of tool wear using three types of sensors: acoustic emission, vibration, and current [1]. The University of Paderborn has similarly published a widely researched dataset for milling systems [37]. This dataset includes a series of machining parameters recorded during milling operations, including cutting forces, spindle speed and feed rate.

The existence of these datasets opens up opportunities for numerous studies in the field of machine monitoring. However, it should be noted that these datasets have been collected from experimental setups over a relatively short period of time, which does not fully capture the challenges and complexities of real-world scenarios, such as data drift over time and the natural degradation of machining components. In order to address this limitation and facilitate research on scalability and robustness of data-driven methods, this thesis introduces a dataset collected from real-world production over a period of two years, providing a valuable resource for future research.

2.2 Data-Driven Machine Monitoring

Driven by the digitization of the industrial world, data-driven models have overtaken traditional physical models in manufacturing. Due to the complexity and noise of manufacturing systems, traditional monitoring systems fail to model complex and flexible dynamic or rule-based systems when used in the real-world environment [80]. In contrast, data-driven models such as ML models learn from historical data through extracting useful information and are distinctive by their adaptability and continuous learning.

2.2.1 Traditional machine learning for industrial time-series

ML methods have been widely used in manufacturing over the past few decades. Traditional ML methods basically start with a preprocessing phase, followed by a feature extraction and selection phase, and a final decision making phase [13]. The preprocessing phase aims to clean and standardize the sensory input data using signal processing techniques such as noise reduction, amplification, interpolation, normalization, etc. The feature extraction phase is the most fundamental step, since it consists in manipulating the noisy input data and constructing a set of features that contains the most relevant information for the decision-making function, thus the accuracy of the model strongly depends on the extracted features [4].

Feature extraction

In machine monitoring, the handcrafted features are divided into time domain, frequency domain and time-frequency domain. Time domain features are widely used in time series ML and extract relevant information from the raw data, such as mean, peak-to-peak, root mean square (RMS), standard deviation, skewness, kurtosis, tooth frequency, power spectrum, etc. However, these features have a poor robustness and are sensitive to disturbances [59]. Among the features in the frequency and time-frequency domain, one can cite the Fast Fourier Transform (FFT) and the Short-Time Fourier Transform (STFT), which are among the most commonly used signal transformations and have proven to be very powerful, especially in predicting tool wear [51]. However, extracting these features, e.g., identifying the spectral bands sensitive to tool wear, is very challenging, time-consuming, and requires a lot of expertise [18, 51]. This makes it an application-specific and non-scalable process.

With the goal of automating the process and reducing human intervention, Christ et al. [10] developed a ML tool called TSFRESH (Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests) that allows automatic extraction and selection of TS features given statistical methods, features of the sampling distribution, and observed dynamics. These features capture various aspects of the time series data, such as trend, seasonality, and statistical properties and feature selection is made using a scalable hypothesis tests. The main drawback of this method remains the high computation time, especially when it comes to high frequency data such as vibration data. Additionally, TSFRESH relies on statistical hypothesis tests to select relevant features. It assumes that the underlying data distribution and statistical properties are consistent. In cases where the data violates these assumptions or shows non-stationarity, which is common in noisy TS data, the selected features may not accurately capture the relevant patterns.

Within the research community, there are alternative options available to TSFRESH. One such alternative is TSFEL (Time Series Feature Extraction Library), introduced by Barandas et al. [6], which offers a similar feature extraction capability. TSFEL aims to enhance computation time and provides an extensive selection of features encompassing statistical, frequency, and information-theoretic measures across various temporal, statistical, and spectral domains. Fulcher et al. [19] have introduced a library known as HCTSA (Highly Comparative Time-Series Analysis) with the purpose of aiding in the exploration of new features and the advancement of novel TS analysis techniques. While TSFRESH focuses on automated feature extraction with a focus on efficiency, HCTSA aims to enhance the understanding and analysis of TS data facilitating comparison and exploration of a wide range of features.

Similar to TSFRESH, these approaches encounter computational challenges when dealing with high-frequency TS due to the extraction of a large number of features. Moreover, the performance of handcrafted feature extractors relies on the quality and preprocessing of the input TS data. The presence of noise, outliers, or missing values can compromise the reliability and interpretability of the extracted features. Another limitation of these tools is their reliance on predefined sets of feature extraction methods, which may not cover all possible feature engineering techniques or specific requirements of a particular application.

Decision making

In the conventional ML pipelines for machine monitoring, the final stage involves the application of a decision-making method to classify or make predictions based on the handcrafted extracted features. Support Vector Machines (SVMs) have emerged as one of the widely used methods in the literature for this purpose [11, 26, 42, 70]. SVMs are advantaged thanks to their strong performance in high-dimensional feature spaces, which is particularly relevant in machine monitoring tasks where a large number of features are extracted from sensor data. One key advantage of SVMs is their ability to provide insights into the importance of different features in the classification process, thus enhancing the interpretability of the decision-making results [11].

Another commonly used method in condition monitoring is Kernel Density Estimation (KDE) [21, 35, 60, 77]. Zhang et al. [77] compared the performance of KDE with SVMs in their research and demonstrated the superiority of KDE. They proposed a KDE method utilizing Kullback-Leibler divergence to quantitatively measure the similarity between two estimated distributions, enabling effective classification of the extracted features. The utilization of KDE in condition monitoring showcases its applicability and effectiveness in capturing complex patterns and variations in the monitored systems.

Decision Trees have been applied in various fault diagnosis applications for induction motors, gearboxes, hydraulic pumps, and other systems over the course of several decades [36]. These applications have demonstrated the benefits of Decision Trees, including their transparency and interpretability. However, there is still room for improvement in terms of their generalization performance. Furthermore, Hidden Markov Models (HMMs) knew significant traction, particularly in manufacturing, due to their ability to learn patterns in an unsupervised manner, making them suitable when obtaining labels is challenging [16, 47, 74]. HMMs are characterized by modeling sequential data, allowing them to capture the temporal dependencies and dynamics present in process variables over time and thus understanding the evolving nature of the monitored system and detecting deviations and anomalies.

Although traditional ML methods have demonstrated their effectiveness in machine monitoring and condition monitoring, they often face limitations in terms of generalization capabilities and their performance heavily relies on the quality and relevance of the handcrafted features that are extracted from the data. Consequently, the success of the models is closely tied to the expertise and domain knowledge of the feature engineering stage.

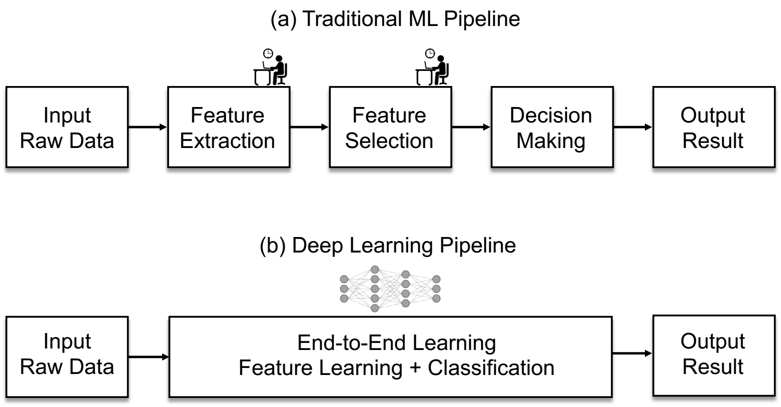

(a) Traditional ML Pipeline
Input Raw Data
Feature Extraction
Feature Selection
Decision Making
Output Result
(b) Deep Learning Pipeline
Input Raw Data
End-to-End Learning Feature Learning + Classification
Output Result

Figure 2.1: Overview of the intelligent framework: Comparison of the traditional versus DL techniques [68].

2.2.2 Deep learning for industrial time-series

With the increasing volume of manufacturing data and the need to handle the complexities of feature extraction, there has been a growing interest in DL methods in several fields, including the manufacturing industry research community [51, 57, 80]. Thanks to its data-driven nature, nonlinearity, adaptability, automatic noise reduction, and hierarchical representation learning, DL has proven to be a powerful approach to handle large amounts of data and build robust feature extractor in an end-to-end manner [59, 68]. As shown in Fig. 2.1, in contrast to classical ML solutions, DL builds deep structured models that start learning from the initial input phase to the final decision phase, tremendously reducing the complexity of the learning framework. The output model learns the distribution of the input data and, through its hierarchical learning framework, starts extracting low to higher level features, undergoing multiple data transformations.

Advancements and architectures

In the field of machine monitoring, DL architectures have received significant attention in recent years. Various DL models have been explored to address the challenges of analyzing noisy industrial data, particularly vibration data characterized by shift-variance, and fault detection applications for rolling element bearings, gears, motors, and hydraulic pumps have benefited from these advancements [80].

A very common architecture is the 1D CNN (Convolutional Neural Network) which has demonstrated remarkable performance in handling such noisy TS data with limited preprocessing steps and wide first-layer kernels [79]. Janssens et al. [28] conducted a comparative analysis between the performance of features learned by convolutional layers and manually extracted features in a machining fault diagnosis application. Their study demonstrated an improvement in accuracy when using CNN-learned features obtained through fully supervised training manner. Building upon the performance of the 1D CNN, researchers have further focused on designing architectures specifically tailored for TS data.

For instance, Li et al. [39] introduced the WaveletKernelNet architecture, which incorporates wavelet transformations directly into the convolutional layers. This approach takes advantage of wavelet analysis, enabling the extraction of both time and frequency information from the signals. However, it is important to note that this method also demands considerable computational power due to the additional transformation involved.

Leveraging the power of convolutional operations to capture temporal dependencies and patterns in sequential data, Temporal Convolution Networks (TCNs) [34]. TCNs use dilated convolutions to capture long-range dependencies effectively which offers a larger receptive field and captures information across different time scales. TCNs present a robust alternative to traditional Recurrent Neural Networks (RNNs) and their variants, such as Long Short-Term Memory (LSTM) networks, offering notable advantages in terms of computational efficiency and processing speed.

Another common convolutional based architectures is the Inception architecture which aims to capture local and global patterns in the input data using multiple parallel convolutional layers with different kernel sizes [61]. The Inception architecture employs a combination of different sized convolutions along with pooling operations to extract meaningful features at different scales. This architecture was originally introduced for image classification tasks in the field of computer vision [62]. However an extension of the Inception architecture specifically designed for TS classification called InceptionTime has been proposed by Fawaz et al. [27]. It adapts the principles of the Inception architecture to the time domain by replacing the 2D convolutions with 1D convolutions having larger kernel sizes as shown in 2.2. The inclusion of a bottleneck layer in the network architecture serves the purpose of dimensionality reduction, resulting in a more parameter-efficient model that mitigates the risk of overfitting, particularly when working with limited training data. This design proves to be highly effective in the analysis of time series data, as it facilitates the extraction of pertinent features from various levels of detail. By capturing both short-term and long-term dependencies, the inception architecture represents a good choice for robust feature extraction for TS data.

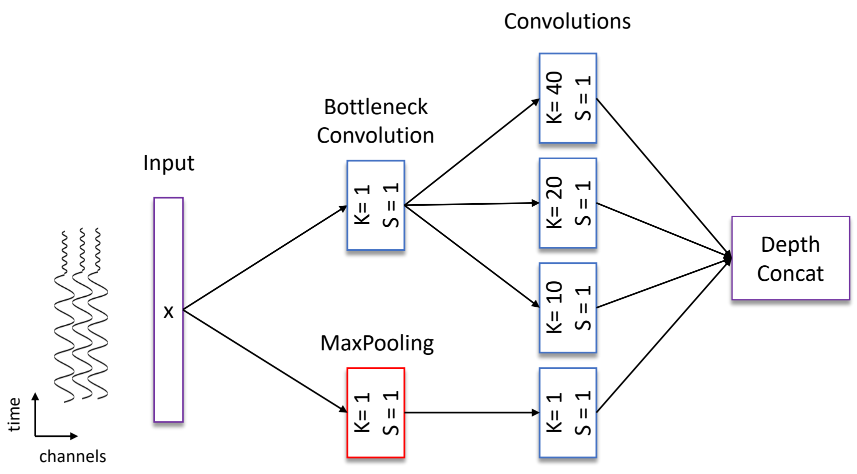


Figure 2.2: Overview of the multi-level convolutional structure employed in the InceptionTime network [27]. K represents the kernel size, and S corresponds to the stride used in the convolutional layers.

Other DL architectures intensively used in machine monitoring are Multi-Layer Perceptions (MLPs) [12], known for their feedforward architecture nature, Deep Belief Networks (DBNs) [18, 38], known for their hierarchical data representation, and mostly Autoencoders (AEs) [29, 44].

AEs are one of the most commonly used types of DL methods in manufacturing [73]. They consist of two main components: an encoder and a decoder as illustrated in 2.3. The encoder takes the input data and maps it to a lower-dimensional latent space representation, while the decoder reconstructs the input data from this latent representation. The representation learning is achieved using a reconstruction loss function that measures the difference between the original input and the reconstructed output.

AEs are commonly employed for detecting anomalies when the model struggles to accurately reconstruct the input "anomalous" data. This utilization highlights their effectiveness in identifying deviations from normal behavior. However, another notable advantage of AEs lies in their ability to learn meaningful representations of the data, which enhances their potential for feature extraction and learning the underlying patterns in input data. In fact, the latent space representation, also known as the bottleneck layer, acts as a compressed and encoded version of the input data. It typically has a lower dimensionality than the input and serves as a compressed representation that captures the essential features of the data.

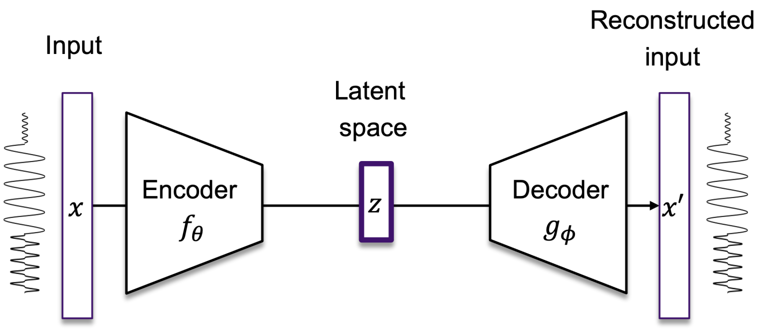


Figure 2.3: Overview of the basic structure of autoencoders.

2.2.3 Limited data in machine monitoring

Obtaining sufficient faulty data to train machine monitoring models poses challenges due to the infrequency of faults, particularly in high-automated machines, as well as the cost and labor associated with conducting fault simulation experiments [40, 78]. Zhao et al. [81] evaluated the performance of the popular architectures on several industrial TS benchmark datasets and concluded that DL-based methods outperform traditional ML methods on several tasks, but only when the training dataset is relatively large. In response to this challenge, researchers have been actively exploring novel methods to train data-hungry DL models using limited amounts of annotated data.

One approach involves the development of data augmentation techniques [63]. These techniques aim to enhance the training data by artificially expanding its size and diversity, thereby providing the DL models with a larger and more representative dataset to learn from. Um et al. [67] conducted a study in the field of sensor data where they implemented different data augmentation techniques, including time warping, random scaling, and noise addition. They demonstrated that these strategies were effective in augmenting the limited available data, leading to improved model training with enhanced robustness and accuracy. Additionally, emerging techniques utilizing generative adversarial networks (GANs) showed encouraging outcomes in addressing the class imbalance issue in industrial machining data. These approaches involve generating faulty samples from a combination of data distributions, effectively restoring balance and enhancing the performance of the models [8, 75, 81, 82]. As an example, Zhou et al. [82] proposed a global optimization GAN that generates more qualified samples for fault diagnosis, improving the accuracy of the model for residual useful life forecast. However, GANs, and data augmentation techniques in general, have certain disadvantages. These include the potential loss of original information as artificial patterns or distortions are introduced into the data, as well as an increased risk of overfitting, particularly in cases where the data samples from the same process type is highly similar especially in the repetitive manufacturing scenarios [71].

In addition to data augmentation, successful advancements have been made in unsupervised learning field, such as the previously mentioned autoencoders, which have proven effective in learning features from no labeled data in the manufacturing domain [42]. Using a CNN based AE, Shaheryar et al. [58] built a semi-supervised approach for fault identification in rotatory machines. The proposed method presents a fine-tuned ensemble of AEs trained separately for each vibration axis signal and proved to be robust against noise present in vibration data. In the same domain, Huang et al [25] evaluated RNN-based AEs over CNN-based AEs in the context of feature learning and showed the superiority of RNNs in data compression performance and the CNNs in their generalization ability. In [5], the authors proposed an LSTM-based approach that utilizes multiple autoencoders (AEs) to estimate the remaining useful life of equipment for condition-based maintenance planning. This approach demonstrated a significant reduction in preventive stoppages. However, the effectiveness of the approach is dependent on the availability of high-quality datasets, which can be challenging to obtain in real-world applications. Furthermore, AEs are susceptible to overfitting when trained on a limited dataset due to their training process that involves minimizing the reconstructed empirical risk [36].

Another technique developed specifically for dealing with the scarcity of available data is Meta-Learning, also referred to as learning to learn [63]. It involves training a model on multiple small tasks with a limited number of training samples per class. By doing so, the model learns to generalize from past tasks and quickly adapt to new, unseen tasks, resulting in enhanced performance and the ability to generalize effectively with limited annotated data. Consequently, Meta-Learning shows great potential as a valuable approach for machine monitoring applications where fault samples are very limited. Recent research in this field further supports its effectiveness and relevance [78]. In the field of rolling bearing fault diagnosis, Zhang et al. [76] proposed a siamese-based few-shot approach to effectively diagnose faults by measuring feature vector similarity. Their method demonstrates superior performance compared to the baseline model, particularly for new unseen fault types or working conditions. Similarly, Ren et al. [54] introduced a novel method called the capsule auto-encoder model, combining few-shot learning and autoencoder capabilities. This model extracts meaningful feature capsules and utilizes dynamic routing to represent health indices, showcasing its ability for few-shot learning and quick updating in fault diagnosis and noisy environments.

Overall, these state-of-the-art methods address the challenges of limited data availability by enhancing the training data, extracting meaningful features from unlabeled data, and enabling models to generalize and adapt to new classes or faults with minimal labeled examples. However, they still have some limitations such as lack of robustness, and interpretability or rely on performant pre-trained FEs trained on large datasets.

3 ML Challenges in Manufacturing

The motivation for this research lies in the poor performance of data-driven machine monitoring methods in manufacturing applications. As mentioned in the previous section, the inferior performance in real-world is mainly due to the extreme industrial environment and its challenges. The first journal paper answers the research question 1 by defining and examining these challenges caused by environmental and industrial factors. These challenges have been embedded in a novel benchmark dataset and made publicly available at `https://github.com/boschresearch/CNC_Machining`. A detailed description of the dataset as well as its main contributions to the research community can be found in the paper.

In addition to the challenges, the paper presents the experimental setup of the thesis. It illustrates the intelligent edge-to-cloud data collection system used in this work, which enhances collaboration between the domain experts on the plant side and the data scientists on the cloud side. The experimental setup is the same as used for the rest of this research, and the ML models developed in the next sections will be implemented and evaluated in the aforementioned architecture.

My contributions in this work are the conceptualization of the data collection system, the data collection and annotation of the different measurements, the state-of-the-art research of the existing datasets, the formulation of the challenges as well as the analysis of the obtained dataset. The research work was supervised by Klaus Diepold. Michael Feil was involved in the implementation of the edge-to-cloud modules. In the paper, he also contributed to the Software Architecture section which was reviewed and reworked by me to meet the post-submission requirements of the reviewers.

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 107 (2022) 131–136

www.elsevier.com/locate/procedia

55th CIRP Conference on Manufacturing Systems

Smart Data Collection System for Brownfield CNC Milling Machines: A New Benchmark Dataset for Data-Driven Machine Monitoring

Mohamed-Ali Tnani*[a,b], Michael Feil[c], Klaus Diepold[b]

[a]*Department of Factory of the Future, Bosch Rexroth AG, Lise-Meitner-Str. 4, 89081 Ulm, Germany*
[b]*Department of Electrical and Computer Engineering, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany*
[c]*Department of Informatics, Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Germany*

* Corresponding author. Tel.: +49 899 2139-651. *E-mail address:* mohamed-ali.tnani@boschrexroth.de

Abstract

Manufacturing processes have undergone tremendous technological progress in recent decades. To meet the agile philosophy in industry, data-driven algorithms need to handle growing complexity, particularly in Computer Numerical Control machining. To enhance the scalability of machine learning in real-world applications, this paper presents a benchmark dataset for process monitoring of brownfield milling machines based on acceleration data. The data is collected from a real-world production plant using a smart data collection system over a two-years period. In this work, the edge-to-cloud setup is presented followed by an extensive description of the different normal and abnormal processes. An analysis of the dataset highlights the challenges of machine learning in industry caused by the environmental and industrial factors. The new dataset is published with this paper and available at: `https://github.com/boschresearch/CNC_Machining`.

Keywords: Smart manufacturing; Internet of Things; Machine learning; Process monitoring; Data mining; Industry 4.0; Industrial dataset

1. Introduction

Manufacturing equipment is characterized by its long-lasting capability of up to 20 or more years [1] and communicating with brownfield components and controllers can be a barrier to scalable real-time applications. Development towards Industry 4.0, and more specifically Cyber-Physical Systems and Internet of Things (IoT), paved the way for digitalization and retrofitting of old machinery with connected sensors, edge intelligence, cloud connection, etc. [2, 3]. One of the most robust and long-standing pillars of the production chain are Computer Numerical Control (CNC) machines.

Highly automated machining centers are characterized by their high-speed manufacturing but also by their complexity. The extreme environmental conditions and the high-speed processing engender operation failures such as tool breakage, improper tool clamping or chip jamming [4]. The high variety in tool types and tool operations (OP), in terms of shape, geometries, materials, coatings, surface finishing and physical changes over time, rises strong robustness and generalization challenges for traditional analytics [4]. The complexity increases with the discrepancy between the same processes caused by changes in the machining parameters and maintenance methods, such as lubrication of components.

Addressing these challenges, a wide variety of research in tool health monitoring [5, 6] and few in process quality [7, 8] has been performed. To enhance the research in the field, some machining datasets have been published. One of them is the SMART LAB Milling Dataset [9], which has been collected at the University of Michigan over 18 different experiments from direct measurements. The goal of the dataset is to investigate the tool wear detection as well as detection of inadequate clamping. A second dataset is from the NASA Milling Dataset [10], which studies the tool wear based on three different types of sensors, acoustic emission, vibration and current. However, both experiments were conducted in a laboratory during a short limited time frame. To the best of the authors knowledge, there exists no CNC research dataset from a real production environment collected over a long period of time and from different machines. These conditions are essential to build robust data-driven models and improve their generalization and thus their reliability in industry.

This paper introduces a new dataset collected during real-life production. The data is collected from three brownfield milling CNC machines at different time frames in a two years interval. The first section of this paper describes the IoT system built to retrofit the old machinery, ease the data collection and enable parallel prediction and annotation. The second section provides in-depth description and analysis of the dataset that will be published with this paper. It is followed by an overview of the environmental and industrial challenges, which have been considered in a systematic way during dataset creation and annotation. This allows the scientific community to work on solutions for these real-world problems and provide comparable results for benchmarking. More specifically, the dataset has been designed to address the challenges of feature drifts between machines and over time, the high diversity of tool operations during production and the severe dataset imbalance in terms of number of samples per class. To overcome these challenges, we propose some data split scenarios which can be used in future work.

2. Experimental Set Up/ Data Acquisition System

2.1. Hardware components

To keep the research as close as possible to the industrial scenario, the data is collected from different 4-axis horizontal CNC machining centers during production. The machines are processing aluminum workpieces as depicted in Figure 1. For the data acquisition, we used an indirect method by collecting accelerometer data from Bosch CISS sensors [11] mounted to the rear end of the spindle housing. Other approaches opt for mounting the sensors in the machining area [12, 13, 5, 7]. This rear area remains unaffected by extreme machining environment, coolant or material chips and is available for retrofitting new sensors to brownfield machines. The sensor maintains a constant distance to the tool center point and the three axes of the accelerometer are in alignment with the linear motion axis of the machine. The sensor coordinate system is indicated in Figure 1.

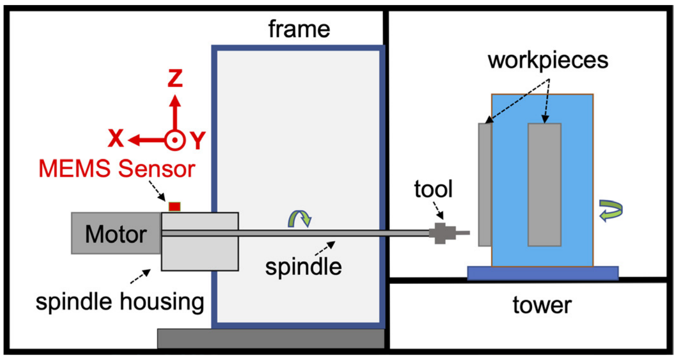


Fig. 1: Schematic sketch of the experimental setup: 4-axis machining center with mounted sensor.

Using the low-cost tri-axial CISS sensor, acceleration data is collected with a sampling rate of 2 kHz. As mentioned in Section 3.2.2, most relevant frequencies to monitor the machining processes are low integer multiples (1..4) of the spindle speed. For tool operations present in this dataset (see Table 1), these frequencies will be in the range of 75 Hz to 1 kHz. According to the Nyquist-Shannon theorem [14], a minimum sampling rate of 2 kHz is sufficient to detect machine anomalies. Sampling with this rate along the 3-axes produces an amount of 4.14 GB per day. Such volumes of data cannot be fully stored and processed in on-premise solutions. It demands a smart data mining system to collect, store, annotate, process and learn from the gathered data.

2.2. Software Architecture for Data Collection

To have reliable annotation, continuous data collection and simultaneous Machine Learning (ML) evaluation, we require an IoT architecture which enables:

1. central aggregation of selected anomalies and processes across different machining centers and locations,
2. local storage and processing of raw sensor data including event annotation by product experts,
3. aggregation of annotated data in a central database,
4. centralized training of ML models, and
5. management and deployment of models and modules from the cloud to the edge device.

Sun et al. [15] proposes the offloading of the ML inferencing to on-premise servers to improve the communication effort and latency. In similiar fashion, Yigitoglu et al. [16] proposed a framework for Fog computing. Motivated by both works [16, 15], the data collection system presented in this work is characterized in an edge-to-cloud architecture. The main goal of this architecture is the simplification of data annotation, the use of expert knowledge in the shop floor, and the centralized storage of annotated data in the cloud. Through an anomaly detector module, potential events and anomalies are pre-selected for annotation. In this section, we outline the edge-to-cloud data collection system.

2.2.1. Edge stack

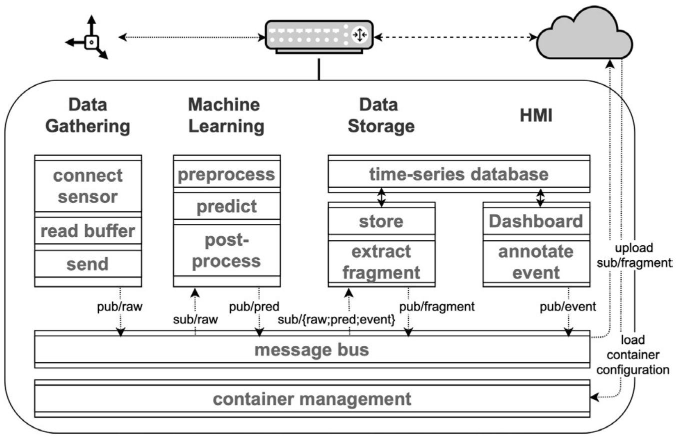


Fig. 2: Concept and interaction of containers in the edge stack.

The edge stack represented in Figure 2 describes the modules running in the production line on site. The modules are managed from the cloud side by an orchestration client running on the edge device. A messaging bus using the Message Queuing Telemetry Transport (MQTT) protocol provides a standardized interface for local inter-application communication. The data gathering and annotation system involves multiple modules. Firstly, a data gathering module establishes a connection to the accelerometer sensor and triggers the read. The data stream is afterwards published on the message bus. Secondly, the data stream is subscribed by a ML module, which with predictions on the stream, supports the quality check process by pre-selecting the correct time frame for anomalies. This allows time-delayed annotations to be entered by the end-of-line quality check, while retaining the majority of data only in the edge time-series database. Ultimately, a dashboard allows the visualization of the ML pseudo-labels and manual annotation via the user interface. Once an event is validated by the experts, the corresponding data segment gets acquired and queried for upload to the cloud. The major benefit of the architecture is the collaboration of data science and domain expertise. It allows additionally in-place distribution of updated ML modules, which support and improve data annotation.

2.2.2. *Cloud Stack*

Publishing large-scale dataset, training ML models centrally or aggregating data from multiple edge devices require a cloud stack. Figure 3 presents the data flow and the main components required in the cloud. Annotated vibration fragments from edge

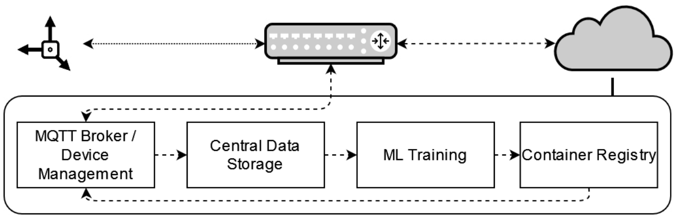


Fig. 3: Services and building blocks in cloud stack.

devices get streamlined to the central data storage. Using the vibration fragments from the central data storage, the data is segmented and preprocessed. Since annotating high frequency data can be very challenging for human experts, verification of correctness of the labels is essential. After verification, the ML model is (re-)trained, gets build and registered in the Container Registry. The edge device management communicates the new model version to the local server. Using this paradigm, we successfully improve our models at the edge through the continuous collection of anomalies.

3. New CNC Machining Dataset

The overall goal of this paper is to enhance the scalability of machine learning in real-world applications by presenting a dataset containing the main challenges that hinder the reliability of ML algorithms in the manufacturing environment. The challenges are caused on the one hand by the variation of material components (spindle, machining tools, raw material produced, etc.) due to wear or discrepancies in the physical structure of parts across machines, and on the other hand by the frequent changes in the production flow as a result of customer requirements and technological progress.

The following section presents the dataset and the various process operations. We introduce how we systematically embed the real-world challenges into the collected data.

3.1. *Data Description*

The data is collected in a production plant from 3 different CNC machines (M01, M02 and M03) on a regular basis during the time interval of October 2018 to August 2021. The time frame is tagged as ”Month_Year” and represents the 6-month interval before the label. For example, ”Aug_2019” would refer to the period between February 2019 and August 2019.

The machine performs a sequence of several operations using different tools on aluminium parts to work the specified design. It is important to mention that the machines produce different parts and the process flow changes over time. To study the drift between machines and over time, the dataset is built with 15 different tool operations that run on all 3 machines at different time frames. Table 1 gives an overview on the characteristics of the different operations.

Table 1 Tools operations collected from M01, M02 and M03.

Tool operation	**Description**	**speed** $[Hz]$	**feed** $[mm\ s^{-1}]$	**duration** $[s]$
OP00	Step Drill	250	≈ 100	≈ 132
OP01	Step Drill	250	≈ 100	≈ 29
OP02	Drill	200	≈ 50	≈ 42
OP03	Step Drill	250	≈ 330	≈ 77
OP04	Step Drill	250	≈ 100	≈ 64
OP05	Step Drill	200	≈ 50	≈ 18
OP06	Step Drill	250	≈ 50	≈ 91
OP07	Step Drill	200	≈ 50	≈ 24
OP08	Step Drill	250	≈ 50	≈ 37
OP09	Straight Flute	250	≈ 50	≈ 102
OP10	Step Drill	250	≈ 50	≈ 45
OP11	Step Drill	250	≈ 50	≈ 59
OP12	Step Drill	250	≈ 50	≈ 46
OP13	T-Slot Cutter	75	≈ 25	≈ 32
OP14	Step Drill	250	≈ 100	≈ 34

For sake of confidentiality the tool operations order has been shuffled and only a part of the production flow is present in the dataset. Each operation in the table represents a specific process performed by a different tool with unique parameters.

As described in the experimental set-up, the data has been collected from the accelerometer with no further information from the machine’s controller. Figure 4 gives an overview on

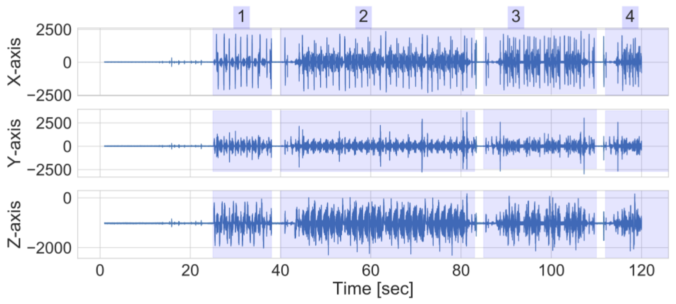


Fig. 4: Overview of the segmentation step of the different tool operations. The X, Y, Z acceleration axes of 4 sequential tool operations are illustrated. The tool number is mentioned in the upper border.

the collected acceleration data from a machining sequence. The data is then manually segmented and structured in the research database. Having no connection to the controller hinders the automated segmentation and thus the process-wise anomaly detection. A non-intrusive solution to monitor and prevent process failures consist of windowing the data stream with a fixed-sized window length and processing the windows steam independently from the process ID.

3.2. Real-world Challenges

Generalization is still one of the primary challenges for industrial ML due the continuous disturbances. Driven by market demand and technical progress, CNC machining production processes are constantly changing with R&D advancement, which goes along with modifications in the tool process operations. Another type of disturbance is caused by the noisy environment in the shop floor and the high imbalance of the normal/abnormal classes. This section presents the different industrial challenges based on the Bosch CNC Machining Dataset described in the previous section.

3.2.1. Environmental challenges

During machining, the different process operations are conducted in high-speed, requiring a frequent mounting and unmounting of tools on the spindle chuck. These factors lead occasionally to process failures mainly caused by tool misalignment, chip clamping, chip in chuck, tool breakage, etc. To reach the optimal product quality, after each batch an expert on the shop floor controls the resulting workpiece in a gauging station and annotate the process health. Nevertheless, labeling during production is still very challenging. Due to the manual drudgery gauging, some processes are wrongly labelled and precise annotations are missing. The published dataset focuses on the quality process failures, i.e., the OK class refers to a healthy process and NOK refers to a faulty process.

A common challenge in industrial datasets is the strong OK/NOK unbalance, especially in process monitoring tasks. Figure 5 shows an unbalance rate of 816:35 between the OK/NOK in our dataset. In our real production, the amount of OK samples are significantly higher. To provide an exemplary dataset, a reasonable number of OK processes were selected

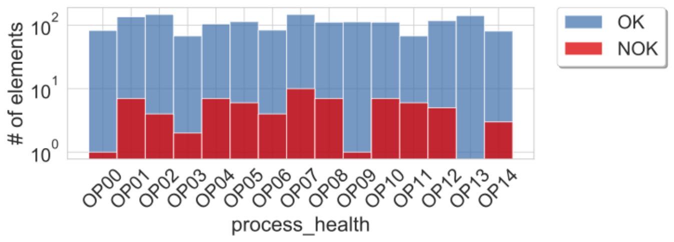


Fig. 5: Class distribution per process operation.

from the different time periods, which reduces the class imbalance.

Besides the process failures, some condition anomalies occur and are detected only after machine maintenance. These anomalies are caused mainly by components wear, hydraulic issues, incorrect settings, etc. However, before reaching a critical phase, a slight deterioration/change over time is seen, causing additional noise in the vibration data. This causes a drift in the OK class between different time frames. In addition to ageing drift, a discrepancy between the conditions of the machines and machine components increases the challenge in real-world applications. The within-class discrepancy over time and between machines is studied in Figure 6.

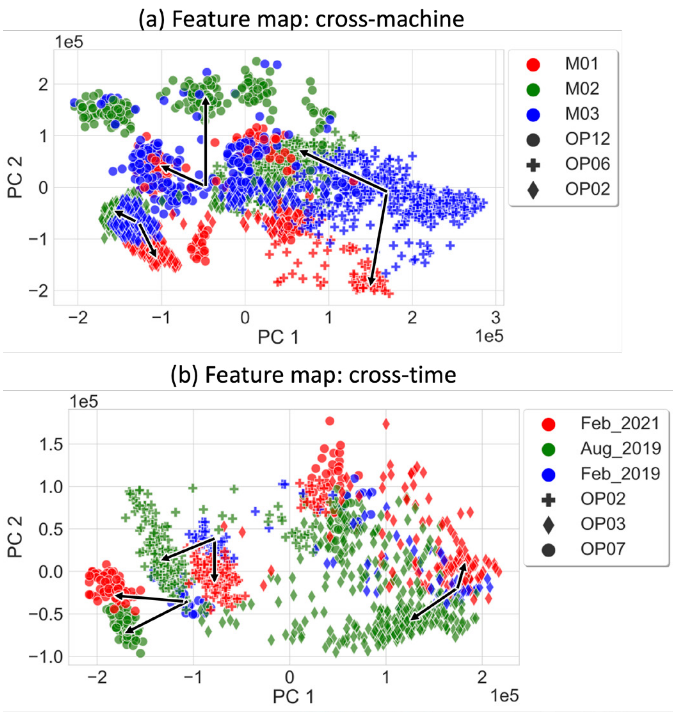


Fig. 6: Example of feature maps of 3 different OPs reduced into 2D using principal component analysis [17]. **(a)** plots the drift between the 3 machines (in Aug_2019). **(b)** plots the drift between the 3 largest time frames (in M01).

Considering the data stream challenges, the raw data from 3 different processes are first snipped using a sliding window with window length equal to 4096. This value has been defined empirically, due to the nature of the collected data and the known process steps. From each window, the most common features for industrial time-series data are extracted using a generic feature extractor Tsfresh [18]. This includes summary statistics, characteristics of samples distribution and observed dynamics. To visualize the discrepancies, the high-dimensional features have been reduced to 2 dimensions using principal component analysis [17]. Figure 6.a visualizes the discrepancy cross-machines in a single time interval (Aug_2019) and highlights the challenge of scaling data-driven algorithms to solve industrial tasks. In a similar manner, in Figure 6.b, the drift of the data over time is depicted for the 3 largest time intervals from a single machine (M01).

To encounter the mentioned challenge, generalization of the ML models must be the main evaluation criteria. By building the training dataset, some processes should be kept aside to evaluate the performance of the models. The dataset published with this paper provides suitable content and structure to enable ML researchers to develop more robust models for such unavoidable environmental challenges from real life.

3.2.2. *Industrial challenges*

To enhance the ML generalization, our dataset presents an example of 15 different tool operations. As mentioned previously, each OP is characterized by a unique parametrization that results in different patterns in the time series signal, making it difficult to predict health status. Using the same pipeline as in Section 3.2.1, the features are extracted from the different OPs and the high-dimensional extracted features are mapped in a two-dimensional space using principle component analysis [17]. An overview of the reduced features is presented in Figure 7.

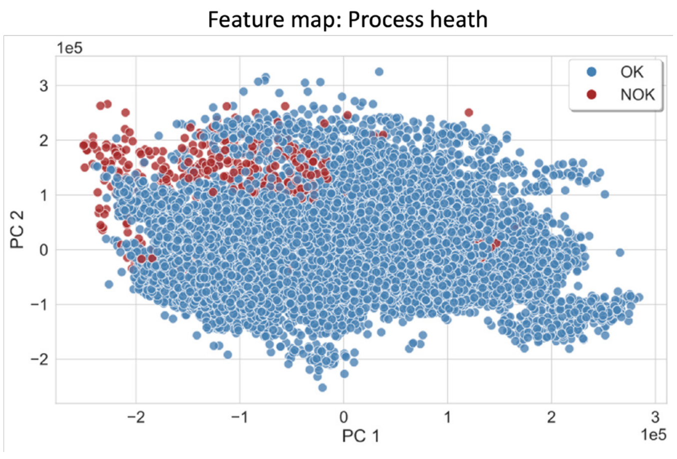


Fig. 7: Feature maps of the complete dataset reduced into 2D using principle component analysis [17].

In Figure 7, the distancing between the OKs and NOKs of the different OPs is illustrated. Some processes of the NOK class are easily distinguishable from the OK class. In others, it is difficult to distinguish between the OK class and the NOK class due to the difference in severity of the anomaly's impact. An example is shown in Figure 8, where a comparison between OP07 and OP08 in time and frequency-domain is conducted. It shows that the impact is more severe in OP07 than in OP08 and a clear divergence between the two processes in both time and frequency domains. However, a common observation is that the anomaly can be detected in frequencies which are integer multiples of the spindle speed. For this example of OP07, the frequency characteristics in the 200 Hz and 400 Hz regions therefore have visibly higher amplitude compared to the healthy process.

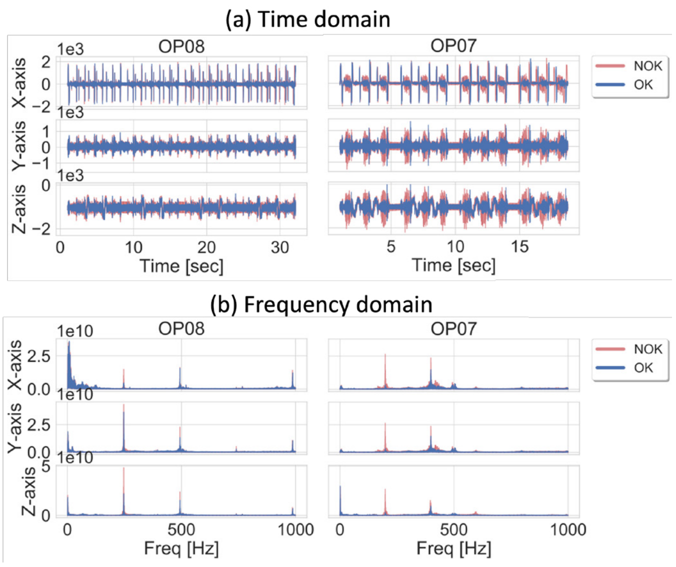


Fig. 8: Comparison of 2 different tool operations: OP07 vs. OP08.

To achieve rapid processing and non-intrusive solutions, time series signals are usually windowed at fixed length (WS). This technique is generally used as a data augmentation technique, especially for NOK data. The drawback of segmenting NOK data is that the label of small segments may not correspond to the complete process. This effect is mainly observed in the first and last extracts, where anomalies are not present yet. When labelling the published data, we truncated the start and end of the OP from the NOK samples. However, this issue can appear in the middle of the process due to fast position change. This can be seen in Figure 9, where a small snippet from the middle of OP08 of the OK and NOK classes matches exactly. To encounter this issue, a reasonable choice of WS needs to be defined.

The CNC Machining dataset provides the needed variety of samples and classes with different levels of discrimination that allow the research community to work on solutions in a systematic way and investigate the robustness of the data-driven methods to industrial challenges.

3.2.3. *Dataset partitioning*

By publishing this dataset, we encourage the research of ML models and learning techniques for noisy time-series data. To realistically measure performance in the real-world challenges, we propose three strategies for partitioning the CNC Machining dataset. With a machine-wise partitioning, as in Figure 10.a, the ability to perform on a new machine outside the training set is addressed. Using time-wise partitioning, as in Figure 10.b,

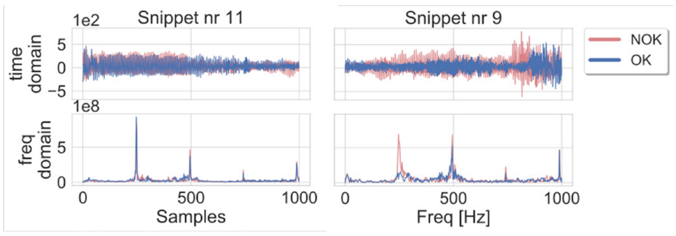


Fig. 9: Data segmentation causing faulty labels. Data taken from "OP08_Feb2019_000" and windowed with $ws = 1000$. For sake of perfect overlap, the OK sample is cropped to the range [4650, 64231].

we address a data drift over time, by withholding some time intervals exclusively for validation and testing.

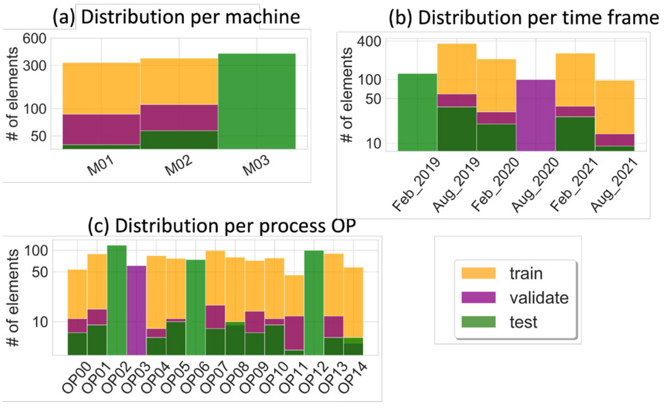


Fig. 10: Three strategies for dataset partitioning.

We intentionally suggest doing this already for the validation set and not only for the test set to be able to check overfitting during training. A third option for partitioning is the application of the same strategy on each process as in Figure 10.c.

4. Conclusion

Generalization is still a major challenge for industrial ML applications. To overcome this limitation, we proposed in this paper a challenging dataset from a real production plant. We depicted our smart data collection system based on an edge-to-cloud IoT architecture. The main benefit of this approach is, firstly, to retrofit brownfield CNC machinery where a direct measurement is extremely complicated, and secondly, enable the data science and domain expertise collaboration. With the presented system, vibration data has been collected from 3 different machines over a long time-interval. The data analysis showed that, with a low-cost accelerometer mounted in the rear side of the machine, process anomalies are detectable. The advantage of this approach is to avoid the extreme conditions from the front side, i.e. the machining area. Finally, to enhance ML and machine monitoring researches, we highlighted the environmental and industrial challenges embedded in the presented dataset. Some dataset scenarios have been proposed to enable the researchers to work on solutions in a systematic way. Future research will focus on development of robust ML architectures. Labeling and segmenting time-series data remain important topics and will be further investigated.

References

[1] Tim Stock and Günther Seliger. Opportunities of sustainable manufacturing in industry 4.0. *Procedia Cirp*, 40:536–541, 2016.
[2] A Quatrano, Simone De, ZB Rivera, and D Guida. Development and implementation of a control system for a retrofitted cnc machine by using arduino. *FME Transactions*, 45(4):565–571, 2017.
[3] Romulo G Lins, Bruno Guerreiro, Robert Schmitt, Jianing Sun, Marcio Corazzim, and Francis R Silva. A novel methodology for retrofitting cnc machines based on the context of industry 4.0. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, 1–6, 2017.
[4] Chandra Nath. Integrated tool condition monitoring systems and their applications: a comprehensive review. *Procedia Manufacturing*, 48:852–863, 2020.
[5] Daniel Frank Hesser and Bernd Markert. Tool wear monitoring of a retrofitted cnc milling machine using artificial neural networks. *Manufacturing letters*, 19:1–4, 2019.
[6] T Mohanraj, S Shankar, R Rajasekar, NR Sakthivel, and Alokesh Pramanik. Tool condition monitoring techniques in milling processa review. *Journal of Materials Research and Technology*, 9(1):1032–1042, 2020.
[7] Zhiyuan Lu, Meiqing Wang, and Wei Dai. Machined surface quality monitoring using a wireless sensory tool holder in the machining process. *Sensors*, 19(8):1847, 2019.
[8] Vinh Nguyen and Shreyes N Melkote. Manufacturing process monitoring and control in industry 4.0. In *Proceedings of 5th International Conference on the Industry 4.0 Model for Advanced Manufacturing*. Springer, 144–155, 2020.
[9] System level Manufacturing and Automation Research Testbed (SMART) at the University of Michigan. Cnc milling dataset. https://www.kaggle.com/shasun/tool-wear-detection-in-cnc-mill, 2018. (Date accessed: 14.12.2021).
[10] A. Agogino and K. Goebel. Best lab, uc berkeley. milling data set, nasa ames prognostics data repository. https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/, 2007. (Date accessed: 14.12.2021).
[11] Bosch Connected Devices and Solutions GmbH. Connected industrial sensor solution. https://www.bosch-connectivity.com/media/downloads/ciss/ciss_datasheet.pdf, 2020. (Date accessed: 14.12.2021).
[12] Yang Hui, Xuesong Mei, Gedong Jiang, Tao Tao, Changyu Pei, and Ziwei Ma. Milling tool wear state recognition by vibration signal using a stacked generalization ensemble model. *Shock and Vibration*, 2019.
[13] Grzegorz Wszołek, Piotr Czop, Jakub Słoniewski, and Halit Dogrusoz. Vibration monitoring of cnc machinery using mems sensors. *Journal of Vibroengineering*, 22(3):735–750, 2020.
[14] Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, jan 1949.
[15] Wen Sun, Jiajia Liu, and Yanlin Yue. Ai-enhanced offloading in edge computing: When machine learning meets industrial iot. *IEEE Network*, 33(5):68–74, 2019.
[16] Emre Yigitoglu, Mohamed Mohamed, Ling Liu, and Heiko Ludwig. Foggy: A framework for continuous automated iot application deployment in fog computing. In *2017 IEEE International Conference on AI & Mobile Services (AIMS)*, 38–45, 2017.
[17] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.
[18] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018.

The patent document that follows paper 1 describes the research case study and provides a supplementary description to the paper. The supplementary information relates firstly to the machining process failure, referred to as "NOK" in the published dataset. During high-speed machining of aluminium materials on CNC machines, anomalies in the process occasionally occur, caused by material chips getting caught between the chuck and the machining tool. The anomaly and the case study are described and explained extensively in the patent document.

Additionally, the patent document provides an overview of the ML module used on the Edge to support data annotation, which was mentioned in paper 1. As the focus of the first paper is on highlighting the challenges of the open source dataset and its analysis, the ML module used to support event annotation and detection has been described in the following publication as a complementary disclosure to paper 1.

My contribution, as before, is the design, implementation and evaluation of the machine learning based data annotation tool and the production of the published document. The data collection project was supervised by Benjamin Menz and Scott Hibbard.

US 20210279573A1

(19) United States
(12) Patent Application Publication (10) Pub. No.: US 2021/0279573 A1
Tnani et al. (43) Pub. Date: Sep. 9, 2021

(54) DETECTING WHEN A PIECE OF MATERIAL IS CAUGHT BETWEEN A CHUCK AND A TOOL

(71) Applicant: **Robert Bosch GmbH**, Stuttgart (DE)

(72) Inventors: **Mohamed-Ali Tnani**, Neu-Ulm (DE); **Benjamin Menz**, Floersbachtal (DE); **Scott Hibbard**, Chicago, IL (US)

(21) Appl. No.: **17/188,757**

(22) Filed: **Mar. 1, 2021**

Related U.S. Application Data

(60) Provisional application No. 62/985,170, filed on Mar. 4, 2020.

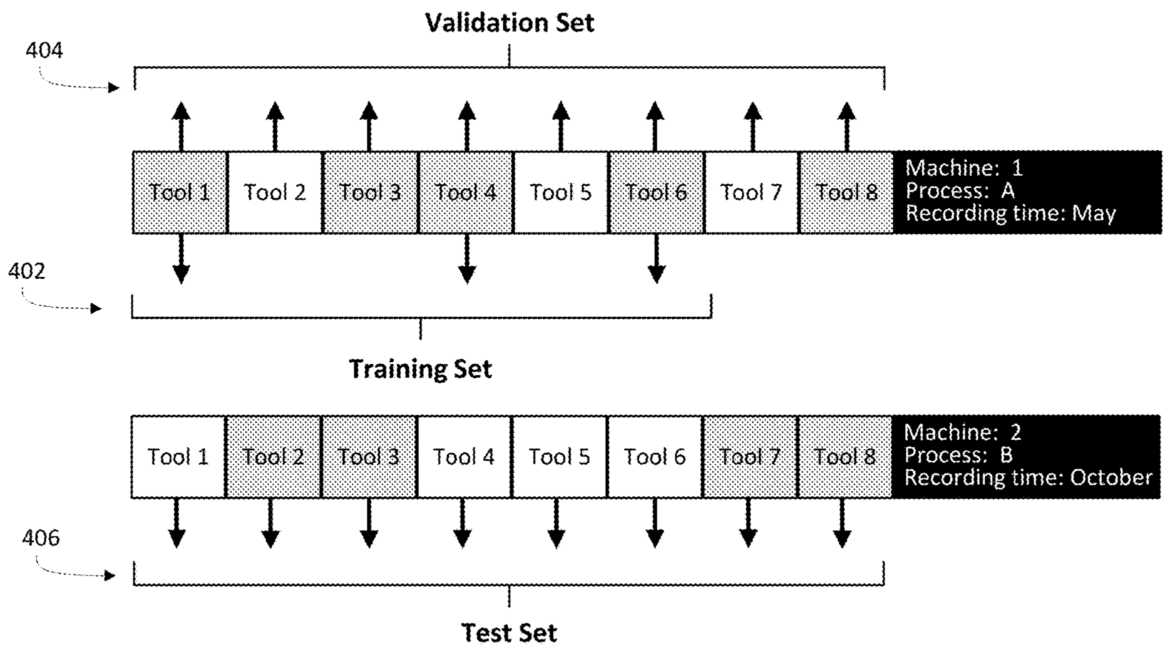
Publication Classification

(51) **Int. Cl.**
G06N 3/08 (2006.01)
G06N 3/04 (2006.01)

(52) **U.S. Cl.**
CPC *G06N 3/08* (2013.01); *G06N 3/04* (2013.01)

(57) **ABSTRACT**

A system for detecting material caught between a chuck and a removable tool. The system includes a sensor mounted on a surface that experiences a vibration caused by a rotating of the removable tool in the chuck. The system also includes an electronic processor configured to receive raw vibration data from the sensor, generate transformed vibration data by transforming the raw vibration data, and using a machine learning model, analyze the raw vibration data and transformed vibration data to determine whether there is a piece of material caught between the removable tool and the chuck.



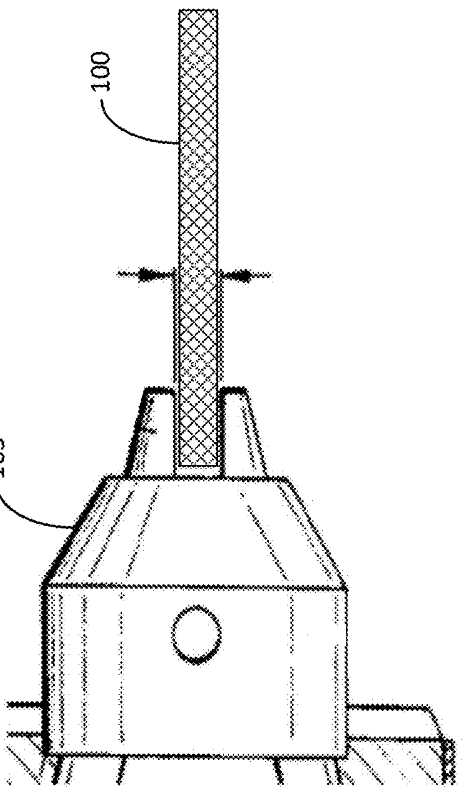


FIG. 1A

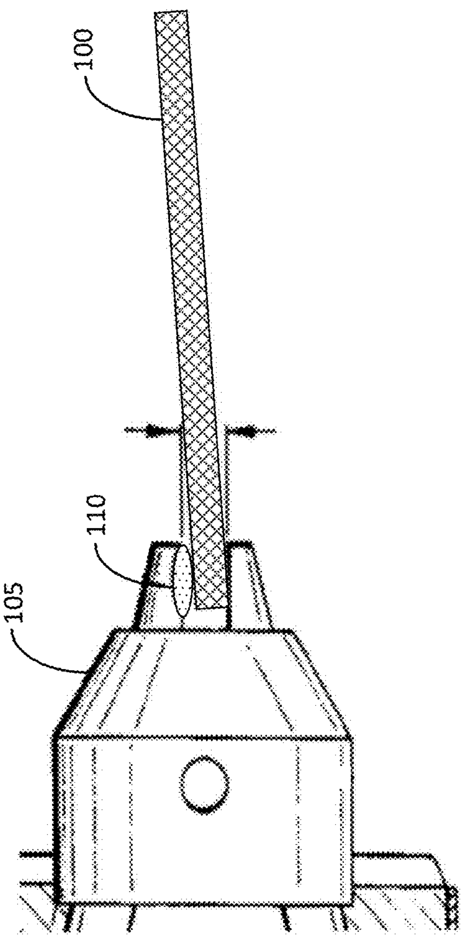
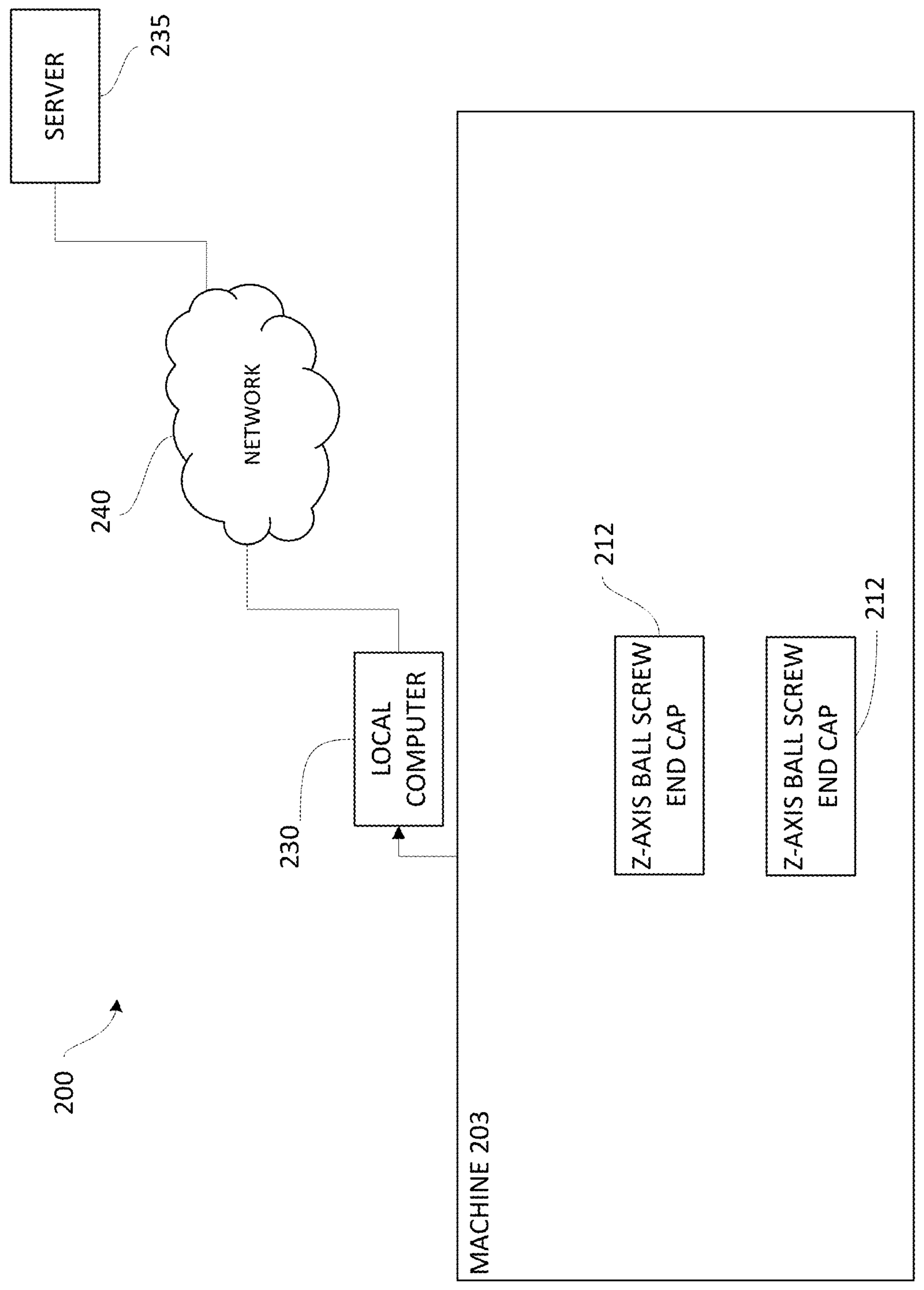


FIG. 1B

FIG. 2


200
SERVER
235
NETWORK
240
LOCAL COMPUTER
230
212
Z-AXIS BALL SCREW END CAP
212
Z-AXIS BALL SCREW END CAP
MACHINE 203

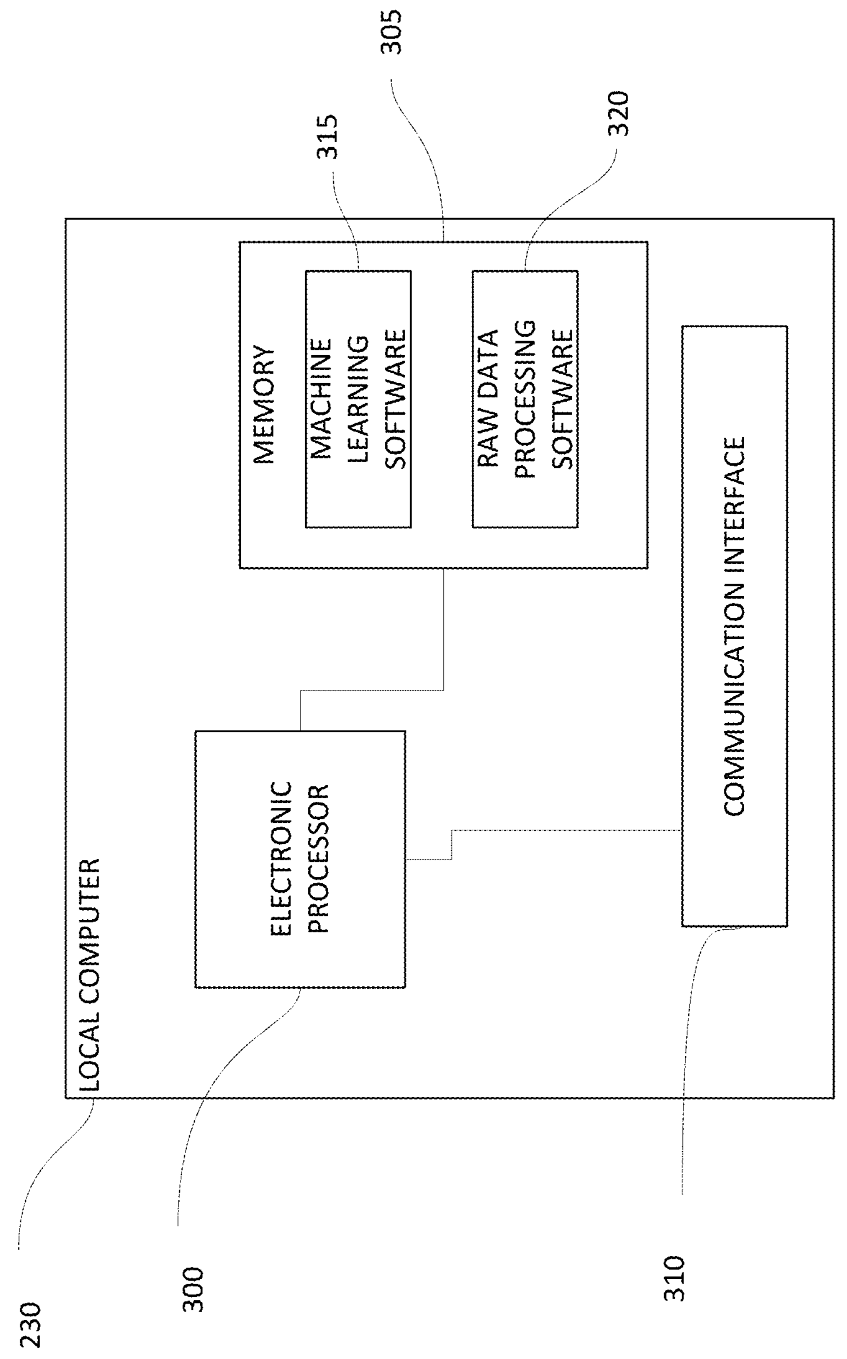

LOCAL COMPUTER
ELECTRONIC PROCESSOR
MEMORY
MACHINE LEARNING SOFTWARE
RAW DATA PROCESSING SOFTWARE
COMMUNICATION INTERFACE
230
300
305
310
315
320

FIG. 3

FIG. 4

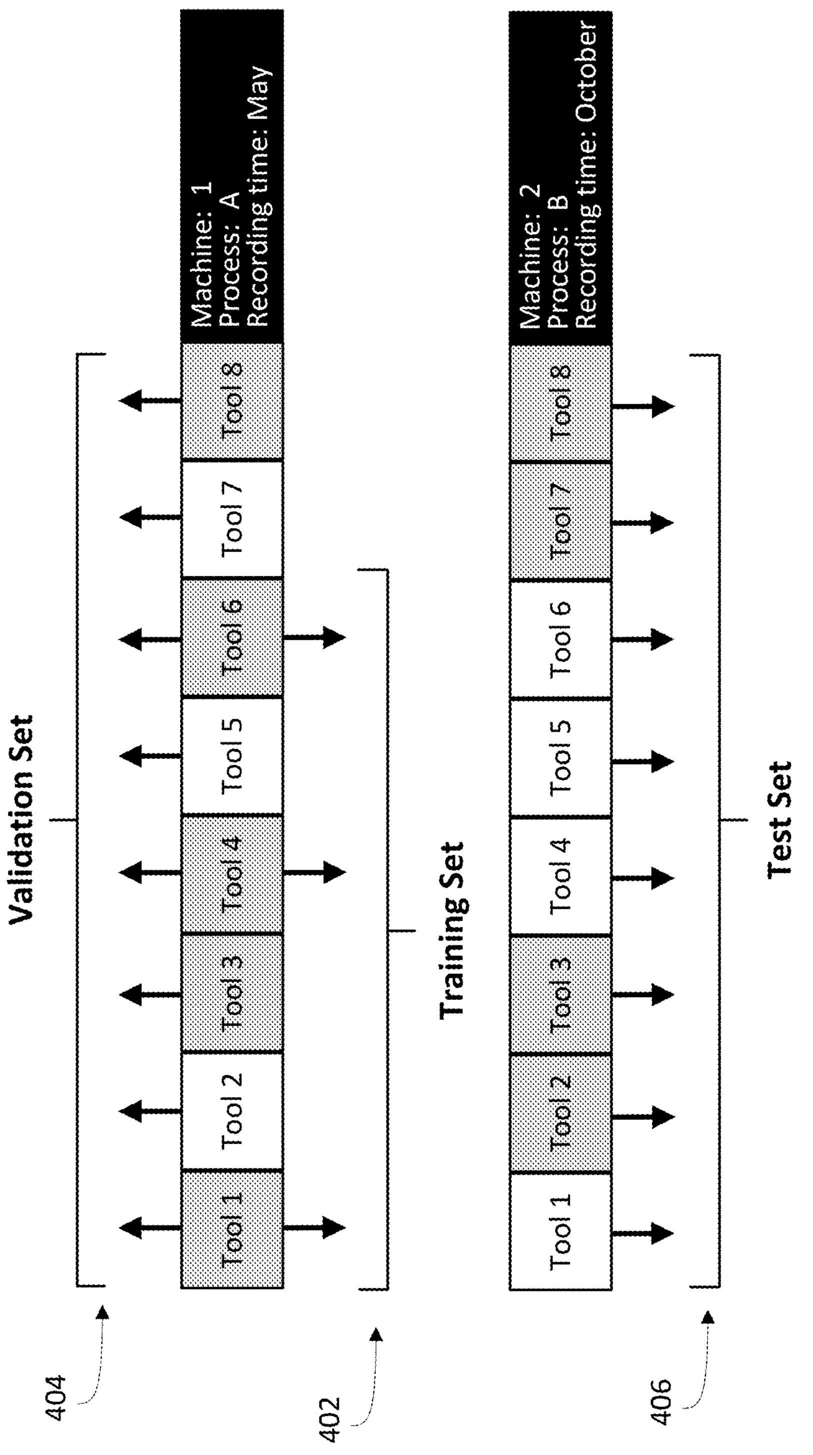
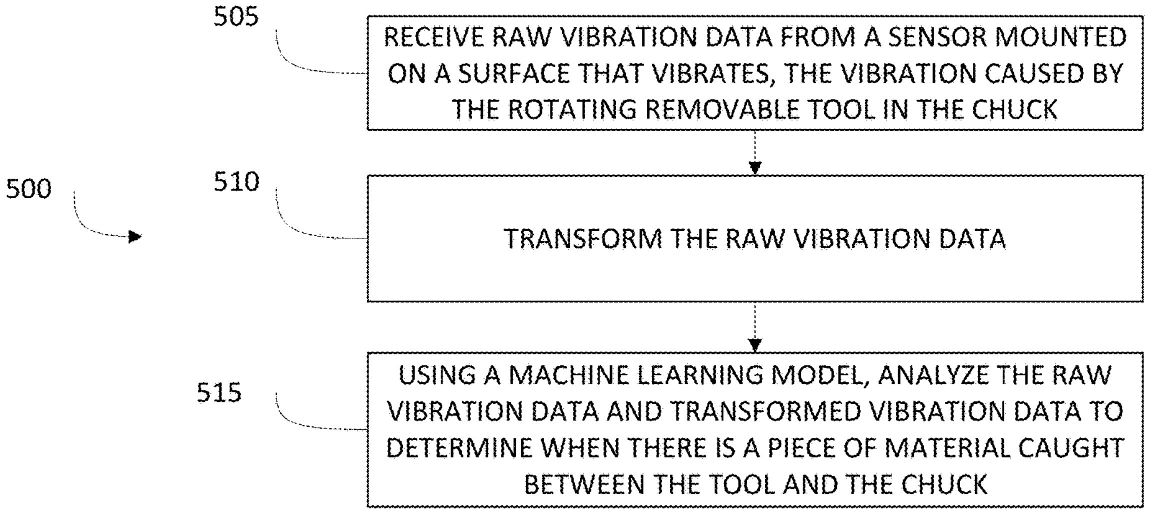

Validation Set
Training Set
Test Set
Machine: 1
Process: A
Recording time: May
Machine: 2
Process: B
Recording time: October
Tool 1
Tool 2
Tool 3
Tool 4
Tool 5
Tool 6
Tool 7
Tool 8
404
402
406

FIG. 5



505
RECEIVE RAW VIBRATION DATA FROM A SENSOR MOUNTED ON A SURFACE THAT VIBRATES, THE VIBRATION CAUSED BY THE ROTATING REMOVABLE TOOL IN THE CHUCK
500
510
TRANSFORM THE RAW VIBRATION DATA
515
USING A MACHINE LEARNING MODEL, ANALYZE THE RAW VIBRATION DATA AND TRANSFORMED VIBRATION DATA TO DETERMINE WHEN THERE IS A PIECE OF MATERIAL CAUGHT BETWEEN THE TOOL AND THE CHUCK

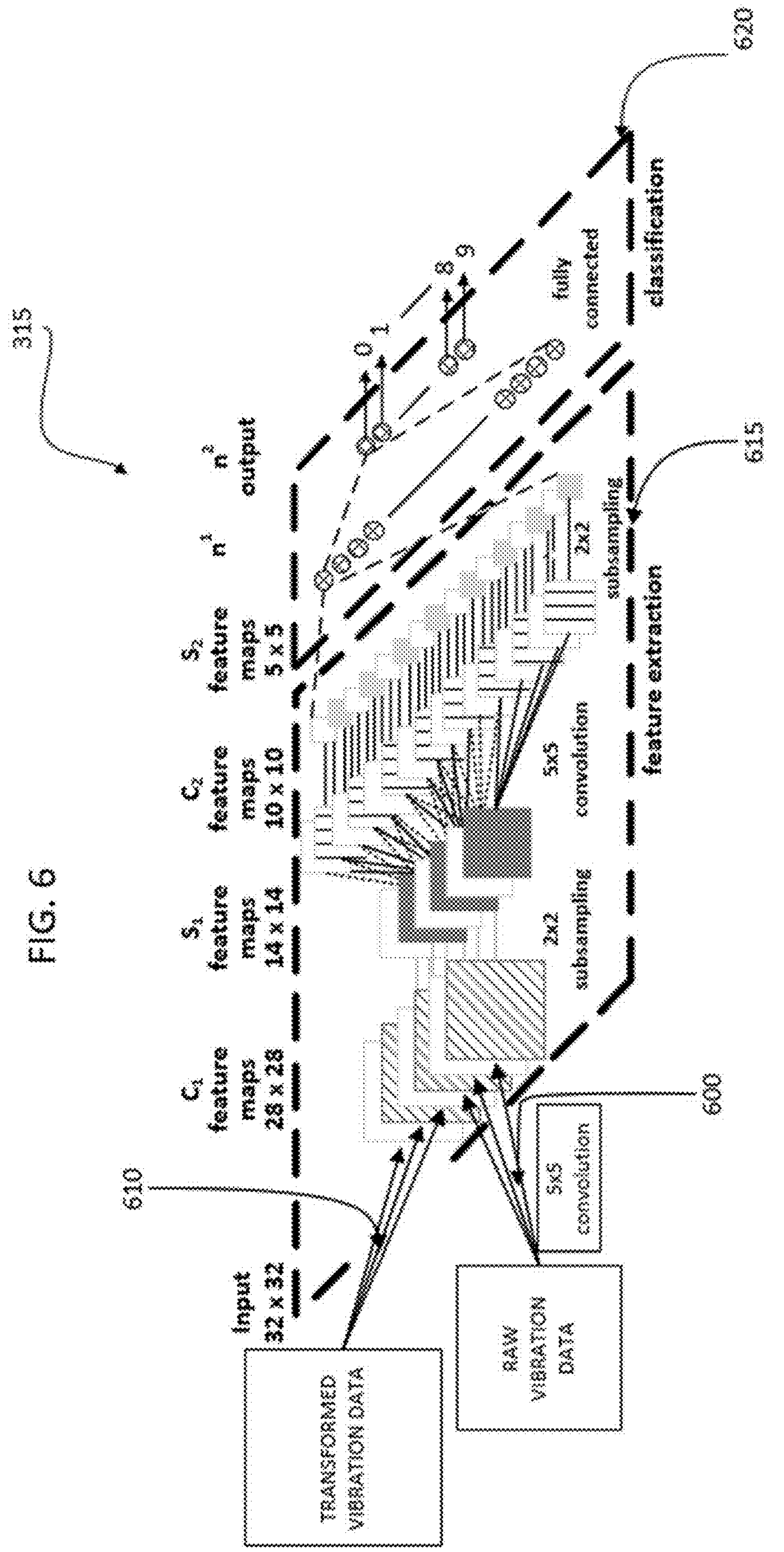
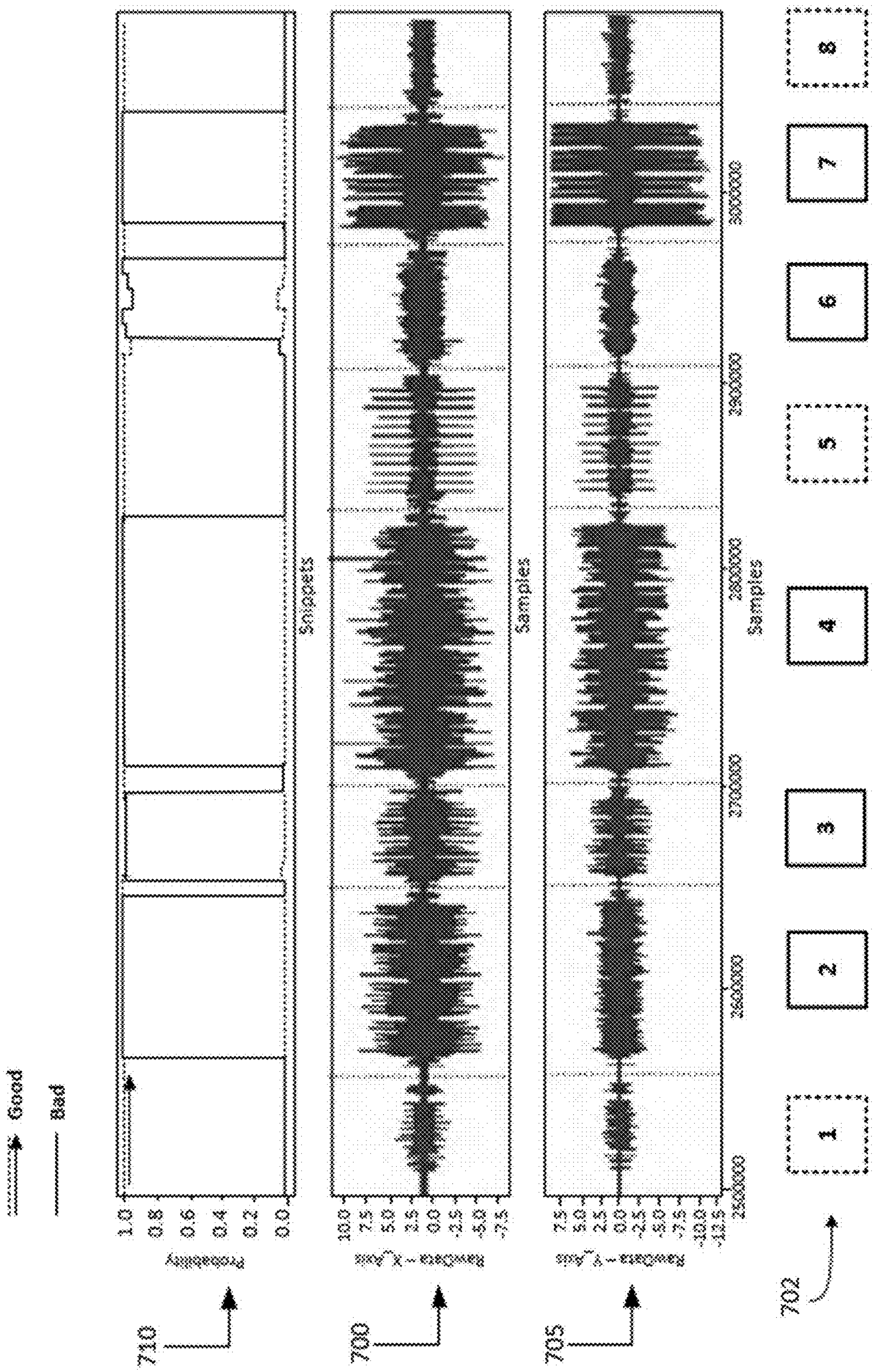

315
Input 32 x 32
C_1 feature maps 28 x 28
S_1 feature maps 14 x 14
C_2 feature maps 10 x 10
S_2 feature maps 5 x 5
n^1
n^2 output
0
1
8
9
fully connected
classification
620
615
2x2 subsampling
5x5 convolution
2x2 subsampling
feature extraction
5x5 convolution
600
610
TRANSFORMED VIBRATION DATA
RAW VIBRATON DATA

FIG. 6

FIG. 7


Good
Bad
710
Probability
1.0
0.8
0.6
0.4
0.2
0.0
Snippets
700
RawData ~ X_Axis
10.0
7.5
5.0
2.5
0.0
-2.5
-5.0
-7.5
Samples
705
RawData ~ Y_Axis
7.5
5.0
2.5
0.0
-2.5
-5.0
-7.5
-10.0
-12.5
2500000
2600000
2700000
2800000
2900000
3000000
Samples
702
1
2
3
4
5
6
7
8

DETECTING WHEN A PIECE OF MATERIAL IS CAUGHT BETWEEN A CHUCK AND A TOOL

RELATED APPLICATIONS

[0001] This application is related to and claims the benefit of and priority to U.S. Provisional Patent Application Ser. No. 62/985,170, filed Mar. 4, 2020, titled "DETECTING WHEN A PIECE OF MATERIAL IS CAUGHT BETWEEN A CHUCK AND A TOOL" (Attorney Docket No. 022896-3233-US01), the disclosure of which is hereby incorporated herein by reference as if set forth in its entirety.

BACKGROUND

[0002] Machining equipment, such as milling machines, often include a spindle chuck into which different tools can be inserted. Equipped with these tools milling machines can be used to form objects, for example, machine parts.

SUMMARY

[0003] FIG. **1**A shows a tool **100** (for example, a cutting tool) in a chuck **105**. In some embodiments, the chuck is connected to a spindle that is turned by a motor (for example, as part of a computer numerical control (CNC) machine). In FIG. **1**A, the tool **100** is aligned with the chuck **105**. As illustrated in FIG. **1**B, sometimes during the manufacturing process, a small piece of material **110**, such as a metal chip, becomes stuck between the tool **100** and the chuck **105**. For example, a metal chip may become lodged between the chuck **105** and tool **100** when the tool **100** in the chuck **105** is changed either manually or automatically. The lodged chip may change the angle of the tool **100** in the chuck **105** as illustrated in FIG. **1**B. Thus, the piece of material may cause the tool **100** to become misaligned with the chuck **105** and, as a result, make inaccurate cuts, tap inaccurate holes, or both. In some cases, the misalignment of the tool **100** in the chuck **105** may cause damage to the tool **100**, the spindle, the chuck **105**, other parts of the machine operating the tool, or a combination of the foregoing. It should be noted that the tool **100**, although not illustrated herein, may include both a tool and a tool holder.

[0004] Existing systems use sensors mounted (via, for example, a bearing) on the spindle connecting the chuck to the motor and limit-based monitoring to determine when a piece of material is caught between the chuck and the tool. However, these existing systems are not easily used on older or legacy machines, and the hardware used to attach the sensor to the spindle often fails. Additionally, these existing systems suffer from limited scalability and required defined measuring cycles, which cause downtime. Some existing systems require connection to a machine control system to determine when monitoring should take place (when is a tool in use), which tool is in use, or both. Some existing systems use different models for determining whether material is caught between a tool and a chuck depending on the tool in use and need to determine which tool is in use in order to select the correct model. Connecting to the machine control system is complex and requires customization for the different hardware of each machine control system vendor and each machine setup.

[0005] Therefore, embodiments herein describe, among other things, a system and method for detecting when a piece of material is caught between a chuck and a tool. Certain embodiments described herein utilize machine learning software to determine when a piece of material is caught between the chuck and the tool based on vibration data from a sensor mounted on a surface of the machine (for example, a motor housing). Certain embodiments described herein do not require a sensor to be mounted on the spindle and overcome many of the aforementioned deficiencies of existing systems. Additionally, the embodiments described herein do not require connection to machine control systems because they use vibration data to determine when a tool is in use and a machine learning model to determine, for a variety of different tools, whether a piece of material is caught between a tool and a chuck.

[0006] For example, one embodiment provides a system for detecting material caught between a chuck and a removable tool. The system includes a sensor mounted on a surface that vibrates. The vibration of the surface is caused by a rotating of the removable tool in the chuck. The system also includes an electronic processor configured to receive raw vibration data from the sensor, generate transformed vibration data by transforming the raw vibration data, and using a machine learning model, analyze the raw vibration data and transformed vibration data to determine whether there is a piece of material caught between the tool and the chuck.

[0007] Another embodiment provides a method for detecting material caught between a chuck and a tool. The method includes receiving raw vibration data from a sensor mounted on a surface that vibrates. The vibration of the surface is caused by a rotating of the removable tool in the chuck, generating transformed vibration data by transforming the raw vibration data, and using a machine learning model, analyzing the raw vibration data and transformed vibration data to determine whether there is a piece of material caught between the tool and the chuck.

[0008] Yet another embodiment provides a method for detecting material caught between a chuck and a removable tool. The method includes receiving raw vibration data from a sensor mounted on a surface that vibrates. The vibration of the surface is caused by an operation of the removable tool in the chuck and using a machine learning model, analyzing raw vibration data, transformed vibration data generated from the raw vibration data, or both to determine whether there is a piece of material caught between the removable tool and the chuck.

[0009] Other aspects, features, and embodiments will become apparent by consideration of the detailed description and accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. **1**A illustrates an example of a tool in a chuck when there is no piece of material caught between the tool and the chuck.

[0011] FIG. **1**B illustrates an example of a tool in a chuck when there is a piece of material caught between the tool and the chuck.

[0012] FIG. **2** is a block diagram of a system for detecting when a piece of material is caught between a chuck and a tool according to one embodiment.

[0013] FIG. **3** is a block diagram of a local computer of the system of FIG. **1** according to one embodiment.

[0014] FIG. **4** is an example illustration of training data used to train the machine learning model utilized during the execution of the method of FIG. **5** according to one embodiment.

[0015] FIG. **5** is a flowchart of a method of using the system of FIG. **2** to detect when a piece of material is caught between a chuck and a tool according to one embodiment.

[0016] FIG. **6** is a block diagram of a neural network used to perform the method of FIG. **5** according to one embodiment.

[0017] FIG. **7** is an illustration of raw vibration data and resulting predictions according to one embodiment.

DETAILED DESCRIPTION

[0018] Before any embodiments are explained in detail, it is to be understood that this disclosure is not intended to be limited in its application to the details of construction and the arrangement of components set forth in the following description or illustrated in the following drawings. Embodiments are capable of other configurations and of being practiced or of being carried out in various ways.

[0019] A plurality of hardware devices and software, as well as a plurality of different structural components may be used to implement various embodiments. In addition, embodiments may include hardware, software, and electronic components or modules that, for purposes of discussion, may be illustrated and described as if the majority of the components were implemented solely in hardware. However, one of ordinary skill in the art, and based on a reading of this detailed description, would recognize that, in at least one embodiment, the electronic based aspects of the invention may be implemented in software (for example, stored on non-transitory computer-readable medium) executable by one or more processors. For example, "control units" and "controllers" described in the specification can include one or more electronic processors, one or more memory modules including non-transitory computer-readable medium, one or more communication interfaces, one or more application specific integrated circuits (ASICs), and various connections (for example, a system bus) connecting the various components. It should also be understood that although certain drawings illustrate hardware and software located within particular devices, these depictions are for illustrative purposes only. In some embodiments, the illustrated components may be combined or divided into separate software, firmware and/or hardware. For example, instead of being located within and performed by a single electronic processor, logic and processing may be distributed among multiple electronic processors. Regardless of how they are combined or divided, hardware and software components may be located on the same computing device or may be distributed among different computing devices connected by one or more networks or other suitable communication links.

[0020] FIG. **2** illustrates a system **200** for detecting when a piece of material is caught between a chuck and a tool. In the example provided, the system **200** includes a machine **203** with a motor/motor housing **205**, a z-axis ball screw **210**, a z-axis ball screw end cap **212**, a chuck **215**, a tool **220**, and a sensor **225**. The system **200** also includes a local computer **230** (an edge device) and a server **235**. In some embodiments, the machine **203** is a computer numerical control (CNC) machine. In some embodiments, the local computer **230** is included in the machine **203**. The motor **205** included in the machine **203** drives the z-axis ball screw **210** and the z-axis ball screw **210** translates the rotary motion of the motor **205** to linear motion to change the z-axis. The chuck **215** connects the z-axis ball screw **210** to the tool **220** so that when the motor **205** turns, the tool **220** moves. The chuck **215** allows different tools to be connected to the z-axis ball screw **210**. The tool **220** attached to the chuck **215** may be changed automatically or manually. In one example, the tool **220** is a turning tool capable of forming objects such as mechanical components out of a piece of material by cutting away at the material. When the tool **220** cuts the material, chips are generated which can get stuck between the tool **220** and the chuck **215** causing misalignment of the tool **220**, such as the misalignment shown in FIG. **1**B.

[0021] The sensor **225** is a vibration sensor, capable of measuring the vibrations generated by the moving tool **220**. For example, the sensor **225** may be a Structure-Borne Sound Sensor (SB SS) or a Connected Industrial Sensor Solution (CISS) manufactured by Robert Bosch LLC. The sensor **225** is communicatively connected to the local computer **230** and sends vibration data to the local computer **230** via various wired or wireless connections. For example, in some embodiments, the sensor **225** is directly coupled via a dedicated wire to the local computer **230**. In other embodiments, the sensor **225** is communicatively coupled to the local computer **230** via a shared communication link such as a Bluetooth™ or other wireless connection. In some embodiments, the sensor **225** is mounted to the motor housing **205**. In other embodiments, the sensor **225** is mounted to the z-axis ball screw end cap **212**, an x-axis ball screw end cap (not shown), or a y-axis ball screw end cap (not shown).

[0022] The local computer **230** and the server **235** communicate over one or more wired or wireless communication networks **240**. Portions of the wireless communication networks **240** may be implemented using a wide area network, such as the Internet, a local area network, such as a Wi-Fi network, short-range wireless networks, such as a Bluetooth™ network, near field communication connections, and combinations or derivatives thereof. In alternative embodiments, the server **235** is part of a cloud-based system external to the system **200** and accessible by the local computer **230** over one or more additional networks.

[0023] It should be noted that while certain functionality described herein as being performed by one component of the system **200**, in some embodiments that functionality may be performed by a different component of the system **200** or a combination of components of the system **200**. It should be understood that the system **200** may include a different number of machines (for example, milling machines) each with a sensor, a different number of local computers, and a different number of servers than the single machine **203**, local computer **230**, and server **235** illustrated in FIG. **2**.

[0024] FIG. **3** is a block diagram of one example embodiment of the local computer **230** of the system **200** of FIG. **2**. The local computer **230** includes, among other things, an electronic processor **300** (such as a programmable electronic microprocessor, microcontroller, or similar device), a memory **305** (for example, non-transitory, machine readable memory), and a communication interface **310**. The electronic processor **300** is communicatively connected to the memory **305** and the communication interface **310**. The electronic processor **300**, in coordination with the memory **305** and the communication interface **310**, is configured to implement, among other things, the methods described herein.

[0025] The memory **305** includes software that, when executed by the electronic processor **300**, causes the electronic processor **300** to perform the method **500** illustrated in FIG. **5**. For example, the memory **305** illustrated in FIG. **3** includes machine learning model **315** and raw data processing software **320**. The machine learning model **315** may be a deep neural network (for example, a convolutional neural network (CNN) or a recurrent neural network (RNN)). In one example, the neural network includes two input channels, allowing the neural network to analyze both raw vibration data and vibration data transformed by the raw data processing software **320** simultaneously to detect when a piece of material is caught between a chuck and a tool. As described below, in some embodiments, the neural network may include a different number of channels than two channels illustrated and described herein.

[0026] In some embodiments, the machine learning model **315** is trained to detect when a piece of material is caught between a chuck and a tool using training data including samples or snippets of vibration data that have been labeled to indicate whether or not they are indicative of a piece of material being caught between a tool and a chuck. The training data includes a training set, a validation set, and a test set. The training set is a set of vibration data samples or snippets used to train the machine learning model **315** (for example, to determine weights and biases in the machine learning model **315**). The validation set is a set of vibration data samples or snippets used to evaluate the machine learning model **315** after each training epoch and test the loss and accuracy of the machine learning model **315** on unseen data. The test set is a set of vibration data samples or snippets used to provide an unbiased evaluation of the final machine learning model **315** and define the degree of generalization of the machine learning model **315**. The training data includes data vibration data from a variety of different machines, using a variety of tools in a variety of states of wear while manufacturing a variety of different objects. The training data may include raw vibration data and transformed vibration data. FIG. **4** provides a visual representation of one example of training data. As illustrated in FIG. **4**, the training set **402**, validation set **404**, and test set **406** may each be different sets of vibration data collected from different machines, using different tools of different ages to manufacture different objects. The vibration data samples for tools illustrated in FIG. **4** having a darker shade represent tools which possess both good process measurements (e.g., vibrations generated by the machine operating without material caught in the chuck) and defect process measurements (e.g., vibrations generated by the machine operating with material is caught in the chuck). The process measurements of the tools that are shared by the train and the validation dataset are split between both datasets. Using such varied training data produces a model trained to detect material caught between a chuck and a tool across a variety of conditions and circumstances.

[0027] In some embodiments, the training data is segmented into snippets allowing the number of training samples having a standard length to be increased. Dataset segmentation consists of slicing a signal into smaller segments (i.e., snippets), which allow enlargement of the training population with samples having a standard length. Each snippet has a window size. For vibration data, drive motor speed and sensor sampling rate should be considered when determining the window size. In some embodiments, a window size is set to cover at least one full revolution of the motor (i.e., the selected window should contain the periodical spatial position of the drive motor). In some embodiments, the window size is determined according to the following formula:

$$\text{Window } Size_{min} = \frac{\text{Sampling Frequency}}{\text{Rotational } Speed_{max}} \tag{1}$$

In some embodiments, each snippet is smaller (for example, half of the window size). In some embodiments, the training data is downsampled. Downsampling is utilized to increase the performance of some neural networks (for example, Long Short-Term Networks) by using smaller window sizes. In some embodiments, the training data is normalized using, for example, Standard-Scaling.

[0028] In some embodiments, the training data is collected via one or more local computers such as the local computer **230** and sent to the server **235**. The server **235** uses the received training data to train a machine learning model **315** and, when the machine learning model **315** is trained, sends the machine learning model **315** to each local computer in the system **200**. When the local computer **230**, executing the machine learning model **315**, cannot determine whether vibration data is indicative of a piece of material being caught between a tool and a chuck, the local computer **230** may send a notification to the server **235** (for example, using a suitable network message or an application programming interface). In some embodiments, the notification may include the vibration data and a label that the machine operator has associated with the vibration data. In response to receiving the notification, the server **235** may retrain the machine learning model **315** and send the retrained machine learning model to each of the local computers in the system **200**. Therefore, the machine learning model deployed to each local computer improves over time from collective awareness and the initial training time needed to apply the machine learning model to monitoring a new machine is reduced.

[0029] Although not illustrated herein, the server **235** may contain components similar to those illustrated in FIG. **3** as being included in the local computer **230**. The functionality described herein as being performed by the local computer **230** or the server **235** may be distributed amongst a plurality of local computers and servers. Additionally, the local computer **230**, the server **235**, or both may contain submodules that include additional electronic processors, memory, or application specific integrated circuits (ASICs) for handling communication functions, processing of signals, and application of the methods listed below. In other embodiments, the local computer **230**, server **235**, or both include additional, fewer, or different components than those illustrated in FIG. **3**.

[0030] FIG. **5** illustrates an example method **500** of detecting when a piece of material is caught between a chuck and a tool. At step **505**, the electronic processor **300** receives raw vibration data from a sensor mounted on a surface of the machine **203** that experiences a vibration. In some embodiments, the vibration is caused by a removable tool (for example, the tool **220**) rotating in the chuck **215**. For example, the sensor **225** may capture the movement of the motor housing **205** or z-axis ball screw end cap **212** in the x-direction, the y-direction, or both. In some embodiments, at step **510**, the electronic processor **300** transforms the raw vibration data to produce transformed vibration data. For example, at step **510**, the electronic processor **300** may execute the raw data processing software **320** to apply a Fast Fourier Transform to the raw vibration data received from the sensor **225**. Applying a Fast Fourier Transform to the raw vibration data can reduce the dimensions of the input space by extracting the power spectral density series. At step **515**, the electronic processor **300**, using a machine learning model (for example, the machine learning model **315**), analyzes the raw vibration data and transformed vibration data to determine when there is a piece of material caught between the tool **220** and the chuck **215**. In some embodiments, as explained in detail with regard to FIGS. **6** and **7**, the machine learning model **315** uses a neural network to predict whether the raw vibration data and transformed vibration data are indicative of vibrations caused by a tool and chuck operating with a piece of material caught between them. FIG. **6** illustrates one example of how the electronic processor **300** (at step **515**) determines when a piece of material is caught between the chuck **220** and the tool **220**. In the example illustrated in FIG. **6**, the machine learning model **315** is illustrated as a convolutional neural network with two input channels. In the example illustrated in FIG. **6**, raw vibration data in the x-direction is fed to the neural network as a signal via a first channel **600** and transformed vibration data in the x-direction is fed to the neural network as a signal via a second channel **610**.

[0031] The neural network has a plurality of layers including feature extraction layers **615** and a classification layer **620**. There are two types of feature extraction layers **615**—convolutional layers and pooling or sub-sampling layers. Each convolutional layer applies filters to the raw and transformed vibration data in the x-direction. In certain embodiments, a filter is a matrix of weight values. The weight values of the filters are set by training the neural network. Sub-sampling layers reduce the size of the input data or signals being processed by the neural network. A sub-sampling layer creates a smaller portion from a larger signal by creating the smaller signal with patterns that represent groups of patterns in the larger signal. The classification layer **620** is responsible for using the extracted features of the raw and transformed vibration data in the x-direction detecting when a piece of material is caught between a chuck and a tool.

[0032] It should be understood that the machine learning model **315** may receive different input via the two input channels than the raw and transformed vibration data in the x-direction illustrated in FIG. **6**. For example, the machine learning model **315** may receive raw and transformed vibration data in the y-direction, raw vibration data in the x-direction and transformed vibration data in the y-direction, or raw vibration data in the y-direction and transformed vibration data in the x-direction. It should be understood that different combinations of vibration data, other than those described herein, may be received by the machine learning model **315** via two input channels. It should also be understood that the machine learning model **315** may be a neural network with a different number of channels than the two channels illustrated in FIG. **6**. For example, the machine learning model **315** may be a neural network with a single input channel and the neural network may receive raw vibration data in the x-direction, raw vibration data in the y-direction, transformed vibration data in the y-direction, or transformed vibration data in the x-direction via the single input channel. In another example, the machine learning model **315** may be a neural network with four input channels and receive raw vibration data in the x-direction, raw vibration data in the y-direction, transformed vibration data in the y-direction, and transformed vibration data in the x-direction via the four input channels. It should be understood that the machine learning model **315** may be a neural network with a different number of channels than those described in the examples presented herein. Additionally, the machine learning model **315** may receive different input via the input channels than the inputs described in the examples presented herein.

[0033] FIG. **7** is an example illustration of raw vibration data in the x-direction **700** for tools **702** (labeled **1** through **8**), raw vibration data in the y-direction **705** for tools **702** (labeled **1** through **8**), and a prediction **710** (made by the machine learning model **315** using raw vibration data in the x-direction **700** and raw vibration data in the y-direction **705**) as to whether a piece of material is caught between a chuck and a tool. In the example illustrated in FIG. **7**, tools with numbers outlined in dashes (in FIG. **7**, tools **1**, **5**, and **8**) are determined to be rotating without a piece of material caught between the chuck and the tool and tools with numbers outlined in a solid line (in FIG. **7**, tools **2**, **3**, **4**, **6**, and **7**) are determined to be rotating with a piece of material caught between the chuck and the tool.

[0034] In some embodiments, when a piece of material is determined to be caught between the chuck **215** and the tool **220**, the electronic processor **300** is configured to send a signal to interrupt the machining process (for example, using a suitable message protocol or discrete signal), send a signal to cause a notification indicating that there is a piece of material caught between the chuck **215** and the tool **220** to a user (for example, a technician), a combination of the foregoing, and the like. For example, the user may be notified of the existence of the piece of the material via a user interface of a user device or the local computer **230**. In some embodiments, interrupting the machining process includes preventing the machine **203** from manufacturing any further objects until a human operator approves the machine **203** for further manufacturing.

[0035] Embodiments described herein are described in terms of detecting a piece of material caught between a chuck and a tool during a rotation of the tool by the chuck and a spindle. However, it should be understood that the embodiments may be used to detect piece(s) of material caught between a chuck, clamp (for example, a blade clamp), or other tool holder and a tool held by the chuck, clamp, or holder during non-rotational movements of a tool by a machine. In one non-limiting example, a tool (for example, a saw blade) used in a reciprocating motion may generate vibrations during the operation of the tool that can be used to determine whether material is caught in the tool holder (for example, a blade clamp). Systems and methods described herein are also applicable to machines operating such tools.

[0036] In the foregoing specification, specific embodiments and examples have been described. However, one of ordinary skill in the art appreciates that various modifications and changes can be made without departing from the scope of the invention as set forth in the claims below. Accordingly, the specification and figures are to be regarded in an illustrative rather than a restrictive sense, and all such modifications are intended to be included within the scope of present teachings.

[0037] In this document, relational terms such as first and second, top and bottom, and the like may be used solely to distinguish one entity or action from another entity or action without necessarily requiring or implying any actual such relationship or order between such entities or actions. The terms "comprises," "comprising," "has," "having," "includes," "including," "contains," "containing" or any other variation thereof, are intended to cover a non-exclusive inclusion, such that a process, method, article, or apparatus that comprises, has, includes, contains a list of elements does not include only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus. An element proceeded by "comprises . . . a," "has . . . a," "includes . . . a," or "contains . . . a" does not, without more constraints, preclude the existence of additional identical elements in the process, method, article, or apparatus that comprises, has, includes, contains the element. The terms "a" and "an" are defined as one or more unless explicitly stated otherwise herein. The terms "substantially," "essentially," "approximately," "about" or any other version thereof, are defined as being close to as understood by one of ordinary skill in the art, and in one non-limiting embodiment the term is defined to be within 10%, in another embodiment within 5%, in another embodiment within 1% and in another embodiment within 0.5%. The term "coupled" as used herein is defined as connected, although not necessarily directly and not necessarily mechanically. A device or structure that is "configured" in a certain way is configured in at least that way but may also be configured in ways that are not listed.

[0038] Various features, advantages, and embodiments are set forth in the following claims.

What is claimed is:

1. A system for detecting material caught between a chuck and a removable tool, the system comprising:
- a sensor mounted on a surface that experiences a vibration caused by a rotating of the removable tool in the chuck; and
- an electronic processor, the electronic processor configured to
  - receive raw vibration data from the sensor;
  - generate transformed vibration data by transforming the raw vibration data; and
  - using a machine learning model, analyze the raw vibration data and transformed vibration data to determine whether there is a piece of material caught between the removable tool and the chuck.

2. The system according to claim **1**, wherein the machine learning model includes a convolutional neural network.

3. The system according to claim **2**, wherein the convolutional neural network includes a first channel whereby the convolutional neural network receives the raw vibration data for analysis and a second channel whereby the convolutional neural network receives the transformed vibration data for analysis.

4. The system according to claim **1**, wherein the sensor is a vibration sensor.

5. The system according to claim **1**, wherein the electronic processor is further configured to send a first signal to interrupt a machining process send a second signal to cause a notification indicating that the piece of material is caught between the chuck and the removable tool to be sent to a user, or both.

6. The system according to claim **1**, wherein the electronic processor is included in a local computer and the electronic processor is configured to receive the machine learning model from a server.

7. The system according to claim **6**, wherein the server is configured to train the machine learning model using vibration data collected from one or more different machines, using one or more different tools of one or more different ages to manufacture one or more different objects.

8. The system according to claim **1**, wherein the electronic processor is configured to transform the raw vibration data by applying a Fast Fourier Transform to the raw vibration data.

9. A method for detecting material caught between a chuck and a removable tool, the method comprising:
- receiving raw vibration data from a sensor mounted on a surface that experiences a vibration caused by a rotating of the removable tool in the chuck;
- generating transformed vibration data by transforming the raw vibration data; and
- using a machine learning model, analyzing the raw vibration data and transformed vibration data to determine whether there is a piece of material caught between the removable tool and the chuck.

10. The method according to claim **9**, wherein the machine learning model includes a convolutional neural network.

11. The method according to claim **10**, wherein the convolutional neural network includes a first channel whereby the convolutional neural network receives the raw vibration data for analysis and a second channel whereby the convolutional neural network receives the transformed vibration data for analysis.

12. The method according to claim **9**, wherein the sensor is a vibration sensor.

13. The method according to claim **9**, the method further comprising sending a first signal to interrupt a machining process, sending a second signal to cause a notification indicating that the piece of material is caught between the chuck and the removable tool to be sent to a user, or both.

14. The method according to claim **9**, the method further comprising receiving, with a local computer, the machine learning model from a server.

15. The method according to claim **14**, wherein the server is configured to train the machine learning model using vibration data collected from one or more different machines, using one or more different removable tools of one or more different ages to manufacture one or more different objects.

16. The method according to claim **9**, wherein transforming the raw vibration data includes applying a Fast Fourier Transform to the raw vibration data.

17. A method for detecting when material caught between a chuck and a removable tool, the method comprising:
- receiving raw vibration data from a sensor mounted on a surface that experiences a vibration caused by an operation of the removable tool in the chuck; and
- using a machine learning model, analyzing at least one selected from the group consisting of the raw vibration data and transformed vibration data generated from the raw vibration data to determine whether there is a piece of material caught between the removable tool and the chuck.

* * * * *

4 Unsupervised Feature Learning

Labeling and annotating the data is still the most significant barrier in machine learning and especially in DL applications. From the conclusions and findings of the first section, this is mainly due to the human expertise required to label the data. As mentioned in the introduction, this research aims to bypass domain knowledge using DL methods, and for this reason, I start by investigating the unsupervised methods, which benefit from the abundance of sensory data and do not require labeling effort.

The following paper answers research question 3 and to some extent research question 2 by investigating the ability to learn from noisy raw data using the autoencoder method, which allows not only the extraction of relevant features from the raw data but also the compression and encoding of the data in the edge system presented in paper 1. To answer research question 2, the results of the paper show that using only unsupervised methods to extract relevant information from the noisy vibration data has its limitations. The extraction is successful for a few process operations but fails when applied to more complex processes, leaving an opening for further improvements. This is mainly due to the wide variety of machine processes mentioned in paper 1 and the different impacts of the anomaly on the different process operations.

Considering the real-time requirements of the real-world application, different state-of-the-art DL blocks are investigated. Finally, a lightweight architecture that best meets the manufacturing requirements is presented. The results of the following paper answer research question 3 and conclude that the width of the DL architectures plays a crucial role in feature extraction for noisy vibration data, as seen with the dilated convolutions and multilevel convolutions, while the depth of the DL architecture increases the complexity and training/inferencing time.

My contribution to this work is the conceptualization, design, and analysis of the results. I formulate the introduction, the preliminaries, the experiments, the results, and the conclusions of the work. Together with Paul Subarnaduti, I implemented and formulated the state-of-the-art and methodology section of the paper. As with the previous publications, the research work is supervised by Klaus Diepold and I am responsible for reviewing and revising the supplementary requirements of the reviewers.

Extract, Compress and Encode: LitNet an Efficient Autoencoder for Noisy Time-Series Data

Mohamed-Ali Tnani*†, Paul Subarnaduti†, Klaus Diepold†
*Department of Factory of the Future, Bosch Rexroth AG, Lise-Meitner-Str. 4, 89081 Ulm, Germany
†Department of Electrical and Computer Engineering, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany

2022 IEEE International Conference on Industrial Technology (ICIT) | 978-1-7281-1948-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICIT48603.2022.10002775

***Abstract*—The Industrial Internet of Things has emerged to enhance massive sensory data collection. Characterized by their large volume and hard interpretability, these data require highly skilled human expertise to compress and extract meaningful information from the raw data using computationally intensive signal processing methods. Deep learning, and autoencoders, in particular, have shown promising results in computer vision and natural language processing. In this paper, different state-of-the-art deep learning blocks are investigated and a lightweight Inception Network, called LitNet, is presented. The experiments are performed with real-world vibration data collected from different CNC machines during production. The models are evaluated based on their compression and feature extraction capabilities. The results show that LitNet outperforms all models and provides a good trade-off between compression and feature extraction. The latent vector analysis shows promising results and that the LitNet encoder can be used as a pre-trained feature extractor for vibration data.**

***Index Terms*—autoencoder, data compression, unsupervised feature learning, vibration data, manufacturing, deep learning**

I. Introduction

Driven by the fourth industrial revolution and the digitalization of manufacturing, the number of Internet-of-Things (IoT) devices is growing exponentially and a massive amount of data is gathered among countless sensors, machines, actuators, etc.

Being the most ubiquitous type of data collected in the industrial and automotive sector, high-frequency time series (HF-TS) generators produce constantly a huge amount of data that requires real-time processing, and ideally a transfer to the cloud for storage or further investigations. This leads to an increasing necessity of time-series (TS) processing methods running locally on the edge devices which require low computation power and agnostic data compression techniques to reduce the bandwidth usage, encrypt the sensitive manufacturing data, and minimize the volume that is sent to the cloud.

However, characterized by their noisiness and hard interpretability, HF-TS data, such as vibration data, requires high signal processing and domain expertise to extract meaningful patterns, detect anomalies, and monitor industrial assets.

Deep Learning (DL) opened the gate to new techniques for extracting useful features with rich information content from high-dimensional data. Autoencoders (AE) [1] is a subset of unsupervised deep learning methods which are distinguished by learning with no pre-assigned labels and having the ability to compress high-dimensional data to low encoded latent space. Using autoencoders on HF-TS fulfills several Industry 4.0 requirements. It reduces the need for domain expertise for data manipulation and annotation, encoding and compressing sensitive data before transferring it over the network, and extracting learned features from noisy high-dimensional data for further use such as process anomaly detection, process classification, and decision making.

In this paper, we present an evaluation of several types of autoencoders using state-of-the-art deep learning blocks. The goal of the papers is to define an efficient unsupervised DL feature extractor that runs on edge devices, compresses the high dimensional noisy data to low-dimensional space, and offers a reduced lossy reconstruction of the original data. The experiments are conducted on an open-source Computer Numerical Control (CNC) machining dataset collected during real-world production. The dataset has been collected from different machines over 2 years time period in order to project real-world manufacturing challenges such as drift over time and the high variety of process operations.

The paper is structured as follows. Section 2 gives an overview of the related work to compression and feature extraction methods in manufacturing. It is followed by a section where the dataset and used state-of-the-art deep learning blocks are introduced. Section 4 describes the autoencoder architecture and the methodology behind the LitNet model. In Section 5, the experimental concept conducted in this paper is presented followed by the model's evaluation. Finally, a conclusion section summarizes the study and gives an overview of the future works.

II. Related Work

During the last decades, data engineering and science have taken the main seats in the world of industry. Machine Learning (ML) and data-driven methods have been strongly researched and showed promising results.

Prior to DL methods, the traditional feature extraction and compression approach primarily involved statistical models and other machine learning algorithms. Principal Component Analysis (PCA) [2] is one of the most popular feature extractors in ML which aims to decompose the dataset into principle orthogonal components.

The introduction of DL algorithms for noisy time-series data, and autoencoders, in particular, is one of the latest advancements. To eliminate some of the bottlenecks of the traditional methods, Jia et al. [3] proposed efficient deep featuring autoencoder, where the model is trained layer-by-layer using the frequency spectra. The encoder part is then fine-tuned with a supervised algorithm. In the context of health monitoring of rotatory machines, several methods have been implemented in the literature. Shaheryar et al. [4] proposed an efficient feature extraction process for fault detection in rotatory machines. It follows a semi-supervised approach where each channel of the raw vibration data is pre-trained with autoencoders. The pre-trained weights are then used to train the stacks of convolutional layers which is then followed by another stack of denoising AE. It achieved much better results in comparison to Deep Belief Networks (DBN) and Support Vector Machines (SVM). Huang et al. [5] tackled the problem by using a Recurrent Neural Network (RNN) Variational Autoencoders (VAE) architecture that was proved to be more efficient in compressing the data compared to standard Convolutional Neural Network (CNN) AE architectures but with minimal generalization capability.

In the context of real-time anomaly detection, CNN-based autoencoders (CAE) have been widely explored for industrial sensor data. Chen et al. [6] have proposed a unique data augmentation technique for the high-dimensional sensor data in the form of sliding windows which is then fed in a CAE model. Few other works related to anomaly detection in rotatory machines primarily involve RNNs or VAEs. Kim et al. [7] proposed a squeezed convolutional VAE network for process anomaly detection that proved to be highly efficient for computing on edge devices.

The architectures proposed in the state-of-the-art literature largely focus on 1D-CNN, RNN-based Autoencoders, or VAEs. This paper goes beyond the multilayer CNN-based AEs and investigates more complex deep learning building blocks.

III. PRELIMINERIES

A. Data Set: CNC Milling Machine

CNC machines are characterized by their precision and high production speed and are among the most widely used rotatory systems in the industry. To analyze and monitor a rotating process, the expert mainly uses a vibration sensor to detect the oscillating motion of the spindle. This work uses a publicly available industrial vibration dataset collected during real-world production [8]. In order to tackle the challenges of data drift and generalization of data-driven approaches, normal and abnormal data from three different machines were streamed over four different periods of five months each. The complexity of the CNC machine monitoring use case is mainly caused by the numerous process operations that take place in a production flow. Therefore, during the collection campaign, data has been collected from 15 different process operations, each having a unique configuration and carried out with a different physical tool. A tri-axial accelerometer is used to collect and process the data and allow the operator to flag the process status. The acceleration data of the X-, Y- and Z- axes are acquired with a sampling rate of 2 kHz. Fig. 1 gives an overview of the dataset. For each process operation, normal and abnormal data were collected from 3 different machines (M01, M02, M03) and at different time frames.

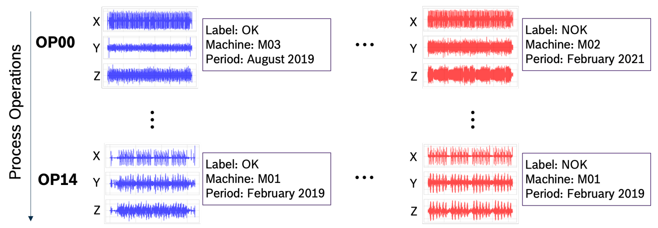


Fig. 1: Overview of the Bosch CNC Milling Machine dataset.

B. Deep leaning building blocks

This work explores several variants of convolutional operations inside an AE network as it tries to decode the importance of the individual variants in terms of feature extraction and data compression. Starting with standard convolutional layers, it gradually transits into wider and deeper forms of networks that require several modifications to use in the context of time-series data. Deep CNN networks like AlexNet [9], VGG [10] have shown promising results in the context of images but consists of millions of parameters and require high computational power and memory consumption. Our paper studies a lighter version of deep-stacked CNN networks for noisy time-series data that runs on small compute instances. Residual connections used in ResNet [11] has helped to improve the performance of dense networks by adding shortcuts at different locations. Our paper explores the importance of the depth of the network and the usability of shortcut connections. Two more unique variants of convolutional layers are investigated: dilated convolutions, the core block of Temporal Convolutional Network (TCN) [12] and depthwise separable convolution (DSC), the core block of Xception [13]. Dilated convolutions are computationally efficient and offer larger receptive fields which extract meaningful temporal dependencies in the input data [12]. DSCs exploit the benefits of channel-wise feature extraction by performing spatial convolutions independently over each channel, followed by a point-wise convolution to project the extracted features in a new channel [13]. This reduces considerably the number of parameters of the block. Lastly, we explore the power of Inception blocks [14] as

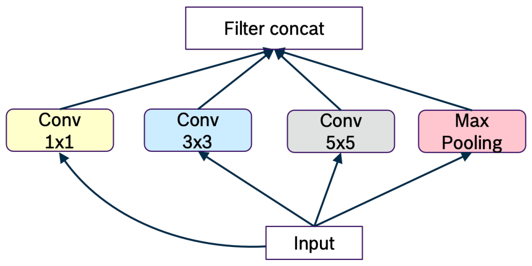


Fig. 2: Building block of the multi-level convolutions "Inception" [14].

shown in Fig. 2 which perform convolutional operations of different filter sizes in a parallel manner and concatenate the outputs afterward. This technique enables a multi-level feature extraction which improves the performance of dense networks by making the block wider instead of deeper. Fig. 2 describes the vanilla Inception block used for two dimensional input data where filter sizes are depicted by NxN, used to extract features at different spatial locations. Fawaz and al. presented in [15] a novel idea using Inception blocks for time series classification. It inherited the concept of Inception V4 architecture and proposed a highly scalable and efficient deep CNN network with larger filter sizes with 10, 20, 40 being chosen as the optimal sizes. However, the drawback of this approach is that it increases the number of parameters and computation time considerably. The approach proposed in our paper takes advantage of convolving with large receptive fields while reducing computational time and model size.

IV. Methodology

A. Autoencoder backbone

This section provides an overview of the generic architecture proposed for all different AEs implemented in the scope of this work. As shown in Fig. 3, convolution layers form the basic building blocks for all the proposed AE architectures. The encoder and decoder networks are divided into three distinct flows: entry, middle, and exit. Each flow serves a vital role in the feature extraction process at different levels.

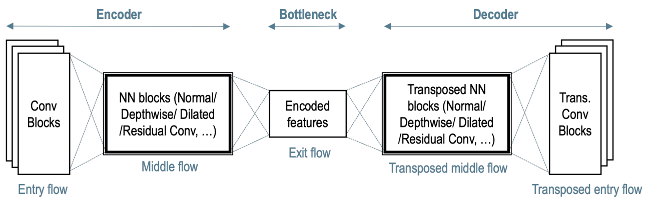


Fig. 3: Overview of the generic modeling of the AEs. Entry and exit flows are identical for all models, only the middle flow varies.

1) Entry flow: It defines the initial layers of the encoder network that takes the raw vibration data as input. It comprises multiple convolution layers that help extract high-level features from the input data. Unlike images, the input vector length for vibrational time-series data is much wider in length, which is equal to 4096 data points for each axis in the scope of this work. The initial convolutional layers typically use a large filter size that aids the network to learn a more accurate representation of the high dimensional data and significantly improves the accuracy of the AEs in reconstructing the original input. As shown in Fig. 4, the noisy data sample is fed to a convolutional layer with a kernel size of 127 followed by a Max-pooling layer for dimensionality reduction. The transposed entry flow, which is the transposition of the encoder entry flow, represents the final layers of the decoder network that completes the decoding of the encoded feature maps.

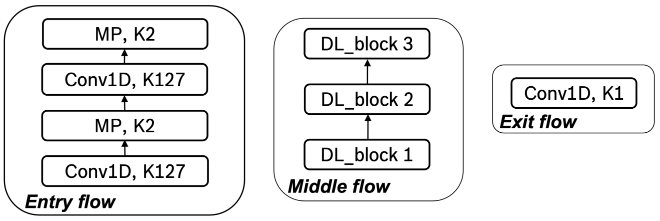


Fig. 4: Overview of the three different flows of the encoder network of AE architecture. K: kernel size, Conv1D: 1-D convolutional layer, MP: max-pooling

2) Middle flow: This flow varies for all the proposed AE architectures as it undergoes significant modifications in each network. It hosts modified versions of several state-of-the-art DL architectures defined in Section III-B that can be used for the purpose of time-series data. The middle flow plays a crucial role in finding an optimal AE architecture as it is mostly responsible for determining the depth and the width of the network. It also undergoes extensive hyperparameter tuning to explore the depth and the computational complexity of the network. However, this work sets the number of DL blocks inside the middle flow to three to propose a network that is suitable for real-time inferencing on edge devices. When passing through the decoding part of the network, the middle flow, as mentioned in Fig. 3, uses several layers of transposed convolutional operations that follow the structure of the encoding part to decode the encoded features. An overview of the model's hyperparameter tuning conducted is presented in the following section.

3) Exit flow: The exit flow is used as the final terminating block for the encoder network and as an entry block for reconstructing the original data in the decoder network. It primarily hosts only one bottleneck layer consisting of a pointwise convolution. As this work aims to explore the data compression capability of AE alongside its feature extraction efficiency, the encoded features from the middle flow are compressed to a low dimensional latent vector. All AE models, except the baseline model, achieve a compression rate equal to 24.

B. LitNet: a Lightweight Inception based AE Network

This paper proposes a unique lightweight AE network based on Inception, which we call LitNet. The architecture closely depicts the ideas used while structuring the Inception-V3 [16] and Inception-V4 network [17]. It implements a simplified version of Inception-V4 architecture in the scope of time-series data which combines the importance of using smaller convolutions and residual connections inside the Inception blocks. The proposed architecture follows the AE backbone proposed in the previous section, i.e. entry, middle, and exit flow. The entry and the exit flows are kept identical as other AE architectures while the middle flow consists of three unique inception blocks, named Inception_A, Inception_B, and Inception_C. The Inception blocks follow an asymmetric architectural design among each other. An extensive hyper-parameter tuning is performed on the Inception block to derive an optimal and efficient model for the Inception blocks. Our LitNet network offers a significant reduction in model complexity, keeping the number of parameters for the network below one million.

The convolutional filters are factorized to different filter sizes in each of the Inception blocks. However, every block consists of four different convolutional filters along with a max-pooling layer to extract essential filters at different temporal lengths. The use of Max Pooling operation instead of Average Pooling in this work has shown better performance on the TS feature extraction. Unlike pixels in an image, it is essential to spot the peaks within the vibrational data at each time step. The convolutional block consists of a convolutional layer followed by a batch normalization and a linear activation. The use of unbounded linear activation function is explained by the fact that vibrational data is a real number, i.e. $\in \mathbb{R}$, and thus allows the reconstruction of the input data. Finally, the architecture is designed to achieve an Inception-inspired AE that has the optimal balance between the depth and the width of the network.

Fig. 5, Fig. 6, and Fig. 7 describes the architecture of Inception_A, Inception_B, and Inception_C blocks respectively. A common colour pattern is followed for all of the above mentioned figures. A convolutional layer with kernel size 1, 3, 5, and 7 are colored in yellow, blue, grey, and green respectively. Whereas, the max-pooling layer is colored in pink.

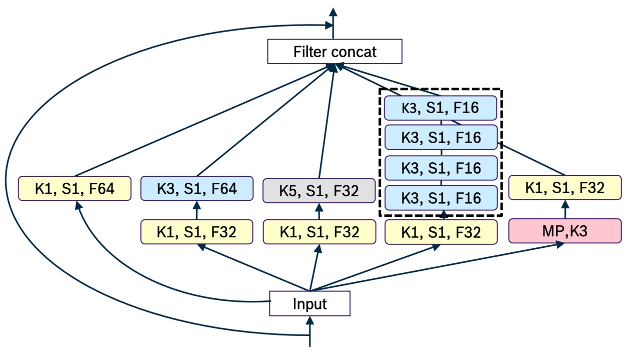


Fig. 5: Architecture of Inception_A block. K: kernel size, S: Stride, F: Filter channels, MP: Max Pooling layer

Inception_A: It can be described as an enlarged version of the 1-D vanilla Inception module where an additional layer of convolutional operation with a larger filter size is added. Most specifically, it follows an approach of factorizing the larger filters into smaller convolutions. As shown in [16], convolutions with large spatial filters tend to be more computationally expensive without contributing much to the overall performance. The architecture is solely based on convolutional filters with a shape of N x 1 where N represents an integer value. Due to the one-dimensionality nature of the HF-TS data, it is not possible to establish asymmetry between the convolutional layers inside the network as explained in [16]. To extract essential information at different temporal lengths, the filters are factorized to 1, 3, 5, and 9 where the filter size of 9 is further factorized into smaller filters. The architecture highlights the re-factorization of the filter size 9 with a black bounding box that consists of four smaller-sized filters of 3 stacked sequentially. A point-wise convolutional layer precedes each of the N x 1 filters as it helps to limit the computational cost of the network. A residual connection is used as a part of the network and connects the input to the output as shown in Fig. 5. One of the reasons to use residual connections between the entry and the middle flow is to transfer essential input features before being compressed to lower dimensions. Besides that, extensive experimentation has led to the choice of using the residual connection only at the first block of the network. The number of filter channels used for each convolutional layer is subject to extensive hyperparameter tuning.

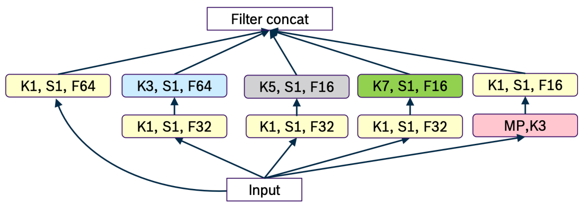


Fig. 6: Architecture of Inception_B block. K: kernel size, S: Stride, F: Filter channels, MP: Max Pooling layer

Inception_B: Comparing the architecture of the previous Inception_A network, this block gradually aims to decrease the filter sizes as we proceed to the final stages of the entire architecture. It is sequentially stacked with the Inception_A block inside the network as it forms the second essential feature extraction block. As the input dimension of the data decreases with the depth of the network, it uses convolutional layers with filter sizes of 1, 3, 5, and 7. Instead of using a filter size of 9, it is decreased to 7. The rest of the architecture is kept identical to the base Inception block besides the point-wise convolutional layers to limit the computational cost.

Inception_C: The third and final block of our network, Inception_C, follows the overall architecture used in the Inception_B. It again follows the approach of factorizing large convolutional filters into smaller filter sizes. However, in this architecture, we went one step further where we re-factorize two large filters instead of one as in Inception_A. As in Inception_B, we use convolutional filters of sizes 1, 3, 5, and 7. We again factorize the filters with sizes 5 and 7. The modifications are highlighted in Fig. 7 with black bounding boxes. We use two small filters of size 3 for filter 5 and three small filters of size 3 for filter 7. Such implementation holds the power to improve the generalization ability of the network, as it looks closely into finer trends in the raw data.

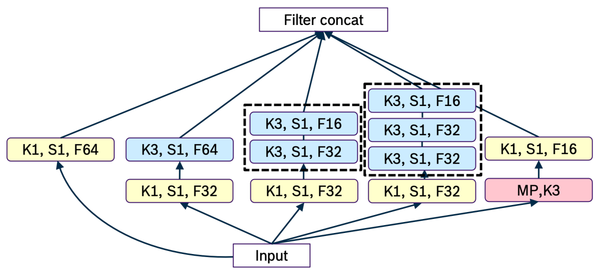


Fig. 7: Architecture of Inception_C block. K: kernel size, S: Stride, F: Filter channels, MP: Max Pooling layer

V. Experiments

A. Data set partitioning

Generalization is one of the major obstacles in industrial ML. In fact, data-driven algorithms are constantly challenged by data drift caused, for example, by components that wear out over time, and by the rapid implementation of new processes in production due to technological progress. The data partitioning in this work maps these scenarios and is presented in Fig. 8. The training set consists of samples from 12 different machine operations out of 15, leaving 3 operations reserved for the test set. The test set contains all 15 machine operations, including the 3 tool operations unseen in the training set. In addition, the samples in the test set are picked from a different time-period than the training set to validate the data drift over time. In a real-world scenario, the data are streamed and processed in chunks. For that purpose, the data are windowed using a sliding window method with a window length of 4096.

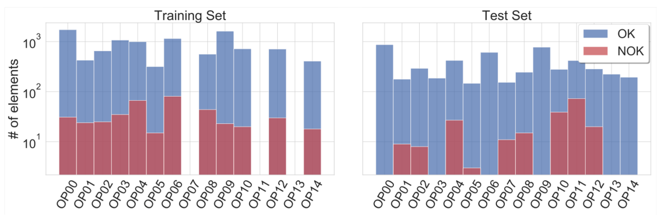


Fig. 8: Class distribution per process operation within the training set (left) and the test set (right).

B. Training Set up

In this work, eight groups of experiments with different AE architectures are conducted. It gradually transitions from a shallow AE network to a more complex deep network. The search for a highly scalable AE architecture begins with Baseline AE. It defines the generic form for the entry flow used for the other AE networks implemented in this work. The middle flow is intentionally kept empty for this network, while the exit flow consists of a single bottleneck layer. The shallow Baseline AE is gradually made more profound with the multiple stacked CNN layers (S-CNN), residual connections (Res-CNN), and depthwise-separable convolutions (DS-CNN) within the middle flow. Besides the standard and depthwise-separable convolutional layers, dilated convolutional layers (Di-CNN) are experimented with using dilation rates of 2, 4, and 8 gradually. The next set of experiments explores the importance of the width of the network with Inception AE, where the 3 DL blocks from the middle flow are replaced by standard 1-D inception blocks having kernel sizes of 1x1, 3x1, and 5x1. The final experiment is the LitNet architecture proposed in the previous section.

The training of the models are performed on GPU NVIDIA Tesla K80 and each model architecture goes through an extensive hyperparameters search using the ranges presented in Table I. In the scope of this work, we have used Optuna framework [18] to tune our training parameters. To have a comparative evaluation of the different model architectures, certain training parameters, such as the number of epochs, batch size, and optimizer, have been fixed. Each AE is trained for 40 epochs with a batch size of 32 and AdamW optimizer [19]. Learning rate is determined through hyperparameter tuning and it varies for the individual model. Although the optimal learning rate for all the models is found in the range of 1e-4.

Parameters	S-CNN	Res-CNN	Di-CNN	DS-CNN	Inception
Filters	[16,..,128]	[64,..,256]	[16,..,64]	[32,..,128]	[16,..,64]
Kernels	[3,..,15]	[3,..,7]	[3,..,5]	[3,..,7]	[3,..,11]

TABLE I: Choices for training parameters for all the proposed model.

VI. Performance Analysis

In this section, we evaluate the models from different aspects. We first evaluate the compression performance based on the reconstruction loss and the generalization capability on unseen processes. The real-time capability of deep learning feature extractors on edge devices is then discussed based on the training and inference time of the different architectures. Lastly, the unsupervisingly extracted features are visualized and discussed.

A. Compression performance

Data compression plays an essential role in industrial IoT. Diminishing the data volume facilitates the data traffic over the network by reducing bandwidth. The AE models presented in the previous section offer a compression rate equal to 24 for all the models. The compression performance is evaluated using the Mean Squared Error (MSE) calculated using the following equation:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}(x_{i,j} - x_{rec_{i,j}})^2 \tag{1}$$

with n denoting the number of samples, m being the channel number, i.e. the $\{X-, Y-, Z-\}$ vibration axis, x the input sample and x_{rec} the reconstructed input. As we can see in Table II, the MSEs of the implemented models outperform the baseline model, showing that deeper architectures improve reconstruction accuracy and reduce compression loss. It also highlights that dilated and multi-level convolutional layers significantly reduce the test loss by a factor of 10 compared to the baseline model. LitNet delivers the best MSE of 0.024, which is a 50% improvement over Di-CNN and Inception.

id	Model	MSE (Test set)	Latent vector length	CR	Training time (s)	Encoding time/w (ms)	Decoding time/w (ms)	Number of parameters
0	baseline	0.530	1024	12	1177	5.4	7	26,849
1	S-CNN	0.375	512	24	1608	6.4	8	47,177
3	Res-CNN	0.216	512	24	6030	29.3	25	1,034,442
4	Di-CNN	0.047	501	24.5	2341	13.5	14.8	194,823
5	DS-CNN	0.379	512	24	12415	18.6	41.6	170224
6	Inception	0.052	512	24	6552	26	24.1	617,732
7	**LitNet**	0.024	512	24	8696	32.2	30.8	723,756

TABLE II: Performance evaluation of the different autoencoders. The training is run on a GPU NVIDIA Tesla K80. The inference is performed on a CPU IntelCore i7 9850H.

DS-CNN performs poorly and has the second-highest reconstruction loss of 0.379 (after the baseline). However, looking at the MSE for each process in Fig. 9.a, a valuable advantage of depth-wise separable convolution becomes apparent. It shows that the MSE of OK samples in the training set is relatively low compared to NOK samples and OK samples that are unseen during training, such as OP13 and OP14. This means

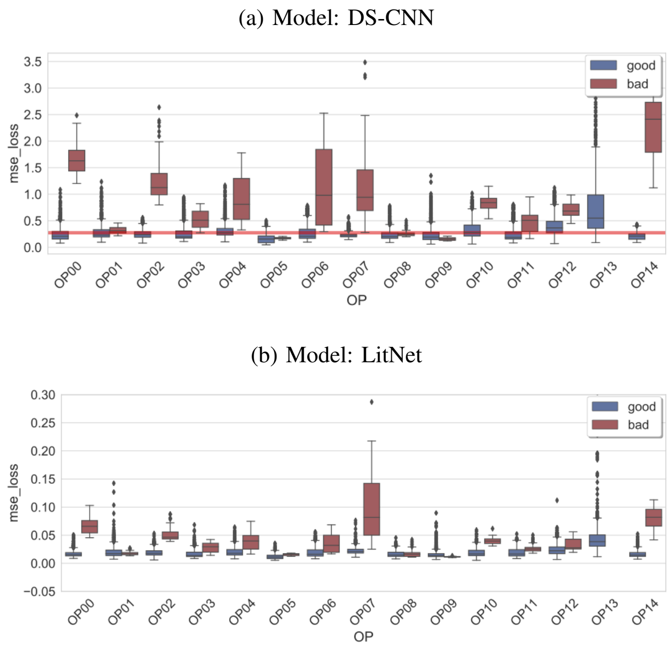


Fig. 9: Distribution of the MSE recontruction loss over the different machine process operations. For analysis purposes, the red line in (a) shows the maximum MSE reached by the LitNet (b).

that depth-wise separable convolutions tend to overfit the samples in the training set and require a large number of samples, making DS-CNN a good candidate for anomaly and novelty detection. Fig. 9.b shows the generalization capability of LitNet in reconstructing unseen process operations. Fig. 10 shows the compression performance of the LitNet model on an unseen machine operation (OP11) from the test set. This proves the data compression reliability of the proposed approach.

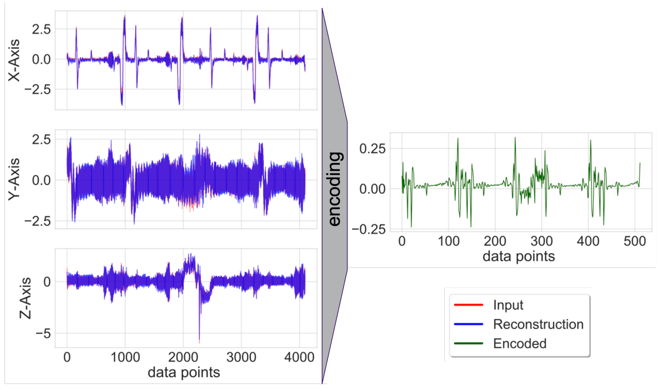


Fig. 10: Compression performance of the LitNet model on an unseen machine operation (OP11) from the test set.

Despite the reconstruction loss, the timing of compression is an essential criterion in the development of industrial applications. One can mention two scenarios: First where solely the encoding, i.e. features extraction, is performed on the edge. The second, where both encoding and decoding take place on the edge. The second scenario is chosen when AEs are used for anomaly detection tasks based on the reconstruction error. However, this can also be used as a monitoring technique to detect when the compression is faulty and retraining is required.

In Table II, the encoding and decoding time for a single-window is illustrated. A window represents one tri-axial vibration data segment of dimension (4096,3) which corresponds to a 2048ms segment. In the input dimension, 4096 represents the window length and 3 represents the number of axes. LitNet is the slowest model in the list and takes 32.2ms to encode a single window. This operation is considered real-time since it uses only 1.57% of the time to process one input segment before the next window is queued. However, it is important to highlight that Di-CNN, the model with the second-best MSE score, requires less than half LitNet's time to encode the tri-axial vibration segment. This is explained by the computation efficiency of dilated convolutional layers which offer larger coverage with fewer parameters. This advantage is reflected in the training time as well. To train the 40 epochs, Di-CNN takes only 2341 seconds, which is 3.7 times faster than LitNet.

For industrial applications, reconstruction quality is key and more critical than computational speed, making LitNet the best performing model, which is slower but still under 35 ms/window.

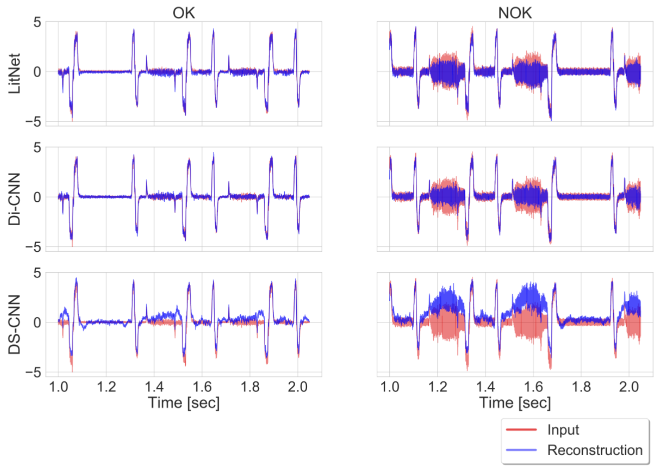


Fig. 11: Comparison of the X-axis vibration data of an unseen tool operation (OP07) with its reconstruction signal mapped by the LitNet, Di-CNN, and DS-CNN models separately. The left and right columns show the tool operations OK and NOK, respectively.

B. Features analysis

In this section, we evaluate the latent space representing the information content extracted by the encoder model. For this purpose, we examine the reconstruction signal of both OK and NOK samples from an unseen tool operation, i.e., OP07. Fig. 11 shows the comparison of the X-axis of the input raw signal with its reconstruction mapped by Di-CNN and LitNet, the two best-performing compression models, and DS-CNN, which showed promising anomaly detection capabilities. LitNet reconstructed both the OK and NOK signals the best. The anomalous patterns occurring at seconds 1.2, 1.6, and 2.0 in the NOK signal were better encoded in LitNet's feature vector than in Di-CNN, where there is some loss of information due to the dilation operation. In contrast, DS-CNN fails to reconstruct these patterns and overshoots in the anomalies sections as seen in the figure.

To further investigate the latent space, the high dimensional feature vector is projected into a lower-dimensional space using principal component analysis (PCA) [2]. To analyze the feature space in 2-D, only the first two principal components are considered. Fig. 12 illustrates the 2-D feature maps of LitNet, Di-CNN, and DS-CNN. The figure shows the distribution of OK and NOK classes. It shows that LitNet and DS-CNN have better separation of the anomaly classes than Di-CNN. This can be explained by the results of Fig. 11, where we see that Di-CNN has difficulty in reconstructing the anomaly patterns. However, DS-CNN slightly outperforms LitNet, and only when looking at the first principal component. However, the second principal component shows the rich information content of the features extracted by LitNet compared to DS-CNN.

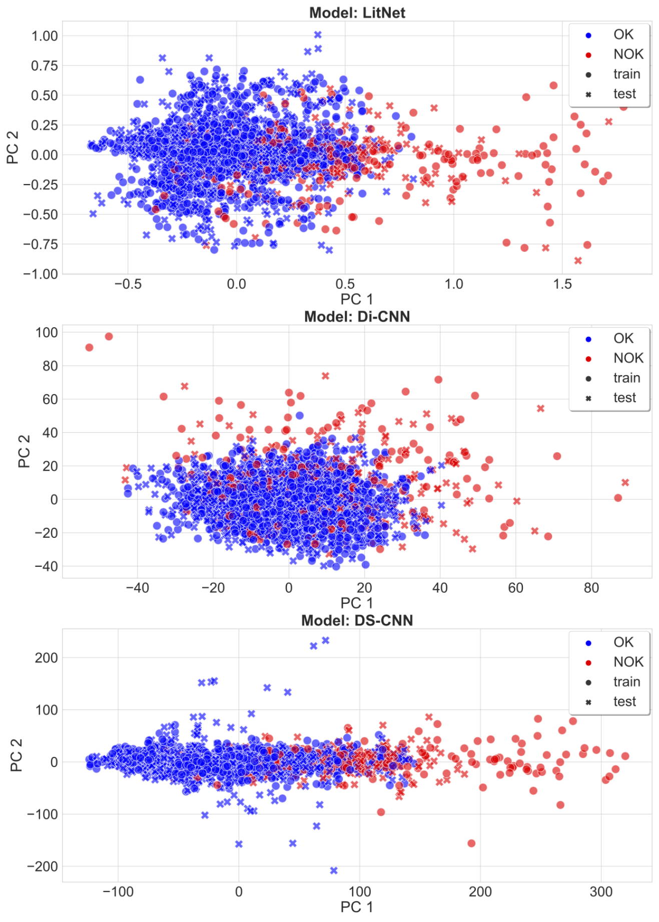


Fig. 12: Feature map of the encoded OK and NOK latent space resulting from the LitNet, Di-CNN and DS-CNN encoding models.

VII. Conclusion

In this paper, different autoencoder architectures have been investigated in terms of compression and feature extraction performance. The experiments performed gradually transits from a shallow autoencoder to a more complex network. A lightweight inception network called LitNet is proposed. It provides multi-level feature extraction for high-frequency time series data using sequential convolutions with small-sized filters, keeping the number of parameters small. Experiments show that deep networks with wide blocks, such as dilated convolutions and multi-level convolutions, perform the best on noisy vibration data. The proposed network outperforms all models in encoding performance with a mean square reconstruction error of 0.024 and provides promising results in feature extraction. The results also show that the proposed model meets the industry requirements as it is shown to be highly scalable across new tool operations. The improvement in generalizability is explained by the stack of smaller filters, which simultaneously reduces the number of parameters in the model, making it real-time capable. Future work will investigate the supervised tuning of the LiNet architecture and its feature extraction capability. Keeping in mind the industrial challenges, methods that use a limited number of labeled samples will be considered.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[2] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[3] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, “Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data,” *Mechanical Systems and Signal Processing*, 2016.

[4] A. Shaheryar, Y. xu, and W. Yousuf, “Robust feature extraction on vibration data under deep-learning framework: An application for fault identification in rotary machines,” *International Journal of Computer Applications*, vol. 167, pp. 37–45, 2017.

[5] Y. Huang, C.-H. Chen, and C.-J. Huang, “Motor fault detection and feature extraction using rnn-based variational autoencoder,” *IEEE Access*, vol. 7, pp. 139 086–139 096, 2019.

[6] T. Chen, X. Liu, B. Xia, W. Wang, and Y. Lai, “Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder,” *IEEE Access*, vol. 8, pp. 47 072–47 081, 2020.

[7] D. Kim, H. Yang, M. Chung, S. Cho, H. Kim, M. Kim, K. Kim, and E. Kim, “Squeezed convolutional variational autoencoder for unsupervised anomaly detection in edge device industrial internet of things,” pp. 67–71, 2018.

[8] M. Tnani, “Bosch Research Dataset: CNC Machining Data,” 2021. [Online]. Available: https://github.com/boschresearch/CNC_Machining

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, pp. 84–90, 2012.

[10] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[12] K. Guirguis, C. Schorn, A. Guntoro, S. Abdulatif, and B. Yang, “Seld-tcn: Sound event localization detection via temporal convolutional networks,” *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021.

[13] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” pp. 1–9, 2015.

[15] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, “Inceptiontime: Finding alexnet for time series classification,” *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[17] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[18] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.

[19] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.

5 Two-Stage Feature Learning

Based on the results from paper 2, unsupervised feature learning revealed some limitations. Motivated by the presence of a limited amount of labeled data in manufacturing, in this section I investigate the semi-supervised methods and in particular the "few-shot" metric-based learning techniques. The research consists of investigating the prototypical few-shots learning method, which has shown promising results in computer vision and natural language processing. It is used as part of the proposed two-stage feature learning method and evaluated in comparison with conventional prototypical learning. The proposed few-shots method is compared with the conventional supervised method and the handcrafted feature extractor.

The results of the following journal article address research questions 2 and 4. The evaluation of the results concludes that the semi-supervised DL method, which combines the benefits of the unsupervised autoencoder method and the prototypical few-shots method, outperforms the handcrafted method in terms of computation time, performance, and robustness. With faster computation time and lower sensitivity to data drift, the proposed method has demonstrated its superiority over the handcrafted methods and reduces the overhead of domain knowledge and human labeling. Moreover, the results show that fine-tuning does not always improve the performance and robustness of DL models and is highly dependent on the method. Based on the results of the paper, we see poor performance when using conventional supervised fine-tuning methods in contrast to "few-shots" learning methods. This is explained by the limited amount of labeled data in real-world use-cases, making the conventional supervised fine-tuning method overfitting on the small dataset.

The contributions of the authors are depicted in the journal.

sensors

Article

Efficient Feature Learning Approach for Raw Industrial Vibration Data Using Two-Stage Learning Framework

Mohamed-Ali Tnani [1,2,*], Paul Subarnaduti [2] and Klaus Diepold [2]

[1] Department of Factory of the Future, Bosch Rexroth AG, Lise-Meitner-Str. 4, 89081 Ulm, Germany
[2] Department of Electrical and Computer Engineering, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany; ge73gid@mytum.de (P.S.); kldi@tum.de (K.D.)
* Correspondence: mohamed-ali.tnani@boschrexroth.de; Tel.: +49-899-2139-651

Abstract: In the last decades, data-driven methods have gained great popularity in the industry, supported by state-of-the-art advancements in machine learning. These methods require a large quantity of labeled data, which is difficult to obtain and mostly costly and challenging. To address these challenges, researchers have turned their attention to unsupervised and few-shot learning methods, which produced encouraging results, particularly in the areas of computer vision and natural language processing. With the lack of pretrained models, time series feature learning is still considered as an open area of research. This paper presents an efficient two-stage feature learning approach for anomaly detection in machine processes, based on a prototype few-shot learning technique that requires a limited number of labeled samples. The work is evaluated on a real-world scenario using the publicly available CNC Machining dataset. The proposed method outperforms the conventional prototypical network and the feature analysis shows a high generalization ability achieving an F1-score of 90.3%. The comparison with handcrafted features proves the robustness of the deep features and their invariance to data shifts across machines and time periods, which makes it a reliable method for sensory industrial applications.

Keywords: feature learning; CNC machining; machine monitoring; machine learning; few-shot learning; vibration data; two-stage learning

check for updates

Citation: Tnani, M.-A.; Subarnaduti, P.; Diepold, K. Efficient Feature Learning Approach for Raw Industrial Vibration Data Using Two-Stage Learning Framework. *Sensors* **2022**, *22*, 4813. https://doi.org/10.3390/s22134813

Academic Editors: Hamed Badihi, Tao Chen and Ningyun Lu

Received: 30 May 2022
Accepted: 23 June 2022
Published: 25 June 2022

1. Introduction

The latest advances in technology coupled with an aim to realize smart intelligent systems have contributed to a rapid move towards the next industrial revolution. Unlike the third industrial revolution powered by electronics and information technology, digitization and automation have been the front runners to revolutionize industry to its fourth chapter. The fourth industrial revolution has proved to be a boon to the traditional machining processes as it brings some key advantages such as improvement in the production and quality, cost reduction, and monitoring of machining processes in real time. As a result, condition monitoring and process condition monitoring systems are integral parts of intelligent manufacturing that support the quality inspection. Such highly automated systems rather support the flow of huge volumes of data that can be analyzed in real time without interrupting any machining workflow [1].

Enabled by the significant advancements in industrial Internet of Things (IIoT), the process involved in collecting and monitoring data from industrial environment is made more convenient. The initial step usually involves the acquisition of different types of signals such as vibration, cutting force, and a few others that can determine the health of machining parts and tool processes. This work largely focuses on the vibration-based signals as it provides critical information about the machining health. However, the vibration signals collected from the sensors are largely affected by several environmental factors and are commonly characterized by their nonlinearity, nonstationarity and noisiness. This brings us to the next steps of monitoring systems that are filtering the collected signals [2]. Feature extraction as means of signal filtering is a crucial step in the data processing pipeline. With the gradual development in machine learning (ML) algorithms and eventually deep neural networks, the idea of feature extraction from the raw vibration signals has varied over time. Traditionally, the feature extraction mainly involved signal processing techniques such as statistical analysis on the time, frequency or time–frequency domain [3–6]. Although these techniques have produced fair results over the years, they also present some major drawbacks. These algorithms often require extensive domain knowledge as well as human expertise specifically designed for a specified task. As the volume of the collected data increases, which is particularly huge in modern automated smart systems, the effort as well as time to produce meaningful representation increases. One implementation done by Christ et al. named as TSFRESH [7] has achieved remarkable results. It can automatically extract statistically based features and observe dynamics without much human expertise. Lines et al. [8] also presented a hierarchical transformation ensemble method for time series classification. However, these methods fail to meet the demands of a fast reliable algorithm due to the high computational time.

Recently, encouraged by the outstanding performance of deep learning (DL) in several fields, some interesting end-to-end DL algorithms have been proposed to replace traditional time-consuming monitoring systems. Unlike image processing, research in machine monitoring has mostly overlooked the advantages of deep neural networks due to their hard interpretability and their nonacceptance in industry [9]. Nevertheless, state-of-the-art research works [10,11] have integrated DL techniques on vibration data that are treated as one-dimensional time series data and showed state-of-the-art results, bypassing the handcrafted-based methods. However, these supervised methods require a huge quantity of labeled data to achieve satisfying performance. Data annotation is another critical factor in real-world production plants, as labeling large quantities of data is often an inconvenient, costly, and erroneous approach under human supervision. Moreover, in a highly automated system, the occurrence of anomalies is a rare event that causes a huge imbalance between OK and NOK samples. These factors deteriorate the performance of supervised DL algorithms that fail to generalize on noisy time-series (TS) data. To tackle these shortcomings, unsupervised feature extraction technique has proved to be promising. In particular, autoencoders have been found to be most beneficial algorithm [12]. Sun et al. [13] showed that a sparse autoencoder with a small number of trainable parameters can learn good features based on induction motor data. Shao et al. [14] also proposed a work that addresses the generalization of autoencoders on unseen working conditions in fault diagnosis.

Despite its huge success, conventional DL techniques require huge quantities of data to offer meaningful generalization on unseen data. In the literature, the problem of insufficient labeled data samples has been handled in different ways. Data augmentation plays a crucial role in processing such raw vibration data. Overlapping input data samples to generate small snippets of new samples is one such technique used by [10]. In the fault diagnosis applications, a few works have showed how data augmentation could generate new synthetic samples using GANs [15,16]. However, they suffer from overfitting problems. To overcome the challenge of limited labeled data, certain ML algorithms named few-shot learning (FSL) methods have been proposed in the state-of-the-art literature [17]. Such a learning paradigm has been designed to tackle scenarios where data with appropriate labels are difficult to produce, such as in an industrial environment. Considering a small training dataset (x, y), FSL can be best described as an optimization algorithm that searches for the best hypothesis space from x to y described by the set of optimal parameters [18]. Current state-of-the-art literature has produced FSL for various applications mostly featuring computer vision tasks and only a few implementations can be found for time-series classification. The authors in [19–22] proposed a metalearning model for few-shot fault diagnosis applications. The prototypical network is also a popular FSL technique for time-series classification. It has proved to achieve state-of-the-art results for both few-shot and zero-shot classification problems [23,24]. Tang et al. [25] proposed a novel few-shot learning approach for time-series classification. In the feature learning research on rolling bearing fault diagnosis, Wang et al. [21] proposed a metric-based metalearning method named relational network which learns fault features from the input FFT frequency signals. A few studies [21,26] also explored few-shot learning for fault diagnosis on rotatory machines such as CNC machines.

To address the problem of costly data annotation and the imbalance between normal and abnormal machine faults, this work aims to propose a novel two-stage feature learning framework using the prototypical few-shot technique. Recently, researchers have largely benefited from the two-stage frameworks, which have gradually attracted a lot of attention. The existing methods in the state-of-the-art literature fall into two categories. The first category is the two-stage predicting category, which aims to improve the performance of the prediction task by decomposing the application task into two sequential tasks. Few studies [6,27,28] have explored the two-stage predicting category. To detect defective rolling element bearings, Yiakopoulos et al. [6] presented a two-stage method, where the first task was to detect the existence of a bearing fault while the second stage task classified the type of detected anomaly. The second category is the two-stage learning framework, which aims to improve the learning following a graduated training methodology. In the image processing field, Das et al. [29] tackled the problem of the high dimensionality and the variable variance among the base classes with a two-stage feature learning approach. The first stage produces a relative feature extractor, while the second stage handles the classification task by measuring the variance using distance metrics such as the Mahalanobis distance. Afrasiyabi et al. [30] aimed to represent rich and robust features from input images using mixture-based feature learning (MixtFSL). The proposed end-to-end approach learned in a progressive manner till the best feature representation was achieved. Ma et al. [31] proposed a two-stage training strategy called partner-assisted learning, where soft anchors were generated by a partner model in the first stage and the main encoder was trained by aligning its outputs with the soft anchors in the second stage. In wind turbine condition monitoring applications, Afrasiabi et al. [32] presented a sequential training pipeline that resolved the limited data problem by generating artificial data in the first stage and training a robust deep Gabor network in the second stage.

This work falls into the second category and proposes a novel two-stage feature learning framework for industrial machining processes. The study focuses on the performance of the resulting feature extractor trained with limited labeled data, its ability to generalize over unseen machining process operations with different working conditions as well as its robustness against data drift. The work is divided into sections. The second section presents the background of the prototypical network (PN) and the different distance measures used in this work. In the third section, we define the smart data sampling technique for noisy time series and the proposed two-stage learning approach. In the fourth section, we introduce the publicly available Bosch machining dataset and present a real-world scenario mapped in the dataset-splitting part. In the fifth section, we describe the experiments performed, followed by an in-depth analysis of the results as well as a comparison with different types of feature extractors. Finally, we conclude with a short summary and the prospect of some future work.

2. Background

2.1. Prototypical Networks

This work greatly focuses on prototypical networks [23] for few-shot learning. For an N-way and K-shot FSL, we have a small training dataset $\mathcal{D}$ with k labeled samples. $\mathcal{D} = \{(x_1, y_1), .., (x_k, y_k)\}$, where x_i represents a D-dimensional input feature vector and each y_i represents its corresponding label. The training is divided into several episodes termed as training episodes. For each episode, training sets are sampled to form a support set $\mathcal{S}$ and a query set $\mathcal{Q}$.

Support Set: A random subset of classes from the training set is selected as support set containing K examples from each of the N classes.

Query Set: A set of "testing" examples called queries.

Taking each class into consideration, prototypical networks generate the embedded points for each example in $\mathcal{S}$ using an embedding function f_θ. For each class N_k, a mean vector of the embedded points C_k is computed using Equation (1) and represents the prototype of the N_k class.

$$C_k = \frac{1}{|S_k|} \sum_{x \in S_k} f_\theta(x) \tag{1}$$

By computing a distribution over classes, the prototypical network classifies the queries using a softmax function over the distances to the prototypes following Equation (2). Snell et al. [23] highlighted the significance of using a squared Euclidean distance as a distance function in image classification tasks. In this paper, we further studied different distancing functions for noisy time-series classification tasks.

$$P_\theta(y = k|x) = \frac{exp(-d(f_\theta(x), C_k))}{\sum_{k'} exp(-d(f_\theta(x), C_{k'}))} \tag{2}$$

Finally, the network is optimized by minimizing the negative log-probability of the true class with an Adam optimizer [33] and updating the parameters θ of the embedding function f using the loss Equation (3).

$$L \leftarrow L + \frac{1}{N_C N_Q}\left[d(f_\theta(x), C_k) + log \sum_{k'} exp(-d(f_\theta(x), C_k)\right] \tag{3}$$

2.2. Distance Metrics

L2 Euclidean: Given two vector points U: $(u_1, \ldots, u_k)$ and V: $(v_1, \ldots, v_k)$, the Euclidean L2 distance is defined as the shortest distance between two vector points, a commonly used similarity metric in various applications.

$$D(U, V) = \sqrt{\sum_{i=1}^{k} (u_i - v_i)^2} \tag{4}$$

DTW distance: DTW or dynamic time warping [34] was coined as a distance metric to find the similarities between two time sequences. Unlike the Euclidean distance, which is prone to both global and local shifts in time dimension, DTW tackles such unintuitive results and aims at finding the minimum warp path between two time sequences. Given two time sequences P and Q and their individual lengths $|P|$ and $|Q|$, respectively, DTW constructs a warp path which is given by

$$W = w_1, w_2, .., w_k, \text{ where } w_k = (i, j) \text{ and } w_k + 1 = (i^*, j^*) \tag{5}$$

The warp path begins at index (1,1) and ends at (|P|,|Q|). The optimal warp path $Dist(W)$ is thereby given by the sum of the distances between the individual warp paths from index i in P to index j in Q, meaning

$$Dist(W) = \sum_{k=1}^{k=K} Dist(w_{ki}, w_{kj}) \tag{6}$$

To reduce the time complexity of DTW from $O(N^2)$ to $O(N)$, FastDTW has been proposed in the state-of-the-art literature [34]. Keeping the whole DTW algorithm similar, it introduces three constraints: coarsening (shrinking the time sequence into smaller time steps), projection (calculating the minimum warp distance at low resolution), and refinement (refining the low-resolution warp path through local adjustments) to reduce the time complexity.

Cosine distance: The cosine distance is another metric that is used to measure the similarity between two vector points. It measures the cosine of the angle between two vector points. The cosine similarity metric and cosine distance metric are correlated and can be found in the following equations.

$$Cosine\ similarity(U, V) = cos(\theta) = \frac{U.V}{\|U\|\|V\|} = \frac{\sum_{i=1}^{k} u_i v_i}{\sqrt{\sum_{i=1}^{k} u_i^2}\sqrt{\sum_{i=1}^{k} v_i^2}} \tag{7}$$

$$D(U,V) = 1 - Cosine\ similarity(U,V)$$

3. Method

In this paper, we propose a generic feature learning method for monitoring machining processes using limited TS data annotations. In the next sections, the mixture-based data selection method is defined, followed by the proposed two-stage feature learning method.

3.1. Mixture-Based Data Selection

The learning performance depends mainly on the input data. This makes data selection not only the first but also a crucial step in FSL since it aims at choosing the training query and support sets. In computer vision, sample selection is straightforward given the standardized format of the image data. However, TS data, such as process vibration data, is characterized by the variation of signal length due to the different measurement lengths. This leads to an imbalance of data after data windowing and degrades the learning performance. In this work, we used a mixture-based data selection technique (MDS), which is illustrated in Figure 1.

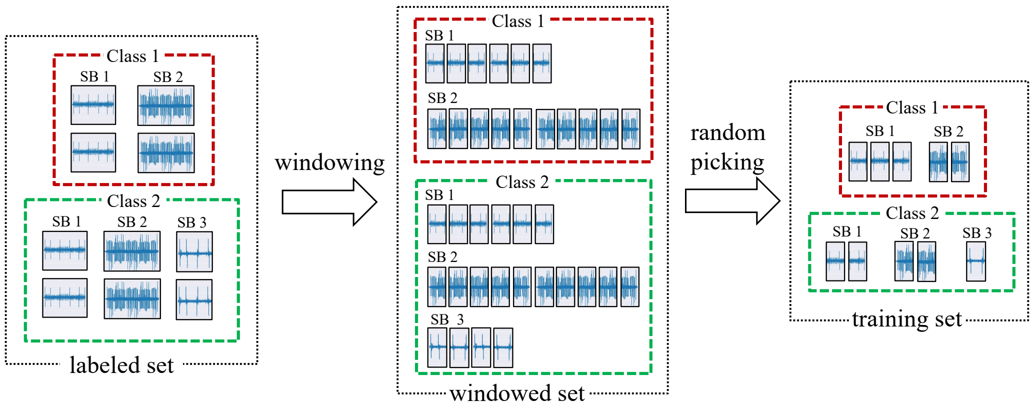


Figure 1. Mixture-based data selection method used in a 2-way and 5-shot FSL for single-axis vibration signals (C = 1). SB: subclass

The data signals $x \in \mathcal{D}$ in each class N_k are first windowed using a sliding window with a fixed w_s. In online industrial applications, data are buffered in chunks, which explains the use of the sliding window when developing industrial data processing techniques and speeds up the computing by avoiding additional data analysis steps. The output of the first step is a set of same-shaped signals $x_w \in \mathbb{R}^{w_s \times C}$, where C is the number of channels. In the case of the vibration data used in this work, C was equal to 3 with reference to the $\{X, Y, Z\}$ axes. For the sake of simplicity, only one axis of the vibration signal (C = 1) is shown in the MDS illustration in Figure 1. The windowing step is followed by a random selection step that samples the training sets, i.e., the query and support sets, during the episodic learning process. For an FSL task with N ways and K shots, the MDS outputs support set $\mathcal{S} = (x, y)^{N \times K}$ and query set $\mathcal{Q} = (x, y)^{N_Q}$ with N_Q being the number of queries per iteration. As stated above, the measurement length mismatch leads to an imbalance between the different subclasses, i.e., the different machining processes. The MDS sampling technique produces an equal number of data samples in the OK and NOK training sets at each training episode, which reduces the negative impact of the imbalance rate and results in more unbiased models. The second advantage of the MDS method is the high informativeness of the training sets in terms of the diversity of signals in each class. In fact, at each training episode, thanks to the windowing step followed by a random selection, the MDS leads to a more diverse selection of samples from different periods, machines, and processing operations, which allows the FSL models to be drift invariant and facilitates the search for discrepancy between the OK and NOK classes. This result increases the robustness of the feature extractor, which is insensitive to the challenging conditions in machining applications.

3.2. *Two-Stage Learning Framework*

The proposed method represents a two-stage learning framework for noisy industrial TS data and is shown in Figure 2. The first stage consists of an unsupervised pretraining stage, while the second step consists of fine-tuning the learned feature extractor using very limited annotations and is therefore referred to as the metalearning stage.

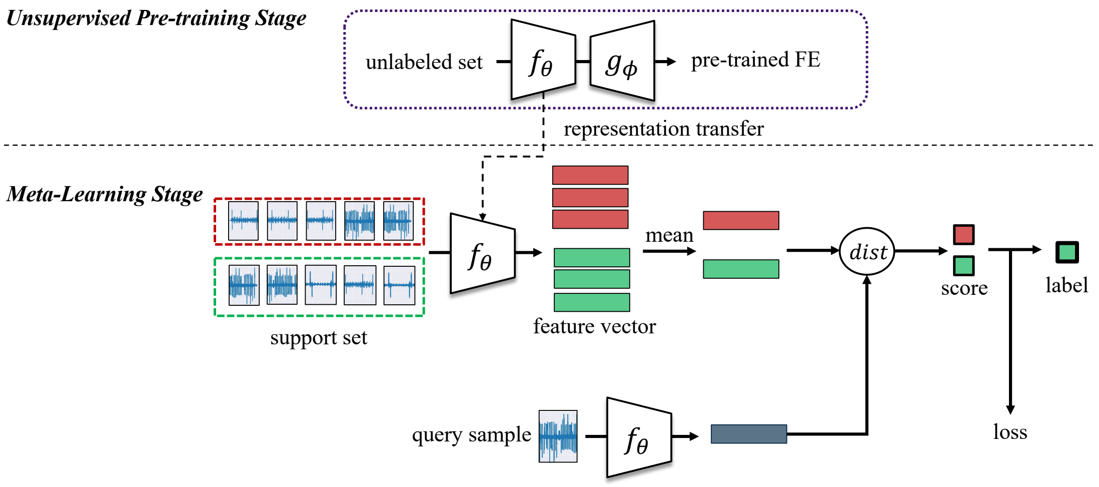


Figure 2. Two-stage few-shot feature learning framework. The OK and NOK classes are shown in green and red respectively.

3.2.1. First Stage: Unsupervised Pretraining

Industrial use-cases are characterized by their large volume of unlabeled data, in particular for time series data. In order to take advantage of the unannotated data and overcome the imbalance effect on supervised learning, the two-stage learning starts with an unsupervised feature learning using the autoencoder (AE) method [12]. In this phase, the encoder f with parameters θ learns the representation of the unlabeled dataset $\mathcal{D}_{unlabeled}$ by encoding the input signal x into a compressed vector x_{enc}. The encoder architecture, which represents the deep feature extractor of the proposed method, was designed based on a convolutional neural network (CNN) and is illustrated in Figure 3. To best evaluate the two-stage learning method, a simple stacked CNN was chosen with 3 consecutive convolutional blocks followed by a final bottleneck layer. Each convolutional block consisted of a 1-D convolutional layer, a batch normalization layer [35], a ReLu (Rectified Linear Unit) activation function, and a max pooling layer.

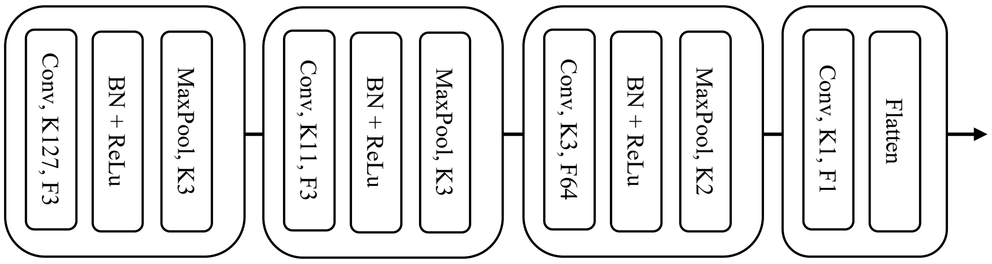


Figure 3. Architecture of the deep feature extractor f_θ. MaxPool: maxpooling layer; Conv: 1D convolutional layer; BN: batch-normalization; K: kernel size; F: filter channels

The decoder g_ϕ is a transposition of the encoder f_θ and performs the decoding of the encoded feature vector x_{enc} into the reconstructed signal x_{rec}. The objective function of the autoencoder E is the mean square error (MSE) between x_{rec} and the input signal x according to the following equation:

$$E = \frac{1}{n}\sum_{i=1}^{n}\left[x - g_\phi(f_\theta(x))\right]^2 \tag{8}$$

The result of this phase consists of the pretrained parameters of the encoder function $f_{\theta_{pretrained}}$ and the decoder part is dropped. The training process of the first stage follows the pseudocode in Algorithm 1.

Algorithm 1 First stage: unsupervised pretraining

```
Input: Unlabeled data set $\mathcal{D}_{unlabeled}$
Output: Pretrained encoder function $f_{\theta_{pretrained}}$
  $\theta, \phi \leftarrow$ Initialize randomly
  for number of epochs do
      compute MSE error $E$ using Equation (8)
      $\theta, \phi \leftarrow$ Update using gradients of $E$        ▷ compute backpropagation
  end for
```

3.2.2. Second Stage: Metric Meta Learning Stage

The second stage consists of fine-tuning the unsupervised pretrained feature extractor f_θ for a specific task using very limited annotated dataset $\mathcal{D}_{labeled}$ in an episodic manner. The first step consists of sampling the training steps using the Section 3.1 method resulting in highly informative support sets. For each signal in the support set, the embedded vector is extracted using the feature extractor f_θ and these deep feature vectors are then averaged by class. This results in N representative C_k prototypes for each class. Using a distancing function, each prototype is then matched against each embedded query point, which is classified by simply finding the closest class prototype. The distancing function is crucial to the feature learning process as it defines the loss function L (3) and therefore the optimization of the feature extractor parameters. To find the optimal distance function for the noisy vibration data, we evaluated in the experimental section different TS measures (Euclidean, cosine, and DTW). The parameters θ are later updated using the gradients of the loss function L using the Adam optimizer function [33]. Once the metric metalearning stage is completed, the resulting feature extractor f_θ is evaluated on an unseen dataset $\mathcal{D}_{test}$ and on the visualization of the embedding space of vibration data. The training process of the second stage follows the pseudocode in Algorithm 2.

Algorithm 2 Second stage: metric-based fine-tuning

```
Input: Labeled data set $\mathcal{D}_{labeled}$, pretrained encoder function $f_{\theta_{pretrained}}$
Output: Two-staged trained encoder function $f_\theta$
  $f_\theta \leftarrow f_{\theta_{pretrained}}$          ▷ initialize encoder with the pretrained parameters
  $L \leftarrow 0$
  for number of epochs do
      Sample $S_Q$ and $S_S$ from $\mathcal{D}_{labeled}$ using the Section 3.1 method
      Generate prototypes $C_S$ using the averaging Equation (1)
      Calculate $L$ for the minibatches using the loss Equation (3)
      $\theta \leftarrow$ Update using gradients of $L$          ▷ compute backpropagation
  end for
```

4. Real-World Case Study

4.1. Data Description

CNC milling machines are widely used in a variety of machining industries, commonly known for their precision and high production speed. The dataset in consideration offers a great insight into the complexity and challenges of the CNC machine monitoring use case as it closely represents a real-world industrial case inside a production plant. This work used a publicly available dataset [36] comprising sensor data recorded with the help of a triaxial accelerometer mounted on top of the machining parts of the CNC machine. The data collection was stretched over four different periods of five months each starting from February 2019 to February 2021. Such collection procedures help to tackle the challenges of data drift and the generalization of data-driven approaches. The training and the test dataset host both the normal and abnormal vibration data samples caused by the tool misalignment. Typical process operations that are being carried out by a machining workpiece greatly vary from drilling to cutting. In the scope of this work, each machine hosted 15 different process operations carried out with different physical tools and under a unique configuration. Each sample was a triaxial (X-, Y-, Z-) acceleration data acquired with a sampling rate of 2 kHz. The data were collected from three different CNC machines (M01, M02, and M03) in contention, each containing 15 different process operations ranging from OP00 to OP14. Each data sample was accompanied with the necessary labeling parameters, such as Label, Machine, and Period.

4.2. Data Splitting

This section describes the data splitting used in this work. The data were mainly divided into 2 unique sets: training set and test set. The training dataset contained 172 different samples with 156 OK samples and 16 NOK samples, while the test dataset contained 1702 different samples with 1632 OK samples and 70 NOK samples. This reflected the real-world scenario where we generally have a limited labeled data set (training set) with an imbalanced OK/NOK ratio and a relatively large number of unlabeled data (test set). This is illustrated in Figure 4 where the color "violet" represents the training set samples, whereas "orange" denotes the test set samples. To assess the generalization to unseen data and the robustness of the models to data drift, the data splitting was performed according to three different criteria:

- Machine-wise: This allowed the evaluation of the scalability of the models across different machines. We had 3 CNC Machines in consideration (M01, M02, and M03). Even though, they generated data samples representing the same tool process operations, they varied due to external conditions. Both the training and the test sets were uniformly distributed across the three machines as shown in Figure 4. With a uniform distribution, the model was not offered any unnecessary bias across a particular machine. M03 was not included in the training and was placed aside for testing.
- Process-wise: This allowed the evaluation of the generalization of the models across unseen tool processes. In industrial applications, new processes are constantly being added due to technological progress and market demand. The training set only

contained 4 different tool operations whereas 11 new tool process operations were introduced in the test set.

- Period-wise: This allowed the evaluation of the robustness of the models across unseen periods. Worn components and aging cause a drift in the data, which affects the data-driven models. For that purpose, the periods of August 2020 and August 2021 were not included in the training and were placed aside for testing.

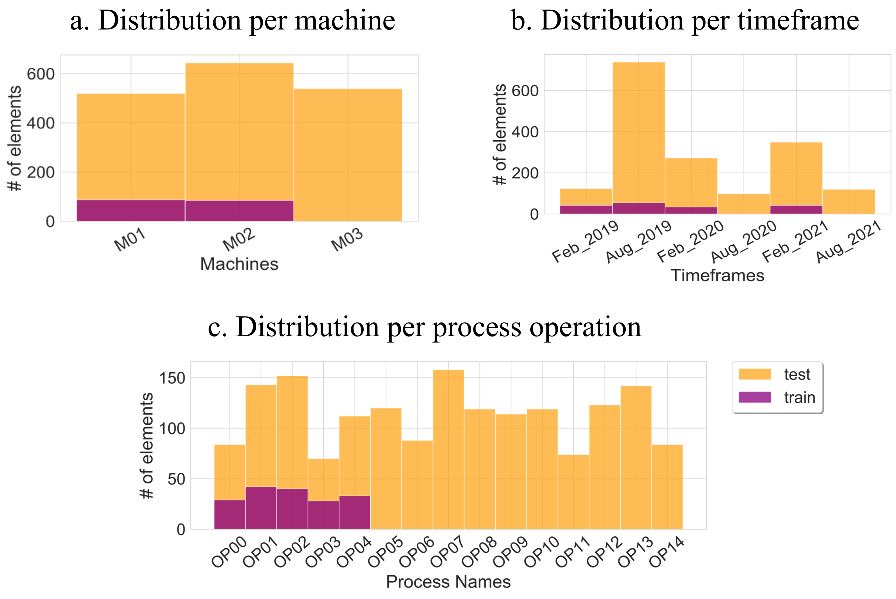


Figure 4. Distribution of the training and test datasets per machine, timeframe, and machining process. To reflect the challenges of industrial practice, a limited amount of data is included in the training set (violet) and unseen data from different machines, time periods and process operations is included in the test set (orange).

5. Experiments and Analysis

The following section describes the experiments carried out in the scope of this work. The goal of this work was to investigate different strategies for training feature extractors (FEs) for raw industrial time-series data and evaluate them in terms of robustness and generalization. The training of the FE was conducted in a progressive manner. We started by evaluating the performance of the single-stage prototypical network. Once the best parameters were obtained, we proceeded to a comparison with the proposed two-stage model framework. The trained FE models were evaluated on unseen data samples from the test set. We concluded by comparing the FE model obtained by the proposed method with the handcrafted FE and the end-to-end supervised trained FE using a distribution analysis coupled with a feature space analysis. All the experiments were performed under similar conditions with identical training parameters (learning rate = 8×10^{-4}, number of epochs = 4, window size w_s = 4096 and optimizer = Adam). N was fixed to 2 as we only considered two distinct classes for our experiments {Class 1: OK, Class 2: NOK}. While training, the data samples from OP00 to OP04 were separated into the OK and the NOK class sets. During each episode, we randomly picked a number K of data samples from these two classes to create the support and query set using the MDS method. The value of K representing the number of shots during each episode was varied to determine its effect on the performance of the model. To test generalizability, the models were evaluated using sample data from all available machining operations (OP00 to OP15). The sample data were then picked following the same way as for the training set. The experiments were conducted three times and averaged over their sum to produce the final results. The PN FE models were thereafter evaluated for 1000 episodes of four epochs. The models were trained on a GPU NVIDIA Tesla K80 and generated in Python (version 3.7.4) using the PyTorch library (version 1.8.1).

5.1. Single-Stage Prototypical Network

Experiment: The single stage prototypical network proceeded with a vanilla implementation of FSL for process failure on the industrial vibration data. The first phase of the experiments used a PN technique with a randomly initialized encoder f_θ with the architecture presented in Figure 3. This experiment was designed to vary two distinctive parameters: K , the number of shots, and *dist*, the distance metric. First, K was varied between 1 and 10 shots and the *dist* was set to the Euclidean distance. Second, *dist* was varied between Euclidean, DTW, and cosine and K was set to seven. Combined with the MDS sampling technique, we focus on obtaining the best set of prototypical learning parameters for industrial vibration data.

Results: Tables 1 and 2 list all the results from the experiments that are compared using different metrics such as "train loss", "test loss", "train Accuracy", "accuracy" (test set), "F1-score" (test set), "precision" (test set), "recall" (test set). For the K -shot analysis, all the models converged with 100% accuracy on the training data. The PN model with one-shot learning had the worst F1-score of 76.70%. This is plausible, especially for machining anomaly detection applications, where we face large variations within a single class and often require more samples on the support set to produce better representations (prototypes) and thus a better generalization. The performance of the models gradually increased with the number of shots as can be seen in Table 1. The convergence of the F1-score was reached by the seven-shot PN model at the 87.3% mark. We also noted that the test loss was reduced to 22.76 with a precision score of 89.3%. Upon further increasing the number of shots to 10, we suffered a minimal deterioration of the training loss that can be explained by the drawback of the averaging function performed on the noisy time-series feature vectors. In fact, averaging a relatively large number of deep TS-type features affects the information richness of the prototype vector at some point.

Table 1. Results of the K-shots experimentation.

Model	Train Loss	Test Loss	Accuracy	F1-Score	Precision	Recall
1-shot	0.1340	87.31	0.765	0.767	0.759	0.773
3-shot	0.0304	36.95	0.848	0.847	0.856	0.835
5-shot	0.0053	29.33	0.860	0.859	0.869	0.848
7-shot	0.0048	22.76	0.876	0.873	0.893	0.855
10-shot	0.0079	21.73	0.882	0.878	0.906	0.853

Table 2. Results of the distance measures experimentation.

Distance	Train Accuracy	Accuracy	F1-Score	Precision	Recall
Euclidean	0.999	0.876	0.874	0.894	0.855
DTW	0.737	0.663	0.535	0.865	0.387
Cosine	1.000	0.842	0.851	0.803	0.905

Table 2 compares the results achieved with different distance metrics. With an F1-score just below 54% and a training accuracy of only 66.3%, the DTW-based PN failed to learn. One assumption why the DTW technique failed can be due to the failure to find the best alignment between the prototype vectors and the query vector due to the cyclic behavior of the data. The Euclidean distance, on the other hand, gave the best results, followed by the cosine distance metric, the former getting an F1-score of 87.6% (2.3% higher). This confirmed the findings from Snell et al. [23] for image classification tasks. However, the cosine-based PN offered a better recall (90.5%) over the Euclidean distance recall (85.5%) meaning that it was more reliable in detecting the faulty processes but returned more false positives. This is usually important for industrial applications where quality checks are crucial and demand to be accurate in detecting anomalies, thus prioritizing detecting faulty parts rather than accurately detecting all the good parts.

5.2. Two-Stage Prototypical Network

Experiment: Using the Euclidean distance and the K equal to seven shots, we evaluated and compared the two-stage proposed FE learning framework with the conventional single-stage learning method. Instead of randomly initializing the FE encoder f_θ, a pretraining CNN autoencoder was added as an additional layer on top of the prototypical network as stated in Section 3.2. The AE was trained on the full dataset irrespective of the splitting scenario mentioned earlier. This was justified by the fact that today, thanks to IIoT advancements, a huge quantity of unlabeled sensory data is available in the industry and could be used for unsupervised training. We considered a batch size of 32 windows each spanning over 4096 data points which were trained for 40 epochs with a learning rate of $8{\cdot}10^{-4}$.

Results: The goal of stage one consisted of pretraining the feature extractor f_θ via a CNN AE network in order to break down the complex architecture of high-dimensional sensor data. The results are shown in Figure 5 where we visualize the feature extracted by the $f_{\theta_{pretrained}}$ and the reconstructed signal using $g_{\phi_{pretrained}}$. The reached training loss value is as low as 0.2.

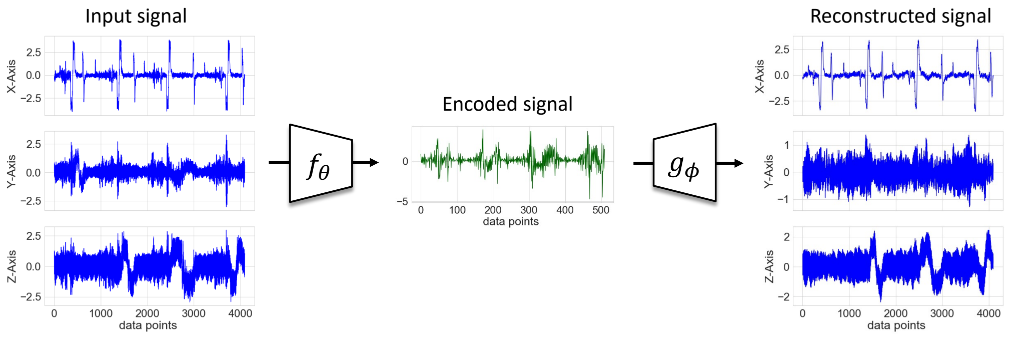


Figure 5. Performance of the resulting AE model at the end of stage one.

Upon initializing with the learned weights, the PN as part of the two-stage model shows a clear improvement over the single-stage network. It significantly tops the performance chart by achieving an F1-score of 90.3% and an accuracy of 91.0% as shown in Table 3. The two-stage model also proves to have better generalizability on unseen data samples from new class labels as the test loss is significantly decreased from 22.76 for the standard PN to 6.582. It can be further explained with the confusion matrices of both models in Figure 6. The two-stage confusion matrix shows a similar improvement with the inclusion of the pretrained network. The proposed model reaches a prediction rate among its OK samples with an accuracy of 97.62% compared to the single-stage being only at 88%. However, we see a slight deterioration in the NOK accuracy of 1.06%. This can be explained by the pretrained weights in the two-stage training, as the number of OK samples dominated the full dataset over the NOK samples, with an 816:35 imbalance rate. It created a slight bias on the OK samples. We also note a longer training time of 1298 seconds due to the unsupervised pretraining of the feature extraction. The effect of pretraining on the FE model is illustrated in Figure 7. The 2D feature map was generated using a principal component analysis (PCA) [37]. The feature maps changed over time upon increasing the number of epochs. After training four epochs with 1000 episodes each, a clear distinction in the clusters between OK and NOK samples on the two-stage model can be seen in Figure 7, in contrast to the single-stage model where a large number of false positives are observed (in the $PC1 > 1$ range). That confirms the results of the confusion matrices from Figure 6.

Table 3. Evaluation of the proposed method against the single-stage method. The training was run on a GPU NVIDIA Tesla K80.

Model	Test Loss	Accuracy	F1-Score	Precision	Recall	Training Time (s)
Single-Stage	22.76	0.876	0.873	0.893	0.855	295.91
Two-Stage	6.582	0.910	0.903	0.973	0.843	1295.68

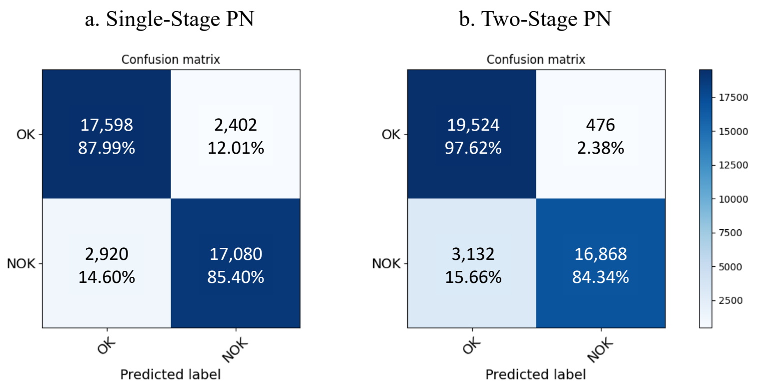


Figure 6. Comparison of the confusion matrices of the single-stage and two-stage learning methods.

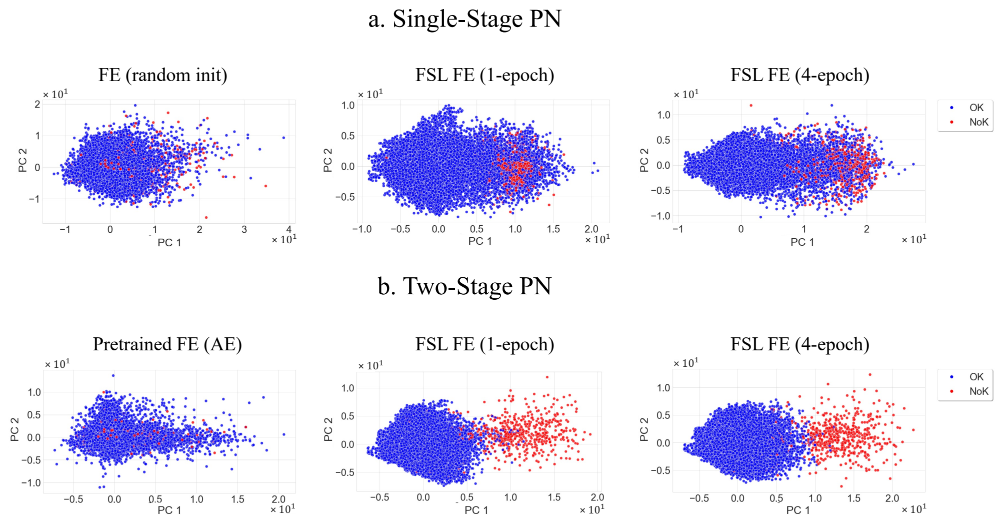


Figure 7. Comparison of the PCA feature spaces of the single-stage and two-stage PN on the pretraining, 1-epoch, and 4-epoch levels. The encoder of the single-stage PN was not pretrained and its parameters were therefore initialized randomly.

5.3. *Handcrafted vs. Supervised vs. Two-Stage FE*

Experiment: This section provides a detailed comparison of the proposed two-stage model with the handcrafted features and with a feature extractor trained using the traditional end-to-end supervised method. The handcrafted features were extracted using TSFRESH, a state-of-the-art handcrafted feature learning algorithm for industrial time-series data. The supervised method consists of building a classifier block on the top of the feature extraction block presented in Figure 3 and training the network in a conventional end-to-end manner. The classifier block consists of two sequential fully connected layers and a sigmoid as activation function. For the proposed and handcrafted methods, the experiments consisted of training the classifier NN separately using the features extracted with the two-stage FE and TSFRESH, respectively. All the experiments were performed under similar conditions with identical training parameters (epochs: 8, learning rate: $8{\cdot}10^{-4}$, batch size: 32, optimizer: Adam, loss function: binary cross-entropy). The end-to-end supervised method uses the state-of-the-art weight-balancing factor.

Results: Table 4 and Figure 8 lay out the results of each of the three methods. The supervised learning delivers the lowest performance among the other methods. Therefore, the F1-score of predicting the correct class only stands at 5.6%. This can be explained by the fact that conventional supervised training requires a huge quantity of labeled data and fails to learn using limited quantity of data. Table 4 shows that features extracted using the two-stage FE outperform the handcrafted FE method with an accuracy of 98.9% (vs. 86.6%) and an F1-score of 88.4% (vs. 84.8%). This performance further highlights the high precision of the proposed method (99.55%) with the classification of the OK class that is shown in the confusion matrix in Figure 8. This confirms the efficiency of the unsupervised pretraining phase where the model learns reliably the dynamic representations of the vibration data and turns more robust against data drift caused by time and wear of the machining components. This can be seen in Figure 9, where the drift across machines and across timeframes is visualized.

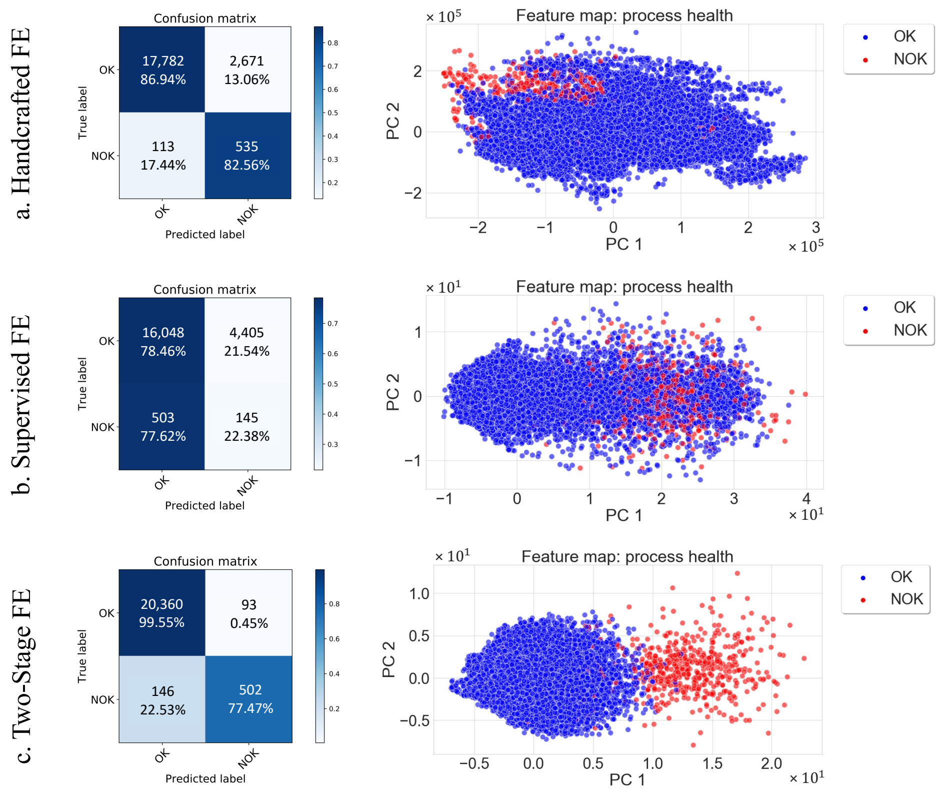


Figure 8. Comparison of the handcrafted (**a**), supervised (**b**), and the proposed method (**c**) trained feature extractors: the top row shows the confusion matrices obtained by the MLP classification, and the bottom row shows the 2D visualization of the FEs' feature spaces.

Table 4. Evaluation of the proposed method against handcrafted and supervised trained feature extractors. The extraction was performed on a CPU Intel Core i7 9850H.

Feature Extractor	Accuracy	F1-Score	Precision	Recall	Training Time per Window (s)
Handcrafted	0.868	0.845	0.862	0.848	2.2502
Supervised	0.767	0.056	0.200	0.032	0.0054
Two-Stage	0.989	0.884	0.905	0.885	0.0054

The features extracted from the OK class of the exact same process operations using the handcrafted method vary from one machine to another and also over time (when considering the same machine). In contrast, Figure 9 shows the robustness of the two-stage FE, where the OK class data points have the same distribution, with no drift for the across-machine and across-time analysis. We note also that this holds true for the process operations not seen during training (OP06, OP07, and OP12 in Figure 8, as well as for the timeframe (Feb_2020) and the machine (M3) not included in the training set. This result is supported by Table 5, which presents the quantitative analysis of the drift between machines and over time based on the handcrafted features and the deep features extracted by the proposed method. The drift between the U and V domains was measured using the Wasserstein distance. The two-stage FE shows excellent robustness to drift across the seen and unseen domains within the OK class. We also see a larger distance between the OK and NOK classes, which is consistent with the results from Figure 8.

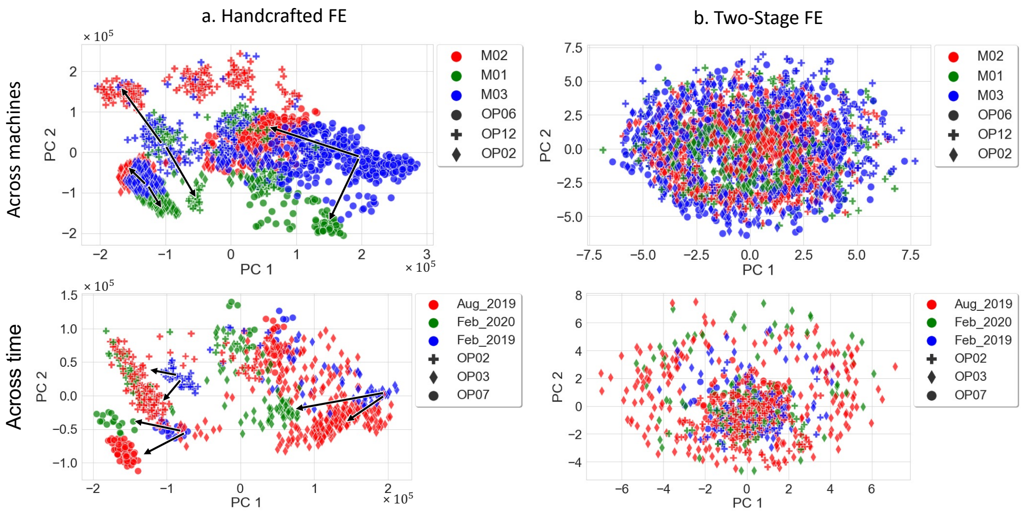


Figure 9. Evaluation of the data drift across machines and time of the handcrafted FE and the proposed method FE within the OK class.

On the other hand, the handcrafted FE provides less robustness as the distance between the OK domains is not consistent and in some cases, even higher than the distance between the OK and NOK domains. In fact, the OK–NOK Wasserstein distance is equal to 35,869, which is more or less equal to the distances: (M2, M3), (M1, M2), (August 2019, February 2019), and (February 2020, February 2019). A further analysis of Figure 8 reveals the superiority of the proposed two-stage method in OK/NOK separation in the feature space generated by the first two principal components. The two-stage method in Figure 8 shows a clear separation of the normal and abnormal classes compared to the handcrafted and supervised FE methods. It is also important to note that the handcrafted FE has slightly better NOK accuracy, which can be seen in the confusion matrices, with 82.56% compared to 77.47% (two-stage FE). However, the major drawback of the handcrafted technique is the high extraction time (2.2502 s/window) compared to the deep feature learning techniques (0.0054 s/window). This is an important feature for industrial applications that require real-time execution when dealing with real-world use cases.

Table 5. Quantitative drift analysis of the handcrafted and two-stage trained feature extractors across time and machines using the Wasserstein distance between the different domains (U and V).

(**a**) Handcrafted FE

	Within only OK class						
U= **V=**	**Aug_2019** **Feb_2019**	**Feb_2020** **Feb_2019**	**Aug_2019** **Feb_2020**	**M1** **M2**	**M1** **M3**	**M2** **M3**	**OK** **NOK**
	37,680.36	36,229.76	6435.13	37,919.52	14,147.56	37,602.05	35,868.79

(**b**) Two-Stage FE

	Within only OK class						
U= **V=**	**Aug_2019** **Feb_2019**	**Feb_2020** **Feb_2019**	**Aug_2019** **Feb_2020**	**M1** **M2**	**M1** **M3**	**M2** **M3**	**OK** **NOK**
	0.484	0.529	0.146	0.337	0.513	1.001	6.22

6. Conclusions

In the field of machine condition monitoring, industrial time-series data face major challenges, such as class imbalance, data drift, and most importantly, the lack of pretrained feature extractors. To overcome these challenges, we proposed an efficient two-stage feature learning approach. The proposed technique bridged the gap between unsupervised learning and few-shot learning, which makes it suitable for the industrial scenario where a large quantity of sensory data is available with a limited number of labels. Intuitively adding an autoencoder to a prototype network has proven to be effective. Through a rigorous experimentation and analysis process, we showed that initializing the network with pretrained weights enabled the FE network to upgrade its learning performance. The two-stage learning method produced a feature extractor with higher generalization capabilities compared to the traditional prototypical network, achieving an F1-score of 90.3% with very limited samples. However, it had the disadvantage of a longer training time and a slight decrease in the recall score, while significantly improving the precision score. The research experiments conducted with the traditional prototypical network showed that Euclidean and cosine distance performed best on noisy industrial data, with the Euclidean distance being the best choice in terms of accuracy and the cosine distance in terms of recall. This makes the cosine a better choice for critical quality-testing applications. Finally, the proposed method slightly outperformed the traditional handcrafted feature extractor with an improvement of 4% in the F1-score. Although handcrafted features have the potential to match the performance of the proposed two-stage learning method in terms of classification performance, they have a disadvantage in terms of computation time and robustness to drift. However, this opens the door for future research on hybrid solutions combining handcrafted and deep features. Indeed, extracting handcrafted values from deep features would reduce computation time since it creates a compression of the raw data with the most informative patterns.

Author Contributions: Conceptualization, M.-A.T.; methodology, M.-A.T.; software, M.-A.T. and P.S.; validation, M.-A.T. and P.S.; formal analysis, M.-A.T. and P.S.; investigation, M.-A.T.; resources, M.-A.T. and P.S.; data curation, M.-A.T. and P.S.; writing—original draft preparation, M.-A.T. and P.S.; writing—review and editing, M.-A.T., P.S., and K.D.; visualization, M.-A.T.; supervision, K.D.; project administration, M.-A.T.; funding acquisition, M.-A.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that the publication of data must be approved by the Bosch Rexroth AG. The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AE	Autoencoder
CNC	Computer numerical control
DL	Deep learning
DTW	Distance time warping
FE	Feature extractor
FSL	Few-shot learning
IIoT	Industrial Internet of things
MDS	Mixture-based data selection
MSE	Mean square error
NN	Neural network
PCA	Principle component analysis
PN	Prototypical network
TS	Time series

References

1. Mohamed, A.; Hassan, M.; M'Saoubi, R.; Attia, H. Tool Condition Monitoring for High-Performance Machining Systems—A Review. *Sensors* **2022**, *22*, 2206. [CrossRef]
2. Iliyas Ahmad, M.; Yusof, Y.; Daud, M.E.; Latiff, K.; Abdul Kadir, A.Z.; Saif, Y. Machine monitoring system: A decade in review. *Int. J. Adv. Manuf. Technol.* **2020**, *108*, 3645–3659. [CrossRef]
3. Bostrom, A.; Bagnall, A. Binary shapelet transform for multiclass time series classification. In Proceedings of the International Conference on Big Data Analytics and Knowledge Discovery, Valencia, Spain, 1–4 September 2015; pp. 257–269.
4. Kate, R.J. Using dynamic time warping distances as features for improved time series classification. *Data Min. Knowl. Discov.* **2016**, *30*, 283–312. [CrossRef]
5. Zhang, X.; Zhang, Q.; Chen, M.; Sun, Y.; Qin, X.; Li, H. A two-stage feature selection and intelligent fault diagnosis method for rotating machinery using hybrid filter and wrapper method. *Neurocomputing* **2018**, *275*, 2426–2439. [CrossRef]
6. Yiakopoulos, C.; Gryllias, K.C.; Antoniadis, I.A. Rolling element bearing fault detection in industrial environments based on a K-means clustering approach. *Expert Syst. Appl.* **2011**, *38*, 2888–2911. [CrossRef]
7. Christ, M.; Braun, N.; Neuffer, J.; Kempa-Liehr, A.W. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing* **2018**, *307*, 72–77. [CrossRef]
8. Lines, J.; Taylor, S.; Bagnall, A. Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. *ACM Trans. Knowl. Discov. Data* **2018**, *12*. [CrossRef]
9. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [CrossRef]
10. Zhang, W.; Peng, G.; Li, C.; Chen, Y.; Zhang, Z. A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals. *Sensors* **2017**, *17*, 425. [CrossRef]
11. Ismail Fawaz, H.; Lucas, B.; Forestier, G.; Pelletier, C.; Schmidt, D.F.; Weber, J.; Webb, G.I.; Idoumghar, L.; Muller, P.A.; Petitjean, F. Inceptiontime: Finding alexnet for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 1936–1962. [CrossRef]
12. Kramer, M.A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.* **1991**, *37*, 233–243. [CrossRef]
13. Sun, W.; Shao, S.; Zhao, R.; Yan, R.; Zhang, X.; Chen, X. A sparse auto-encoder-based deep neural network approach for induction motor faults classification. *Measurement* **2016**, *89*, 171–178. [CrossRef]
14. Shao, H.; Jiang, H.; Lin, Y.; Li, X. A novel method for intelligent fault diagnosis of rolling bearings using ensemble deep auto-encoders. *Mech. Syst. Signal Process.* **2018**, *102*, 278–297. [CrossRef]
15. Cabrera, D.; Sancho, F.; Long, J.; Sánchez, R.V.; Zhang, S.; Cerrada, M.; Li, C. Generative adversarial networks selection approach for extremely imbalanced fault diagnosis of reciprocating machinery. *IEEE Access* **2019**, *7*, 70643–70653. [CrossRef]
16. Zareapoor, M.; Shamsolmoali, P.; Yang, J. Oversampling adversarial network for class-imbalanced fault diagnosis. *Mech. Syst. Signal Process.* **2021**, *149*, 107175. [CrossRef]
17. Wang, Y.; Yao, Q.; Kwok, J.T.; Ni, L.M. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–34. [CrossRef]
18. Chen, W.Y.; Liu, Y.C.; Kira, Z.; Wang, Y.C.F.; Huang, J.B. A closer look at few-shot classification. *arXiv* **2019**, arXiv:1904.04232.

19. Wang, D.; Zhang, M.; Xu, Y.; Lu, W.; Yang, J.; Zhang, T. Metric-based meta-learning model for few-shot fault diagnosis under multiple limited data conditions. *Mech. Syst. Signal Process.* **2021**, *155*, 107510. [CrossRef]
20. Zhang, A.; Li, S.; Cui, Y.; Yang, W.; Dong, R.; Hu, J. Limited data rolling bearing fault diagnosis with few-shot learning. *IEEE Access* **2019**, *7*, 110895–110904. [CrossRef]
21. Wang, S.; Wang, D.; Kong, D.; Wang, J.; Li, W.; Zhou, S. Few-shot rolling bearing fault diagnosis with metric-based meta learning. *Sensors* **2020**, *20*, 6437. [CrossRef]
22. Sun, Q.; Liu, Y.; Chua, T.S.; Schiele, B. Meta-transfer learning for few-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 403–412.
23. Snell, J.; Swersky, K.; Zemel, R. Prototypical networks for few-shot learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4080–4090.
24. Sun, J.; Takeuchi, S.; Yamasaki, I. Prototypical Inception Network with Cross Branch Attention for Time Series Classification. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China,18–23 June 2021; pp. 1–7.
25. Tang, W.; Liu, L.; Long, G. Interpretable time-series classification on few-shot samples. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
26. Jiang, C.; Chen, H.; Xu, Q.; Wang, X. Few-shot fault diagnosis of rotating machinery with two-branch prototypical networks. *J. Intell. Manuf.* **2022**, *33*, 1–15. [CrossRef]
27. Yu, K.; Lin, T.R.; Tan, J. A bearing fault and severity diagnostic technique using adaptive deep belief networks and Dempster–Shafer theory. *Struct. Health Monit.* **2020**, *19*, 240–261. [CrossRef]
28. Xu, H.; Ma, R.; Yan, L.; Ma, Z. Two-stage prediction of machinery fault trend based on deep learning for time series analysis. *Digit. Signal Process.* **2021**, *117*, 103150. [CrossRef]
29. Das, D.; Lee, C.G. A two-stage approach to few-shot learning for image recognition. *IEEE Trans. Image Process.* **2019**, *29*, 3336–3350. [CrossRef] [PubMed]
30. Afrasiyabi, A.; Lalonde, J.F.; Gagné, C. Mixture-based feature space learning for few-shot image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canad, 11–17 October 2021; pp. 9041–9051.
31. Ma, J.; Xie, H.; Han, G.; Chang, S.F.; Galstyan, A.; Abd-Almageed, W. Partner-assisted learning for few-shot image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 10573–10582.
32. Afrasiabi, S.; Afrasiabi, M.; Parang, B.; Mohammadi, M.; Kahourzade, S.; Mahmoudi, A. Two-stage deep learning-based wind turbine condition monitoring using SCADA data. In Proceedings of the 2020 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES), Jaipur, India, 16–19 December 2020; pp. 1–6.
33. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
34. Salvador, S.; Chan, P. Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intell. Data Anal.* **2004**, *11*, 561–580. [CrossRef]
35. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
36. Tnani, M.A.; Feil, M.; Diepold, K. Smart Data Collection System for Brownfield CNC Milling Machines: A New Benchmark Dataset for Data-Driven Machine Monitoring. *Procedia CIRP* **2022**, *107*, 131–136. [CrossRef]
37. Wold, S.; Esbensen, K.; Geladi, P. Principal component analysis. *Chemom. Intell. Lab. Syst.* **1987**, *2*, 37–52. [CrossRef]

6 Conclusion & Future Work

Time-series-based machine learning techniques encounter significant challenges in real-world industrial applications. Using a smart data collection system, this work investigated and embedded these challenges in a novel benchmark CNC machining dataset that models the real-world scenario. The challenges mainly arise from the data drift and the wide variety of process operations in industry. In addition to data variance, data-driven models suffer from the high imbalance of OK/NOK classes and require a high level of expertise for data labeling and feature extraction, limiting the scalability of state-of-the-art machine monitoring techniques.

Addressing the challenge of data labeling in industry, unsupervised deep learning methods are investigated, namely autoencoders. Multi-stage feature learning and sequential small filters have resulted in a lightweight, more efficient architecture on noisy vibration data. Rigorous examination of the various architectures has shown that the wide network architectures outperform the deep networks when dealing with noisy time-series data. Unsupervised methods, however, do have feature learning limitations but remain a viable method for data compression and encoding.

To overcome the shortcomings of unsupervised methods, a two-stage learning framework was proposed. By combining autoencoder learning and prototypical few-shot learning, the resulting method outperform handcrafted feature extractors and conventional supervised fine-tuning methods. With a very limited amount of labeled data, the proposed feature extractor achieves an F1 score of 90.3%. This confirms the superiority of semi-supervised methods over unsupervised methods and the necessity of data annotation to accurately identify patterns. With an F1 score below perfection, a fully autonomous deep learning-based solution is still unrealistic, proving that machine learning is still a supporting rather than a supervising tool.

This work opens up doors for further exploration in the field of hybrid feature learning, where the combination of learned and handcrafted features, using for example weak supervision techniques, can be investigated. By leveraging weak supervision, researchers can further improve the performance of the two-stage learning method and explore the synergistic effects of combining different types of features. This approach would enable the indirect incorporation of human expertise in signal processing into the learning mechanism, leading to more robust and interpretable models. Additionally, future research focuses on enhancing the explainability and interpretability of the learned features. While visualizing the embedding space is a valuable step, it is not sufficient to enable operators and domain experts to fully comprehend the decision-making process of the models. In fact, building trust in the system is crucial for successful deployment in industrial environment. Integrating human-in-the-loop approaches, which allow human operators to provide feedback and override model decisionss, would not only improve feature learning but also enhance the interpretability of the models.

Bibliography

[1] A. Agogino and K. Goebel. *BEST lab, UC Berkeley. Milling Data Set, NASA Ames Prognostics Data Repository*. `https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/`. (Date accessed: 25.05.2023). 2007.

[2] C. Aldrich and L. Auret. *Unsupervised process monitoring and fault diagnosis with machine learning methods*. Vol. 16. (3). Springer, 2013.

[3] A. Asuncion and D. Newman. *UCI machine learning repository*. 2007.

[4] Y. Bai, Z. Sun, J. Deng, L. Li, J. Long, and C. Li. “Manufacturing quality prediction using intelligent learning approaches: A comparative study”. In: *Sustainability* 10(1) (2017), p. 85.

[5] X. Bampoula, G. Siaterlis, N. Nikolakis, and K. Alexopoulos. “A deep learning model for predictive maintenance in cyber-physical production systems using lstm autoencoders”. In: *Sensors* 21(3) (2021), p. 972.

[6] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa. “TSFEL: Time series feature extraction library”. In: *SoftwareX* 11 (2020), p. 100456.

[7] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[8] D. Cabrera, F. Sancho, J. Long, R.-V. Sánchez, S. Zhang, M. Cerrada, and C. Li. “Generative adversarial networks selection approach for extremely imbalanced fault diagnosis of reciprocating machinery”. In: *IEEE Access* 7 (2019), pp. 70643–70653.

[9] J. C. Chen and W.-L. Chen. “A tool breakage detection system using an accelerometer sensor”. In: *Journal of Intelligent manufacturing* 10(2) (1999), pp. 187–197.

[10] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. “Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)”. In: *Neurocomputing* 307 (2018), pp. 72–77.

[11] S. Cuentas, R. Peñabaena-Niebles, and E. Garcia. “Support vector machine in statistical process monitoring: a methodological and analytical review”. In: *The International Journal of Advanced Manufacturing Technology* 91 (2017), pp. 485–500.

[12] D. Dimla Sr and P. Lister. “On-line metal cutting tool condition monitoring.: II: tool-state classification using multi-layer perceptron neural networks”. In: *International Journal of Machine Tools and Manufacture* 40(5) (2000), pp. 769–781.

[13] A. Dogan and D. Birant. “Machine learning and data mining in manufacturing”. In: *Expert Systems with Applications* 166 (2021), p. 114060.

[14] C. Drouillet, J. Karandikar, C. Nath, A.-C. Journeaux, M. El Mansori, and T. Kurfess. “Tool life predictions in milling using spindle power with the neural network technique”. In: *Journal of Manufacturing Processes* 22 (2016), pp. 161–168.

[15] Y. Duan, J. S. Edwards, and Y. K. Dwivedi. “Artificial intelligence for decision making in the era of Big Data–evolution, challenges and research agenda”. In: *International journal of information management* 48 (2019), pp. 63–71.

[16] H. M. Ertunc, K. A. Loparo, and H. Ocak. “Tool wear condition monitoring in drilling operations using hidden Markov models (HMMs)”. In: *International Journal of Machine Tools and Manufacture* 41(9) (2001), pp. 1363–1384.

[17] M. Frankowiak, R. Grosvenor, and P. Prickett. “A review of the evolution of microcontroller-based machine and process monitoring”. In: *International journal of machine tools and manufacture* 45(4-5) (2005), pp. 573–582.

[18] Y. Fu, Y. Zhang, H. Qiao, D. Li, H. Zhou, and J. Leopold. “Analysis of feature extracting ability for cutting state monitoring using deep belief networks”. In: *Procedia Cirp* 31 (2015), pp. 29–34.

[19] B. D. Fulcher and N. S. Jones. “hctsa: A computational framework for automated time-series phenotyping using massive feature extraction”. In: *Cell systems* 5(5) (2017), pp. 527–531.

[20] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. “A survey on concept drift adaptation”. In: *ACM computing surveys (CSUR)* 46(4) (2014), pp. 1–37.

[21] A. Giantomassi, F. Ferracuti, S. Iarlori, G. Ippoliti, and S. Longhi. “Electric motor fault detection and diagnosis by kernel density estimation and Kullback–Leibler divergence based on stator current measurements”. In: *IEEE Transactions on Industrial Electronics* 62(3) (2014), pp. 1770–1780.

[22] R. Godina and J. C. Matias. “Quality control in the context of industry 4.0”. In: *International Joint conference on Industrial Engineering and Operations Management*. Springer. 2018, pp. 177–187.

[23] A. W. Hashmi, H. S. Mali, A. Meena, I. A. Khilji, M. F. Hashmi, et al. “Machine vision for the measurement of machining parameters: A review”. In: *Materials Today: Proceedings* (2021).

[24] P. B. Huang, C.-C. Ma, and C.-H. Kuo. “A PNN self-learning tool breakage detection system in end milling operations”. In: *Applied Soft Computing* 37 (2015), pp. 114–124.

[25] Y. Huang, C.-H. Chen, and C.-J. Huang. “Motor fault detection and feature extraction using RNN-based variational autoencoder”. In: *IEEE access* 7 (2019), pp. 139086–139096.

[26] M. Iliyas Ahmad, Y. Yusof, M. E. Daud, K. Latiff, A. Z. Abdul Kadir, and Y. Saif. “Machine monitoring system: a decade in review”. In: *The International Journal of Advanced Manufacturing Technology* 108(11) (2020), pp. 3645–3659.

[27] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean. “InceptionTime: Finding AlexNet for time series classification”. In: *Data Mining and Knowledge Discovery* 34(6) (2020), pp. 1936–1962. ISSN: 1573-756X. DOI: 10.1007/s10618-020-00710-y.

[28] O. Janssens, V. Slavkovikj, B. Vervisch, K. Stockman, M. Loccufier, S. Verstockt, R. Van de Walle, and S. Van Hoecke. “Convolutional neural network based fault detection for rotating machinery”. In: *Journal of Sound and Vibration* 377 (2016), pp. 331–345.

[29] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu. “Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data”. In: *Mechanical systems and signal processing* 72 (2016), pp. 303–315.

[30] T. Kalvoda and Y.-R. Hwang. “A cutter tool monitoring in machining process using Hilbert–Huang transform”. In: *International Journal of Machine Tools and Manufacture* 50(5) (2010), pp. 495–501.

[31] E. D. Kirby and J. C. Chen. “Development of a fuzzy-nets-based surface roughness prediction system in turning operations”. In: *Computers & Industrial Engineering* 53(1) (2007), pp. 30–42.

[32] H. Kishawy, H. Hegab, U. Umer, and A. Mohany. “Application of acoustic emissions in machining processes: analysis and critical review”. In: *The International Journal of Advanced Manufacturing Technology* 98(5) (2018), pp. 1391–1407.

[33] M. Lamraoui, M. Barakat, M. Thomas, and M. E. Badaoui. “Chatter detection in milling machines by neural network classification and feature selection”. In: *Journal of Vibration and Control* 21(7) (2015), pp. 1251–1266.

[34] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. “Temporal Convolutional Networks for Action Segmentation and Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[35] W. J. Lee, G. P. Mendis, M. J. Triebe, and J. W. Sutherland. “Monitoring of a machining process using kernel principal component analysis and kernel density estimation”. In: *Journal of Intelligent Manufacturing* 31 (2020), pp. 1175–1189.

[36] Y. Lei, B. Yang, X. Jiang, F. Jia, N. Li, and A. K. Nandi. “Applications of machine learning to machine fault diagnosis: A review and roadmap”. In: *Mechanical Systems and Signal Processing* 138 (2020), p. 106587.

[37] C. Lessmeier, J. K. Kimotho, D. Zimmer, and W. Sextro. “Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification”. In: *PHM Society European Conference*. Vol. 3. (1). 2016.

[38] C. Li, R.-V. Sánchez, G. Zurita, M. Cerrada, and D. Cabrera. “Fault diagnosis for rotating machinery using vibration measurement deep statistical feature learning”. In: *Sensors* 16(6) (2016), p. 895.

[39] T. Li, Z. Zhao, C. Sun, L. Cheng, X. Chen, R. Yan, and R. X. Gao. “WaveletKernelNet: An interpretable deep neural network for industrial intelligent diagnosis”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52(4) (2021), pp. 2302–2312.

[40] X. Li, W. Zhang, Q. Ding, and J.-Q. Sun. “Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation”. In: *Journal of Intelligent Manufacturing* 31 (2020), pp. 433–452.

[41] Y. Li, L. Zou, L. Jiang, and X. Zhou. “Fault diagnosis of rotating machinery based on combination of deep belief network and one-dimensional convolutional neural network”. In: *IEEE Access* 7 (2019), pp. 165710–165723.

[42] R. Liu, B. Yang, E. Zio, and X. Chen. “Artificial intelligence for fault diagnosis of rotating machinery: A review”. In: *Mechanical Systems and Signal Processing* 108 (2018), pp. 33–47.

[43] Y. Liu, L. Guo, H. Gao, Z. You, Y. Ye, and B. Zhang. “Machine vision based condition monitoring and fault diagnosis of machine tools using information from machined surface texture: A review”. In: *Mechanical Systems and Signal Processing* 164 (2022), p. 108068.

[44] C. Lu, Z.-Y. Wang, W.-L. Qin, and J. Ma. “Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification”. In: *Signal Processing* 130 (2017), pp. 377–388.

[45] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang. “Learning under concept drift: A review”. In: *IEEE Transactions on Knowledge and Data Engineering* 31(12) (2018), pp. 2346–2363.

[46] S.-I. Manufacturing and A. R. T. (at the University of Michigan. *CNC Milling Dataset*. `https://www.kaggle.com/shasun/tool-wear-detection-in-cnc-mill`. (Date accessed: 25.05.2023). 2018.

[47] Q. Miao and V. Makis. “Condition monitoring and classification of rotating machinery using wavelets and hidden Markov models”. In: *Mechanical systems and signal processing* 21(2) (2007), pp. 840–855.

[48] M. Milfelner, F. Cus, and J. Balic. “An overview of data acquisition system for cutting force measuring and optimization in milling”. In: *Journal of Materials Processing Technology* 164 (2005), pp. 1281–1288.

[49] T. Mohanraj, S. Shankar, R. Rajasekar, N. Sakthivel, and A. Pramanik. “Tool condition monitoring techniques in milling process—a review”. In: *Journal of Materials Research and Technology* 9(1) (2020), pp. 1032–1042.

[50] A. R. Mohanty. *Machinery condition monitoring: Principles and practices*. CRC Press, 2014.

[51] C. Nath. “Integrated tool condition monitoring systems and their applications: a comprehensive review”. In: *Procedia Manufacturing* 48 (2020), pp. 852–863.

[52] A. Rai. “Explainable AI: From black box to glass box”. In: *Journal of the Academy of Marketing Science* 48(1) (2020), pp. 137–141.

[53] R. B. Randall and J. Antoni. “Rolling element bearing diagnostics—A tutorial”. In: *Mechanical systems and signal processing* 25(2) (2011), pp. 485–520.

[54] Z. Ren, Y. Zhu, K. Yan, K. Chen, W. Kang, Y. Yue, and D. Gao. “A novel model with the ability of few-shot learning and quick updating for intelligent fault diagnosis”. In: *Mechanical Systems and Signal Processing* 138 (2020), p. 106608.

[55] M. Rizal, J. A. Ghani, M. Z. Nuawi, and C. H. C. Haron. “An embedded multi-sensor system on the rotating dynamometer for real-time condition monitoring in milling”. In: *The International Journal of Advanced Manufacturing Technology* 95(1) (2018), pp. 811–823.

[56] D. Schumacher. *Drilling CNC Machine*. `https://pixabay.com/photos/drill-milling-machining-chips-84848/`. Accessed: 2022-07-30. 2013.

[57] G. Serin, B. Sener, A. M. Ozbayoglu, and H. O. Unver. “Review of tool condition monitoring in machining and opportunities for deep learning”. In: *The International Journal of Advanced Manufacturing Technology* 109(3) (2020), pp. 953–974.

[58] A. Shaheryar, X.-C. Yin, and W. Yousuf. “robust feature extraction on vibration data under deep-learning framework: An application for fault identification in rotary machines”. In: *International Journal of Computer Applications* 167(4) (2017), pp. 37–45.

[59] A. Siddhpura and R. Paurobally. “A review of flank wear prediction methods for tool condition monitoring in a turning process”. In: *The International Journal of Advanced Manufacturing Technology* 65(1) (2013), pp. 371–393.

[60] A. Stief and J. Baranowski. “Fault diagnosis using interpolated kernel density estimate”. In: *Measurement* 176 (2021), p. 109230.

[61] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions”. In: (2015), pp. 1–9.

[62] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[63] Y. Tian, X. Zhao, and W. Huang. “Meta-learning approaches for learning-to-learn in deep learning: A survey”. In: *Neurocomputing* 494 (2022), pp. 203–223.

[64] 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.

[65] M.-A. Tnani, P. Subarnaduti, and K. Diepold. “Efficient feature learning approach for raw industrial vibration data using two-stage learning framework”. In: *Sensors* 22(13) (2022), p. 4813.

[66] M.-A. Tnani, P. Subarnaduti, and K. Diepold. “Extract, Compress and Encode: LitNet an Efficient Autoencoder for Noisy Time-Series Data”. In: *2022 IEEE International Conference on Industrial Technology (ICIT)*. IEEE. 2022, pp. 1–8.

[67] T. T. Um, F. M. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić. “Data augmentation of wearable sensor data for parkinson’s disease monitoring using convolutional neural networks”. In: *Proceedings of the 19th ACM international conference on multimodal interaction*. 2017, pp. 216–220.

[68] 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), pp. 144–156.

[69] M. Wang and J. Wang. “CHMM for tool condition monitoring and remaining useful life prediction”. In: *The International Journal of Advanced Manufacturing Technology* 59(5) (2012), pp. 463–471.

[70] A. Widodo and B.-S. Yang. “Support vector machine in machine condition monitoring and fault diagnosis”. In: *Mechanical systems and signal processing* 21(6) (2007), pp. 2560–2574.

[71] D. Xu, Z. Zhang, and J. Shi. “A New Multi-Sensor Stream Data Augmentation Method for Imbalanced Learning in Complex Manufacturing Process”. In: *Sensors* 22(11) (2022), p. 4042.

[72] K. Yang, G. Wang, Y. Dong, Q. Zhang, and L. Sang. “Early chatter identification based on an optimized variational mode decomposition”. In: *Mechanical Systems and Signal Processing* 115 (2019), pp. 238–254.

[73] Z. Yang, B. Xu, W. Luo, and F. Chen. “Autoencoder-based representation learning and its application in intelligent fault diagnosis: A review”. In: *Measurement* 189 (2022), p. 110460.

[74] J. Yu. “Health condition monitoring of machines based on hidden Markov model and contribution analysis”. In: *IEEE Transactions on Instrumentation and Measurement* 61(8) (2012), pp. 2200–2211.

[75] M. Zareapoor, P. Shamsolmoali, and J. Yang. “Oversampling adversarial network for class-imbalanced fault diagnosis”. In: *Mechanical Systems and Signal Processing* 149 (2021), p. 107175.

[76] A. Zhang, S. Li, Y. Cui, W. Yang, R. Dong, and J. Hu. “Limited data rolling bearing fault diagnosis with few-shot learning”. In: *Ieee Access* 7 (2019), pp. 110895–110904.

[77] F. Zhang, Y. Liu, C. Chen, Y.-F. Li, and H.-Z. Huang. “Fault diagnosis of rotating machinery based on kernel density estimation and Kullback-Leibler divergence”. In: *Journal of Mechanical Science and Technology* 28 (2014), pp. 4441–4454.

[78] T. Zhang, J. Chen, F. Li, K. Zhang, H. Lv, S. He, and E. Xu. “Intelligent fault diagnosis of machines with small & imbalanced data: A state-of-the-art review and possible extensions”. In: *ISA transactions* 119 (2022), pp. 152–171.

[79] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang. “A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals”. In: *Sensors* 17(2) (2017), p. 425.

[80] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao. “Deep learning and its applications to machine health monitoring”. In: *Mechanical Systems and Signal Processing* 115 (2019), pp. 213–237.

[81] Z. Zhao, T. Li, J. Wu, C. Sun, S. Wang, R. Yan, and X. Chen. “Deep learning algorithms for rotating machinery intelligent diagnosis: An open source benchmark study”. In: *ISA transactions* 107 (2020), pp. 224–255.

[82] F. Zhou, S. Yang, H. Fujita, D. Chen, and C. Wen. “Deep learning fault diagnosis method based on global optimization GAN for unbalanced data”. In: *Knowledge-Based Systems* 187 (2020), p. 104837.

[83] Y. Zhu, M.-A. Tnani, T. Jahnz, and K. Diepold. “Active Transfer Prototypical Network: An Efficient Labeling Algorithm for Time-Series Data”. In: *Procedia Computer Science* 217 (2023), pp. 1427–1436.

Copyrights

RightsLink

Home | Help | Live Chat | Sign in | Create Account

Extract, Compress and Encode: LitNet an Efficient Autoencoder for Noisy Time-Series Data

Conference Proceedings: 2022 IEEE International Conference on Industrial Technology (ICIT)

Author: Mohamed-Ali Tnani

Publisher: IEEE

Date: 22 August 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW