CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Predictive Maintenance in Production Robots in a Real World Industrial Setting

A comparative study using different machine learning models in a real world application

Master's thesis in Computer science and engineering

ADAM SVENSSON
KARL WENNERSTRÖM

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023

Master's thesis 2023

Predictive Maintenance in Production Robots in a Real World Industrial Setting

A comparative study using different machine learning models in a real world application

ADAM SVENSSON
KARL WENNERSTRÖM

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Predictive Maintainance in Production Robots in a Real World Industrial Setting
A comparative study using different machine learning models in a real world application
ADAM SVENSSON
KARL WENNERSTRÖM

Supervisor: Yinan Yu, Department of Computer Science and Engineering
Advisor: Niklas Björk, Sandvik Coromant AB
Advisor: Fräs Jan Jonsson, Sandvik Coromant AB
Examiner: Mary Sheeran, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Predictive Maintainance in Production Robots in a Real World Industrial Setting
A comparative study using different machine learning models in a real world application

ADAM SVENSSON
KARL WENNERSTRÖM
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

With the exponential growth of data and advancements in AI technology, Predictive Maintenance (PdM) has emerged as a vital practice for optimizing equipment maintenance and minimizing unplanned downtime. This study was performed in collaboration with Sandvik Coromant, a company producing steel products and actively collecting data for evaluation purposes, to investigate how the collected data can be utilized for decision-making processes. Specifically, the study analyses vibrational data to address the issues related to unexpected tool malfunctions. Based on the available data, anomaly detection was identified as the most suitable approach to leverage the stored data based on its characteristics. A comparative study where various anomaly detection models were evaluated demonstrated that a reconstruction-based LSTM autoencoder yields the highest performance. The reconstruction approach exhibited its effectiveness in detecting and flagging potential abnormalities, capturing 71% of the malfunctions with an F_1 score of 0.75 for the data used for the comparison. Extending the model to other tools displayed the challenges posed within time-series analysis, proving unique characteristics for each case. The findings from this study provide valuable insights into the implementation of anomaly detection techniques for leveraging collected data and enhancing decision-making processes in Sandvik Coromant and similar industrial settings.

Keywords: Machine learning, Time series reconstruction, Time series forecasting, Anomaly detection.

Acknowledgements

We would like to express our deepest gratitude to our supervisor, Yinan Yu, for always guiding us when fumbling in the dark with your insightful feedback and late-night replies. Your ability to find a new perspective on each situation contributed greatly to this work.

In addition, we would like to extend our appreciation to the entire team at Sandvik Coromant, with special recognition to Niklas Björk for assisting us in organizing and outlining the project, and Fräs Jan Jonsson for patiently addressing all of our numerous questions. To all of you: you made us feel welcome from day one and your invaluable support and guidance have been instrumental in our progress. You helped us with your high spirits and provided us with the resources needed to complete this project. We are grateful for your trust in our abilities and the opportunity to work on such an exciting project.

Also, a big thank you to Syntronic for providing us with workstations at your office.

Last but not least, we are deeply thankful to family and friends for all the love and support you have given us during this project, despite our absence. We could not have done this without you.

Adam Svensson, Karl Wennerström, Gothenburg, 2023-06-16

Contents

List of Figures

List of Tables

1

Introduction

The mitigation of equipment failure in production environments is crucial for all manufacturing companies. Such failures can result in maintenance costs that account for a significant portion of overall manufacturing expenses, ranging from 15% to 60% [1]. These expenses may include the production of faulty products, equipment repairs, and production downtime [2]. To address this issue, advanced technologies such as artificial intelligence (AI) have been proposed as a solution, enabling smart manufacturing and paving the way for Industry 4.0, the fourth industrial revolution [3]. In an Industry 4.0 setting, components, machines, and production systems are integrated and connected, allowing for the real-time transfer of information between production site components and employees [4]. The access to time series data, collected at regular intervals from machine and sensor signals, forms the foundation for utilizing AI advancements in predictive maintenance (PdM) models, to make informed and predictive decisions that improve production efficiency.

PdM models, which can be statistical, machine learning (ML), or deep learning (DL) based, rely on historical data to forecast equipment breakdowns and provide insight into the decision-making process [5]. The choice of PdM model depends on the characteristics of the data and the specific needs of the facility. Two common approaches are (1) predicting the remaining useful life (RUL) of the equipment or (2) identifying unexpected behavior, also known as anomaly detection. Previous research has explored statistical models such as ARIMA, ML models such as SVM and RF, and DL models including AutoEncoder, MTAD-GAT, LSTM, ANN, and RNN. However, there is no general recipe for achieving an accurate PdM model as each project has unique circumstances [4]. Anomaly detection studies are often conducted on synthesized data, leaving the need for further investigation in real-world applications.

In this project, a real world scenario is studied where time series data from production machines at Sandvik Coromant AB is analyzed, in order to find an appropriate approach to reach an industry 4.0 scenario. The vast amount of data collected on a daily basis helps monitoring and evaluating the production, but the aim is to utilize the data for decision-making processes in order to reduce production downtime and thus increase the overall production efficiency. Due to the challenges posed by time series forecasting, such as imbalanced data, insufficient labeled data and erroneous data, the first objective is to decide whether the approach to predict RUL or detecting anomalies is the appropriate approach. Then, a comparative study between

various models within the chosen technique is carried out, focusing on the trade-off between detected anomalies and false alarms, together with other requirements presented by the company. Finally, an analysis of the results is performed, investigating the effects that a PdM model can accomplish.

1.1 Background

In this section, a background to the requirements for industry 4.0 is presented, together with the current workflow at Sandvik Coromant AB.

1.1.1 Achieving Industry 4.0

Industry 4.0 is seen as a way to increase manufacturing efficiency, reduce costs, and improve product quality [5]. The ultimate goal of Industry 4.0 is to create a fully connected and automated production process, where machines can communicate with each other and production can be optimized in real-time. The use of Industry 4.0 technologies is becoming increasingly important for manufacturing companies to stay competitive in the global market [1]. It enables companies to adapt to changes in demand and produce goods in a flexible and efficient way. However, the adoption of Industry 4.0 technologies requires significant investment, as it often involves the redesign of production lines, the installation of new equipment and the training of personnel to operate and maintain the new technology.

Industry 4.0 has brought significant changes to the manufacturing industry, including advancements in maintenance strategies. There are four types of maintenance strategies: (1) corrective, (2) preventive, (3) predictive, and (4) prescriptive [5]. In corrective maintenance, also known as run-to-failure (R2F), the production process continues until equipment failure occurs. While such a strategy exploits equipment lifetime, each failure cause major impact in the production environment. Preventive maintenance, on the other hand, is scheduled maintenance. The time interval between maintenance occasions should be large enough to utilize the expected lifetime of the equipment while also minimizing the risk of an equipment failure. Predictive maintenance is a strategy that utilizes equipment information to enable more flexible actions, allowing for both optimal utilization of equipment and the discovery of unexpected behavior that may indicate premature malfunction. Data is streamed from sensors and analyzed to make predictions on equipment status, using various machine learning techniques, and appropriate actions are taken accordingly. The highest level of maintenance is prescriptive maintenance, which combines a predictive system with suggested measures and, in some cases, even automates these actions. However, the successful implementation of these techniques requires certain prerequisites [3][1].

To transition from real-time data monitoring and manual equipment inspection to a data-driven decision-making approach, certain conditions must be met [1]. These conditions relate to data collection during production processes, evaluation capabilities, and the IT system for monitoring equipment conditions. The specific, relevant requirements are listed in Tab. 1.1.

Table 1.1: Necessary requirements to successfully achieve an industry 4.0 scenario.

Area	Requirements
Data collection	Continuous inspection Sensors and other data Digital recording
Performance measurement	Digital trend analyses
IT	Condition monitoring software Big data platform

The foundation is the collection of data. All processes regarding monitoring or analyzing the data in any fashion require existing and accurate data. In [3], the authors propose a list of question to evaluate how well the situation at hand is equipped to approach a PdM scenario. The appropriate questions, related to the data, are posted below (not ordered by relevance).

1. Can the problem be formulated as a binary classification problem?
2. Are predictor features available?
3. Are all sources of variation captured by the predictor features?
4. Are the predictor features strongly related to the underlying physics of the project?
5. Is the training data (i.e., sample size) big enough?
6. Are there labels (i.e., good, defective) available?
7. If there are no labels, how long would it take to generate labels?

Analyzing these questions can provide valuable insights into the feasibility and potential success of a PdM project. It is worth noting that these questions can be adapted to suit the specific requirements of each individual project and its unique setting. For instance, depending on the project's characteristics, it may be more appropriate to formulate the problem as a multi-classification or regression task.

In the following section, we will discuss the requirements for a successful PdM implementation from the perspective of the company.

1.1.2 Background at Sandvik

Sandvik Coromant AB is a Swedish multinational engineering company that specializes in manufacturing and selling cutting tools and solutions for the metalworking industry. The company was founded in 1942 in Sandviken, Sweden and todays manufacturing is mainly done in their industrial facility in Gimo, Sweden. Sandvik Coromant is a subsidiary of the Sandvik Group, a global industrial engineering

company. Their product portfolio includes a wide range of cutting tools, including milling cutters, turning tools, drills, and boring tools, as well as tooling systems and solutions for various applications. The products are used in a variety of industries, including aerospace, automotive, medical, oil and gas, and general engineering.

Sandvik Coromant has a global presence, with sales offices and manufacturing facilities in over 130 countries. The company employs over 7,000 people worldwide and generated sales of approximately SEK 20 billion in 2020. Sandvik Coromant is recognized as a leading provider of cutting tools and solutions.

At Gimo, there are more than 80 production machines, where some of them are tasked with producing the same products. A production robot has on average a life length of up to 25 years, according to experts at Sandvik, thus causing the production facility to have various age of their machines. The machines involved in this project operate using a single tool at a time, which is referred to as the *master tool*. However, they are designed to hold multiple tools in order to minimize the time required to switch between tools. Additionally, each tool has one or several backups, referred to as the *sibling tool*, which can be used if the master tool is unavailable or has malfunctioned. Generally, the tool with the shorter estimated remaining life is used. For each tool, the theoretical remaining life is displayed, counting down from a fixed level, the maximum life length, specified by the type of the tool. After each use, laser measurements are utilized to inspect the condition of the tool. If it is broken, production stops and the quality of the previously produced products are manually inspected.

The production site in Gimo is Sandvik's most automated and digitized factory, with a plant performance team dedicated to analyze the information derived from both raw machine signals and manually installed sensors, capturing machine specific signals and vibrational data from the machine. In general, hundreds of signals are collected from each machine and their integrated sensors. This stream of data is the key to efficient plant performance, enabling monitoring and valuable insights regarding operational statistics, managed via a digital platform. Thus, the requirements posed in Tab. 1.1 are fulfilled. Further insight in how eligible the current workflow is for industry 4.0, is obtained by answering the questions declared above.

The company has been collecting data for many years for some features, while other features correspond to more recently installed sensors. The sampling frequency is configurable, but generally the values are logged per second, averaging the values collected during that time frame. Overall, it is safe to say that the collected data compose *enough* data. The features collected from the machines and their sensors constitute the predictor features. Amongst these are information about physical aspects related to the drilling and turning processes, for example vibrations and rotational speed. Indications of when equipment is malfunctioning are also reported. The company also keep track of estimated remaining life and the time each equipment has been operating. Combining this with equipment malfunction enables both labels for a regression model but also the construction of anomaly sequences. According to the answers to the questions, the conditions appear to be favourable for approaching a PdM strategy.

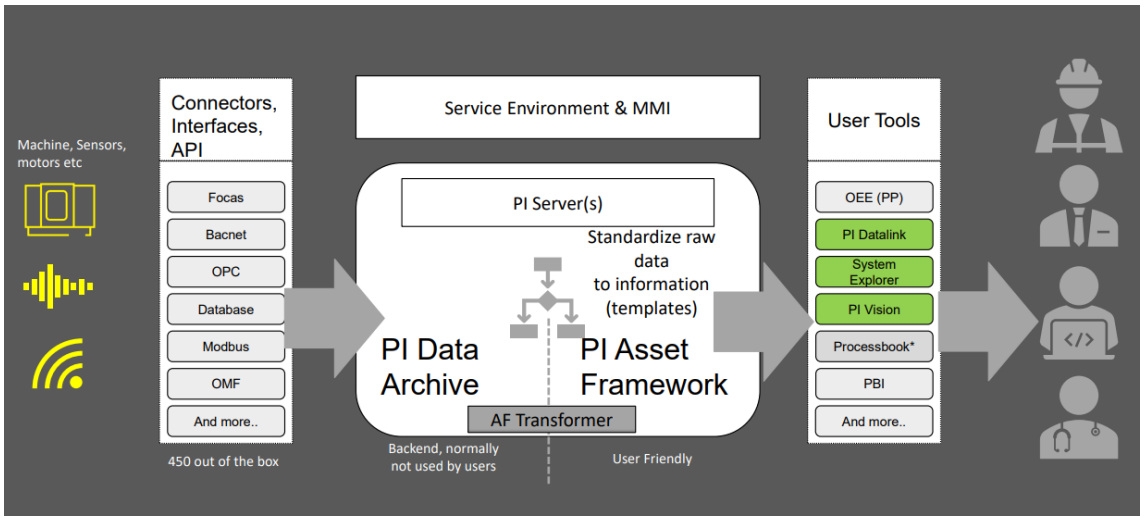


Figure 1.1: The digital platform framework present at the company. Connectors are used to collect raw machine signals from the machines, allowing to connect to various electronic components. Raw signals are stored in the PI Data Archive and is preprocessed in PI Asset Framework to possess human readable information, which can then be used in several user tools.

1.1.2.1 Data Infrastructure

An overview of the digital platform is presented in Fig. 1.1. The raw signals are stored in a PI Data Archive, which is a traditional data historian database storing signals together with a timestamp. In order to reduce the amount of storage, most signals are only stored whenever there is a significant change between measurements. The threshold for what is a significant change differs between signals.

In order to translate raw data to information, they apply logic operations to extract useful information about the signals. This information is further processed and used in user-friendly tools for both plant workers by monitoring machine health and office workers for analyses of plant performance, such as overall equipment efficiency.

1.1.2.2 Problem

The current workflow at Sandvik Coromant AB for the selected machines incorporates the preventive maintenance strategy, which infers that the equipment is replaced according to scheduled occasions. In the case of a tool breakage, the recently manufactured products need to be manually inspected to distinguish between defected and undamaged goods to ensure product quality. This results in production downtime for equipment maintenance and quality assessment, but also a loss of revenue due to increased usage of materials and resources. If these issues can be mitigated, substantial cost savings can be achieved by the company. The most promising information is described to be encapsulated in the vibrational data, where experts already can visually determine that a tool is new by larger vibrations than usual.

1.2 Our Project

In this section we present the scope of this project and our research questions.

1.2.1 Project aim

The overall goal with this project is to investigate how the collected time-series data from the production of Sandvik Coromant in Gimo can be utilized in order to mitigate the effects caused by tool malfunction. In addition, it will also be investigated which improvements such a data driven solution would entail. This will be achieved by studying collected data from specific machines, equipped with sensors to report vibrational signals. A subset of tools is selected to perform the experiments on. Starting from tool-level, we identify a model, statistical or ML-based, to identify tool malfunctions based on vibrational data, with the aim that such a model can generalize to unseen tools.

In order to concretize the goals of this project, we formulate one broader research question:

- *How can production be improved with the use of collected time-series data?*

The high-level question can be broken down into three sub-questions appropriate for the outline of the project:

1. *What is an appropriate approach to utilize the collected data based on its characteristics?*
2. *What is an appropriate model for detecting malfunctions in the equipment of industrial robots at Sandvik Coromant AB?*
3. *Which improvements can be achieved by (2) without loss of quality and minimizing the production downtime?*

Given the sub-questions, the primary focus of this project is to find a model according to (2) by comparing models of various character and complexity. Their performance will be evaluated in terms of detected malfunctions combined with the amount of raised false alarms.

1.3 Thesis Outline

In this section we give a brief overview of structure the project. The second chapter contains the theory that has been used in the project. It introduces time series, machine learning building blocks and theory related to anomaly detection. The methodology is further explained in chapter three. In chapter 4, the results of the experiments are showcased followed by discussion in chapter 5. The conclusions as well as the future work and improvements are presented in chapter 6. Lastly, the key takeaways are provided in chapter 7.

2

Theory

The theory chapter of this thesis presents the theoretical framework underpinning the research. It provides an overview of key concepts and ideas that shape the study and is used when exploring the research questions. First, an overview of the related work is presented. Then, a description of time series data and its characteristics is given. Finally, machine learning and related concepts are introduced to form the foundation needed to understand the approaches investigated in this project: RUL prediction and anomaly detection.

2.1 Related Work

Predictive Maintenance Approaches: PdM in production environment is a critical research area as it can lead to significant improvements in production downtime, product quality, and cost savings. There is no one-fits-all solution. Some studies have approached PdM in a supervised fashion, by predicting RUL [6][2][7]. Another common approach is to use anomaly detection, a technique applicable within many areas, either by forecasting data [8][9], reconstructing data [10][11][12], or a combination of the two [13][14].

RUL Prediction Models: Predicting RUL has been performed with various models and within different production scenarios. Lee et al [6] use support vector machines (SVM) and artificial neural networks (ANNs) to predict the RUL of a spindle motor and cutting tool in a milling process. Signals of vibrations, current, and acoustic emissions are collected during run-to-failure experiments, at a frequency of 250Hz. From the same experiments, labels are manufactured. Another study performs a full implementation in a manufacturing facility producing personal care goods [2]. By using signals collected at a 3-6s interval, containing, for example, temperature, electrical current and air pressure, a comparison of various machine learning models demonstrates that a random forest (RF) model is the most successful one. Ren et al. [7], analyzes vibrational measurements from bearing parts of an engine, collected from sensors in an experimental setting. They acknowledge the challenges within feature compression and perform time-frequency transformation to simultaneously analyze the vibrations in both the time and frequency domains. The authors combine deep autoencoders with deep neural networks to increase prediction performance. The mentioned studies highlight that various approaches to predict RUL are possible.

Anomaly Detection Models - forecasting vs reconstruction: Anomaly detection is another approach to PdM. It involves identifying abnormal or anomalous behavior in machines or systems, which could be an early indication of potential failure or malfunction. An end-to-end framework for anomaly detection, Sintel [15], includes three stages: preprocessing, modeling and postprocessing stage. The preprocessing stage transforms raw signals into usable time series. Once the signals are processed and ready to use in various anomaly detection models, the transformed input signals are reconstructed, forecasted, or both. In the postprocessing stage, the predicted signals are compared to the actual signals, resulting in error measurements between the expected and actual signals. This project follows the proposed pipeline.

In the forecasting anomaly detection scenario, Ho et al. [16] compared ARIMA, feed forward neural networks and deep learning models, such as RNN, to predict failures in compressors. The findings showed that ARIMA performed well for short-term forecasting, but for longer horizons, deep learning models performed better. Similarly, Makridakis et al. [17], draw the same conclusion, after analyzing time series from various fields, and highlight the trade-off between accuracy and model complexity. Thus, even if the deep learning models perform better, the required computational power might make them infeasible. Hundman et al. [9] utilizes deep learning, specifically LSTM networks, to detect anomalies within spacecraft data obtained from NASA. The authors propose a dynamic error threshold, to mitigate false positives (FPs) by a pruning process where previously classified anomalies can be reclassified as normal. However, this method requires that all predictions are available before reclassifying is made possible. Thus, for this project, the dynamic error threshold is not feasible. Due to its proven effectiveness, ARIMA will be used as one of the models for short-term forecasting in our project.

Reconstruction-based methods aim to learn a compressed, low-dimensional representation of the input data that can reconstruct the original input with high accuracy. Such techniques assume that the learned latent space captures the dominant patterns in the dataset. However, rare events such as anomalies are often not well-represented in the latent space and are less likely to be accurately reconstructed. Deep learning models have shown promising results for detecting anomalies in multivariate time series. Malhotra et al. [18], proposes an LSTM autoencoder model for detecting anomalies in datasets with different levels of predictability. More specifically, the model performs well on a dataset with more unpredictable behavior, containing sensor readings of, for example, coolant temperature, torque, and acceleration. Also, generative adversial networks (GANs), have been suggested as models for different scenarios. Li et al. [12], propose multivariate anomaly detection with GAN (MAD-GAN) to detect cyber-attacks in water treatment systems. Time series anomaly detection with GAN (TadGAN), proposed by [19], outperforms MAD-GAN on multiple datasets, presenting different reconstruction error measurements, such as *point-wise difference* and *area difference*.

Hybrid Anomaly Detection Models: For the combination of reconstruction and forecasting anomaly detection models, Wong et al. [13] propose autoencoder with regression (AER) to detect anomalies in univariate time series. AER extends

the normal usecase of an autoencoder by yielding three components: a one-step-backward prediction, a reconstructed sequence, and a one-step-ahead prediction. A multivariate approach is proposed by Zhao et al. [14], called multivariate time series anomaly detection with graph attention network (MTAD-GAT). MTAD-GAT treats each univariate time series as a feature, utilizing two parallel graph attention networks to capture both feature and temporal dependencies. It combines reconstructing the time series, performed by a variational autoencoder (VAE), with a one-step-ahead prediction. The threshold for detecting anomalies is determined by the pruning technique mentioned above.

Overall Model Performance: Context, data availability, and computational constraints play a significant role in model selection. For instance, RF performs well for RUL prediction. When there is not enough annotated data, supervised RUL prediction is not feasible, where anomaly detection models are typically considered. ARIMA is effective for short-term forecasting, and deep learning models perform better for longer-term predictions. For unpredictable time-series, reconstruction-based models have proven to work well and especially LSTM autoencoders. Thus, ARIMA and an autoencoder with LSTM layers are investigated for this specific setting. In addition, MTAD-GAT, a hybrid model, is implemented with modifications to suit real-time scenarios in this project.

2.2 Time Series Analysis

A time series is a collection of observations $X_t, t \in \mathcal{T}$, where $\mathcal{T}$ is a finite index set $\{1, 2, ..., n\}$. Every observation is made sequentially over time, meaning that the data points are collected or recorded in a specific order according to their corresponding timestamps. If a single variable changes over time, the time series is referred to as a univariate time series. On the other hand, if there are multiple variables changing over time, the collection is defined as a multivariate time series. Historical time series data contains valuable information that can be harnessed for various purposes. It often exhibits temporal dependencies, characterized by repeated patterns or seasonality, as well as underlying trends. Identifying these patterns is critical in forecasting scenarios. It can also have distributional patterns which are studied when a reconstruction scenario is relevant.

Time series forecasting implies predicting future values of the time series, based on historical values. It can be achieved in various ways and is commonly evaluated by comparing the error between the prediction and the actual value, meaning that evaluation can only be performed when the predicted time step has occurred. When the time series is univariate, only one variable is predicted. On the other hand, when the time series is multivariate, then multiple variables are predicted [13].

Another approach to analyze a time series is through the process of *time series reconstruction*. Reconstruction-based models operate by generating a reconstruction of a given time series sequence. These models utilize an encoding step to embed the data in a low-level representation from which a reconstruction is made in a decoding

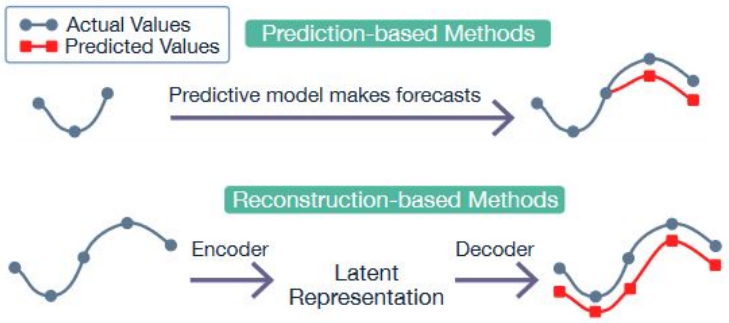


Figure 2.1: Forecasting-based and reconstruction-based models. Image from [13].

step. The idea is to learn a representation of the normal data, so that when inputting a sequence with abnormal values the reconstruction will deviate a lot compared to the true values [20]. In a multivariate scenario, these steps are performed individually for each univariate time series. Evaluation of a reconstruction-based model is done by comparing the reconstructed values with the true values. The comparison can be performed between either whole sequences or specific points, depending on the use-case. An illustration of forecasting and reconstruction made on a univariate time series, respectively, can be seen in Fig. 2.1.

2.3 Machine Learning

Machine learning is a sub-field of AI that involves developing algorithms and models that can learn from data and make predictions based on that data. Industries are exposed to an increasing amount of data, which makes machine learning models very powerful since they can automate processes and improve their accuracy as the data collection increases. Within machine learning there are different approaches depending on the available data. In this project, two main categories are relevant, namely *supervised learning* and *unsupervised learning*.

Supervised learning is a type of machine learning where the model is trained on labeled data. Generally, such a machine learning model $h(\cdot)$ can be described as

$$h_\Theta : \mathbb{R}^{N \times k} \rightarrow \mathbb{R}^{N \times \hat{k}}, \tag{2.1}$$

where Θ is a set of weights and the aim is to map the input data $\mathbf{X}$ to predictions $\hat{\mathbf{Y}}$. N is the number of input samples and each sample has k features. For each input an output is yielded with $\hat{k}$ features. The objective is to find a set of weights Θ^* such that

$$\Theta^* = \arg\min_\Theta J_\Theta(\hat{\mathbf{Y}} - \mathbf{Y}), \tag{2.2}$$

where J is a loss function, measuring the discrepancy between the predicted and the actual value, and $\mathbf{Y}$ is the target. By splitting the data into train and test sets the idea is to learn Θ^* which can be used to predict the output for the unseen data in the test set. Supervised learning can be further divided into, for example, classification and regression tasks. In classification tasks, the objective is to predict a categorical

target variable. In regression tasks, the objective is to predict a continuous target variable, for example the RUL.

On the other hand, a scenario where target labels are not available lies within the category of *unsupervised learning*, where the goal is to identify hidden patterns, groupings, and structures in the data. Anomaly detection is typically an unsupervised scenario, where the labels regarding if a point is an anomaly or not, are not present during training. Instead, the model aims at mimicking the data by either forecasting or reconstructing the data. In forecasting, $\mathbf{Y}$ from Eq. 2.2 becomes $\mathbf{X}_{t+1}$ given that data until time step t is provided as input. In reconstruction, $\mathbf{Y}$ is instead the input data $\mathbf{X}$.

For both supervised and unsupervised learning, the input data is often transformed by some scaling or embedding operation in order to equate the importance of each feature. Scaling can be performed to obtain the features in the same range or distribution. Embedding operations like one-hot encoding are done for categorical variables where each unique category becomes a binary feature if the category is present (1) or not (0).

2.4 Neural Networks

Neural networks are a type of machine learning algorithm, that are made up of layers of fully or partly interconnected nodes, or neurons, that process information in a way that allows them to learn and make predictions based on input data. Neural networks are able to handle complex, nonlinear relationships among the variables in the time series data. As the number of layers and neurons increases $|\Theta|$ increases, resulting in a more complex model, enabling it to extract deeper relationships within the data. However, this complexity also makes the model more computationally expensive [21].

Multivariate time series often have multiple variables that interact with each other in intricate ways and neural networks can be capable of capturing these interdependencies and patterns. Relevant features can be automatically extracted from the data by them, which can help to reduce the dimensionality of the data and improve the accuracy of predictions. In addition, neural networks can be trained to model both short-term and long-term dependencies in the data, which is crucial for accurately predicting future values in a time series.

The basic building block of a neural network is the neuron, which receives input signals from other neurons and processes them to produce an output signal. The input signals are weighted according to their importance and the neuron applies an activation function to the weighted sum of the inputs to produce its output. The activation function can be linear or nonlinear and it is often chosen based on the specific task at hand [21].

Neurons are organized into layers, with each layer processing information and passing it on to the next layer. The first layer of a neural network is the input layer, which receives the initial data. The output layer produces the final output of the network

and one or more hidden layers in between provide additional processing and feature extraction.

There are many different types of networks. For this project, some relevant ones are *recurrent networks*, *graph attention networks*, and *autoencoders*.

2.4.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are designed to work with sequential data such as time series data or text data, where the previous inputs and outputs of the network are used to inform the current output [21]. In multivariate time series analysis, RNNs can be trained to model complex and nonlinear relationships between variables at different time steps. This enables them to capture temporal dependencies and patterns that may not be apparent using other methods.

A long short-term memory (LSTM) model is a specific type of RNN that uses a complex gating mechanism to control the flow of information through the network, allowing it to selectively forget or remember information from previous time steps [22]. Gated recurrent units (GRUs) are a simplified version of LSTM networks, that uses fewer gating units that adaptively control the information flow inside the unit, leading to a powerful model with fewer parameters to train [23].

The specific approach depends on the availability of the data. For example, RNNs can be used in supervised learning approaches when predicting RUL by leveraging labeled historical data to learn temporal patterns and dependencies that contribute to the degradation of the equipment. An additional application of RNN is the detection of anomalies in unlabeled data. This is accomplished by training the RNN on a dataset consisting mainly of normal or expected data, enabling the network to learn these patterns. Subsequently, the deviations between the true values and the predicted values can serve as an indicator of potential anomalies.

2.4.2 Graph Attention Network

Graph Attention Networks (GATs) are a specialized type of neural network architecture that is specifically designed for processing graph-based data, as opposed to the grid-like data structure of images that traditional neural networks are designed for. GATs excel at handling non-Euclidean data, such as graphs, and employ attention mechanisms to identify and focus on the most significant parts of the graph for solving the given problem. This enables the model to pay close attention to specific nodes and edges in the graph, which can enhance the overall accuracy and efficiency of the network [24]. In multivariate time series, GATs can be used to model both the temporal dependencies and the dependencies across the feature space, as seen in Fig. 2.2.

In more detail, the input to a single graph attention layer is a set of node features $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}$, where each $x_i \in \mathbb{R}^k$, where n and k are the numbers of nodes and features respectively. The self-attention scores, determining the contribution of node j's features to node i, are computed according to

$$e_{ij} = a^T \text{LeakyReLU}(\boldsymbol{W}[x_i \oplus x_j]) \tag{2.3}$$

as proposed in Brody et al [25]. Here $\boldsymbol{W}$ and a^T is a learnable weight matrix and vector, respectively. The operator $\oplus$ denotes concatenation. For a small scalar $k > 0$, LeakyReLU is a modified ReLU-function defined in Eq. 2.4 [26].

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ -kx, & \text{if } x < 0 \end{cases} \tag{2.4}$$

A softmax function is applied to the attention scores e_{ij} and the normalized attention scores between node i and node j is represented by the coefficients α_{ij} and can be expressed as

$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}, \tag{2.5}$$

where $\mathcal{N}_i$ is the set of nodes, with edges connected to node i. The output from the single graph attention layer, x_i' is computed in Eq. 2.6 using a non-linear activation function σ.

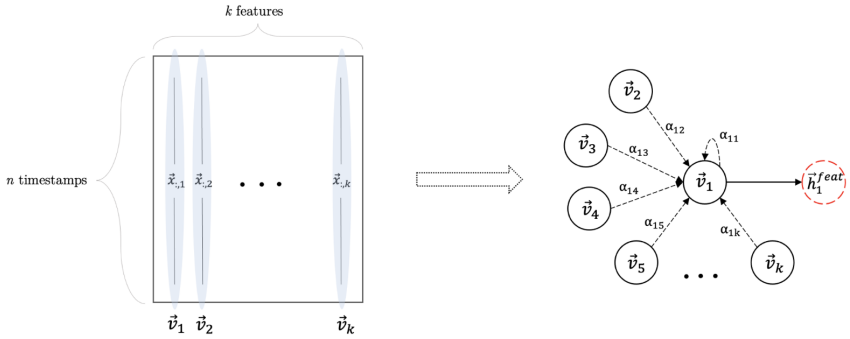
$$x_i' = \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \, \boldsymbol{W} x_j \Big) \tag{2.6}$$

When utilizing GAT in a multivariate time series scenario, two graphs are created: one where each node represents a single feature across all timestamps shown in Fig. 2.2a and another where each node represents a single timestamp across all features shown in Fig. 2.2b. Thus, a GAT network can leverage both the temporal dependencies across time steps and the relations between features making it useful in detecting patterns in multivariate time series datasets.

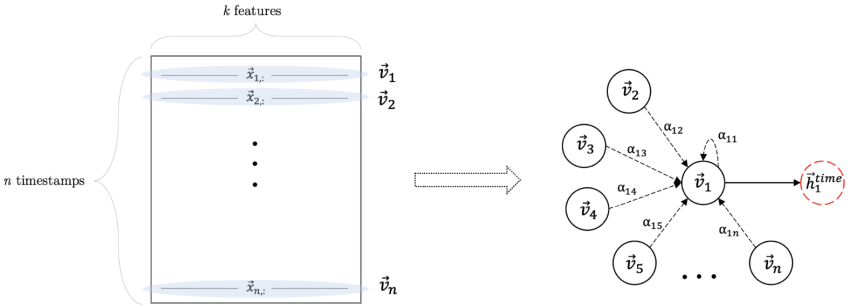
2.4.3 Autoencoder Architecture

Autoencoders are a type of neural network commonly used for unsupervised learning tasks [21], with the main task of reconstructing data. They consist of two parts: an encoder and a decoder. The encoder compresses the input data using a series of layers such as convolutional, fully connected, or RNN layers [27]. These layers progressively reduce the dimensionality of the input, capturing the most important and relevant features. The output of the encoder, which is the compressed representation, is often referred to as the latent space or bottleneck layer. The decoder consists of layers that gradually expand the dimensionality, mirroring the encoding layers but in reverse. The reconstruction process is driven by the learning objective of minimizing the loss between the reconstruction and the observation.

A representation of its architecture can be seen in Fig. 2.3. In a scenario where sequential data is analyzed, autoencoders are often combined with RNNs to keep temporal dependencies. In these cases LSTM or GRU layers are used for the encoder and decoder part, passing also the sequential dependencies between the layers. Hence, autoencoders are suitable for time series reconstruction.



(a) Feature oriented GAT layer treats the input data as a fully connected graph, where each node corresponds to the values of a single feature over all timestamps within the sliding window.



(b) Time oriented GAT layer treats the input data as a fully connected graph, where each node corresponds to the values of a single timestamp over all features within the sliding window.

Figure 2.2: Multivariate time series can be transformed into fully connected graphs in the temporal and feature dimensions, allowing for modeling in each dimension separately. Images from [14].

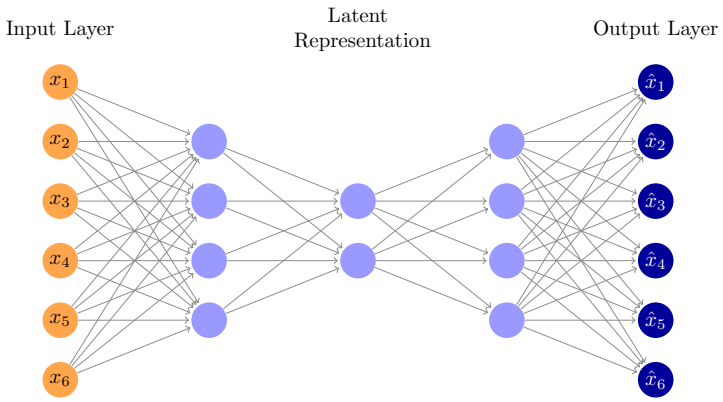


Figure 2.3: Basic autoencoder architecture overview. From input, the data is compressed to a latent representation through a decoder. And the decoder reconstructs the input data from the latent representation.

2.5 ARIMA

ARIMA models utilize a statistical modeling approach to find the best fit of a time series to model future values of a times series [28]. This technique facilitates the recognition of significant patterns and trends within the data, which can subsequently be leveraged to predict future values. This approach combines three different models:

1. Autoregression (AR) - A model that uses past values of a time series to predict future values.
2. Integrated (I) - A model that transforms a non-stationary time series (that is, the mean, variance and autocorrelation is varying over time) into a stationary one, by taking the difference between consecutive samples.
3. Moving Average (MA) - A model that uses the residuals of past predictions to predict future values.

ARIMA is defined by three parameters, p, d and q, which respectively represent the order of autoregressive, integrated and moving average parts of the model. The time series is differenced d times, while p and q are the number of lagged (previous) observations and residuals, respectively, to take into consideration.

The AR component can be represented as follows:

$$Y_t = c + \phi_1 \cdot Y_{t-1} + \phi_2 \cdot Y_{t-2} + \ldots + \phi_p \cdot Y_{t-p} + \varepsilon_t \tag{2.7}$$

where Y_t is the observed value at time t, c is a constant term, and $\phi_1, \phi_2, \ldots, \phi_p$ are the autoregressive coefficients, which represent the weights assigned to the lagged values. $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$ are the lagged observations and ε_t is the error term at time t. The differencing operation is defined as $\Delta Y_t = Y_t - Y_{t-1}$, performed d times. The moving average component considers the relationship between the current observation and the residuals from past predictions, represented as

$$Y_t = \mu + \varepsilon_t - \theta_1 \cdot \varepsilon_{t-1} - \theta_2 \cdot \varepsilon_{t-2} - \ldots - \theta_q \cdot \varepsilon_{t-q} \tag{2.8}$$

where μ is a constant term of the time series, ε_t represents the error term at time t and $\theta_1, \theta_2, \ldots, \theta_q$ are the moving average coefficients, which represent the weights assigned to the lagged error terms.

Combining the AR, I, and MA components, the ARIMA model can be represented as

$$Y_t = c + \phi_1 \cdot Y_{t-1} + \phi_2 \cdot Y_{t-2} + \ldots + \phi_p \cdot Y_{t-p} + \varepsilon_t - \theta_1 \cdot \varepsilon_{t-1} - \theta_2 \cdot \varepsilon_{t-2} - \ldots - \theta_q \cdot \varepsilon_{t-q} \tag{2.9}$$

ARIMA has been widely used in various fields where time series are present and future values are of interest. In an anomaly detection setting, ARIMA can be used

for forecasting future values of a time series. Anomalies, or unusual patterns in the data, can be detected by comparing the forecasted values with the actual values.

2.6 Anomaly Detection

Anomalies, or outliers, refer to points or patterns in a dataset that deviate from the expected or normal behavior. The appearance and cause of an anomaly can vary depending on the data and application. The types of anomalies are described in [29]. These are as follows:

- *Point outliers:* A point outlier refers to a data point that exhibits abnormal behavior at a specific point in time compared to either the other values in the time series (global outlier) or its adjacent data points (local outlier). Point outliers can be either univariate or multivariate, depending on whether they affect one or more time-dependent variables, respectively.
- *Subsequence outlier:* A subsequence outlier, also called collective outlier relates to a sequence of points whose combined behaviour is not expected. Each individual point in the sequence outlier does not necessarily need to be a point outlier.
- *Contextual outlier:* A contextual outlier is a type of anomaly that is dependent on the context in which the data is observed. Unlike point and sequence outliers, contextual outliers are defined based on their relationship to other variables and the context in which they occur.

These types of outliers can be detected by various approaches, either by forecasting or reconstructing the data. Anomaly detection models are generally unsupervised in the sense that they are not provided with anomaly labels during training. Instead, a sample is determined to be an anomaly based on its anomaly score, which represents the discrepancy between the model output and the observations, in comparison to an anomaly threshold.

2.6.1 Forecasting Based Anomaly Detection

Forecasting based anomaly detection models perform time series forecasting and compare the predictions to the true values. If the error between the predictions and the true value exceeds a certain threshold, then the point is considered as an anomaly. Forecasting one point at a time typically identifies unexpected deviations in the data constituted by point anomalies.

A forecasting based anomaly detection model is trained on normal data, that is, data without any known anomalies. Hence, the model knows the normal behaviour and consequently, abnormal data will not be forecasted well. By relying solely on the forecasting error, labels of anomalies are not required during the training phase, constituting an unsupervised model. This makes them useful for detecting previously unseen types of anomalies, as the model can learn to recognize deviations from normal patterns without being explicitly trained on anomalous data.

Common models used in forecasting based anomaly detection are ARIMA and LSTM models.

2.6.2 Reconstruction Based Anomaly Detection

Reconstruction-based anomaly detection models operate by reconstructing a given time series sequence and comparing it to the actual time series. The general idea behind these models is that if the generated reconstruction closely matches the input sequence, then the sequence is considered normal, otherwise it is anomalous. To achieve this, the model is typically trained on a set of normal (non-anomalous) time series sequences [30]. During training, the model learns to generate accurate reconstructions of these sequences. Once trained, the model can then be used to generate reconstructions of new input sequences. If the reconstruction error (that is, the difference between the generated reconstruction and the input sequence) exceeds a certain threshold, then the input sequence is flagged as anomalous [13]. Since multiple values are reconstructed, subsequence or contextual outliers can be detected without containing point outliers. Autoencoders combined with RNNs are commonly used for time series reconstruction.

2.6.3 Anomaly Score Thresholds

Anomaly scores are computed by comparing the actual time series values to predicted ones. Each data point in the test set is assigned a score, which reflects the quality of the prediction. The lower the score the better the estimation [31]. To evaluate the performance of a model, anomaly classifications are constructed by comparing the anomaly scores to a threshold and classifying as anomaly if the score exceeds the threshold. In our study, we have utilized the *maximum training error*, *epsilon* and *rolling median absolute deviation* (MAD) methods to determine the threshold level and subsequently classify the instances accordingly. The maximum training error as threshold utilizes the fact that only normal data is present during training, thus errors exceeding those experienced during training should be classified as anomalies. Epsilon and MAD threshold methods require further explanation.

2.6.3.1 Epsilon Threshold Method

Suppose that we have the input data $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, where each $x_i \in \mathbb{R}^k$ is a vector containing values for all k features. Also, suppose that we have trained a model and predicted values $\hat{\mathbf{X}} = \{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, ..., \hat{\boldsymbol{x}}_N\}$. The errors $\boldsymbol{e}_i$ for each feature $i \in 1, 2, ..., k$, is then defined as

$$\boldsymbol{e}_i = |\hat{\boldsymbol{x}} - \boldsymbol{x}|_i.$$

Computing the mean μ and standard deviation σ of the training errors for each feature and time stamp t, each predicted value is classified as an anomaly if

$$e_{i,t} \geq \mu_i + \epsilon\sigma_i, \tag{2.10}$$

where $\epsilon \geq 0$.

Table 2.1: Classification outcomes after applying anomaly detection models.

Actual	Prediction	Result
Normal	Normal	TN
Anomaly	Anomaly	TP
Normal	Anomaly	FP
Anomaly	Normal	FN

2.6.3.2 Rolling MAD Threshold Method

Suppose the same scenario is available as in 2.6.3.1, the rolling MAD threshold is obtained by studying the median errors in the training set in a sliding window fashion. For a window size w, the threshold for each feature i is determined as

$$\max\Big(M(x_{i,t}, x_{i,t+1}, ..., x_{i,t+w})\Big),\ \forall t \in 0, 1, ..., N - w, \tag{2.11}$$

where $M(\cdot)$ is the median and N is the number of samples.

2.6.4 Anomaly Detection Evaluation Metrics

Finding a meaningful evaluation metric when working with anomaly detection models is necessary since the distribution of normal versus abnormal points is usually highly imbalanced [31]. Anomaly detection is an unsupervised method but based on the reconstruction and/or forecasting error, classifications can be obtained by comparing the error to the threshold. By comparing the classifications to the true values, the outcomes presented in Tab. 2.1 can be obtained. Using the different classification outcomes, the performance of a model can be determined by different metrics.

Due to the highly imbalanced dataset it is not reasonable to use accuracy as a metric, instead the following metrics will be explored.

$$\begin{aligned} \text{Precision } (P) &= \frac{TP}{TP + FP} \\ \text{Recall } (R) &= \frac{TP}{TP + FN} \\ F_1 &= 2\,\frac{P \cdot R}{P + R} \end{aligned} \tag{2.12}$$

Precision measures the proportion of correctly predicted instances out of all positively predicted instances. In this project, this corresponds to the ratio between true alarms and the total number of alarms. High precision indicates that false alarms are effectively reduced and enhances the reliability towards the model. Recall quantifies the ratio of correctly predicted positive instances to the total number of actual positive instances. This yields a fraction of anomalies successfully detected. High recall indicates that the capabilities of the model towards capturing true anomalies are enhanced. The F_1 score offers a balanced evaluation of performance as it

takes into account both precision and recall. A high F_1 score reflects the ability to accurately capture true anomalies while also minimizing false alarms.

The *Precision-Recall Curve* (PRC) provides a visualization of the trade-off between precision and recall in a classification model. The construction of the curve involves computing precision and recall at various classification thresholds. This is achieved by varying the thresholds, calculating precision and recall for each threshold, and subsequently plotting the precision-recall pairs on a graph to form the PRC.

The *Receiver Operating Characteristic* (ROC) follows the same construction scheme as PRC but instead visualizes the trade-off between TP rate and FP rate for the model. By computing the *Area Under the Curve* (AUC) score, a measurement of the model's performance is provided. The score ranges from 0 to 1 where a higher score indicate a more accurate model.

3

Methods

This section outlines the methodology employed for this project. The section begins with an understanding and exploration of the data to describe its characteristics and patterns. In order to optimize the data for modeling, various steps are considered, including data cleaning, feature extraction, selection, and transformation. Several models, such as a simplistic baseline model, MTAD-GAT, Autoencoder, and ARIMA, are described and compared in terms of their applicability and performance. Lastly, potential pitfalls in the data and modeling processes are identified and discussed for full transparency regarding the robustness and reliability of our findings.

3.1 Data Understanding

In this section a brief introduction to the datasets is presented. We describe how the data is extracted and key properties of the datasets.

3.1.1 Data Description

The signals from production machines and their integrated sensors are stored in a data historian, optimizing its efficiency by only logging values when a change of certain magnitude occurs. Since this happens at various rates for different variables, the continuous variables are interpolated between timestamps while the discrete variables keep their value until change to obtain the data on similar frequency. The data is extracted from two machines of the same type, denoted machine 1 and 2, respectively. The motivation for the chosen machines is recently installed vibration sensors, enabling similar data collection from both machines. Also, using machines of the same type serves the purpose of investigating generalization possibilities, since they share the same set of tools, enabling training on one machine and testing on the other.

The data is collected at a frequency of one observation per second, covering the time period from 2022-05-17 to 2023-03-14, corresponding to the time of the installation of the vibration sensors until the data collection phase of this thesis. During the extraction stage, data pertaining to periods of production downtime is filtered out. For both machines, the same subset of signals is extracted. These signals contain vibrations and machine signals such as material ID, production signal etc. An excerpt of the signals, together with a short description can be found in Tab. 3.1.

Table 3.1: Important signals: name and description.

Signal name	Description [unit]
MDC_MaterialID	Material ID [-]
MDC_ProductionSignal	Production signal [-]
MDC_ToolNbr1	Current tool in spindle [-]
PDA_Actual_Spindel_RPM	Rotational speed of the spindle [rpm]
Object_1_value	Vibration x-joint [milligravity]
Object_2_value	Vibration y-joint [milligravity]
Object_3_value	Vibration z-joint [milligravity]
MDC_AlarmID1-5	Machine alarm. 3190 if tool malfunction. [-]

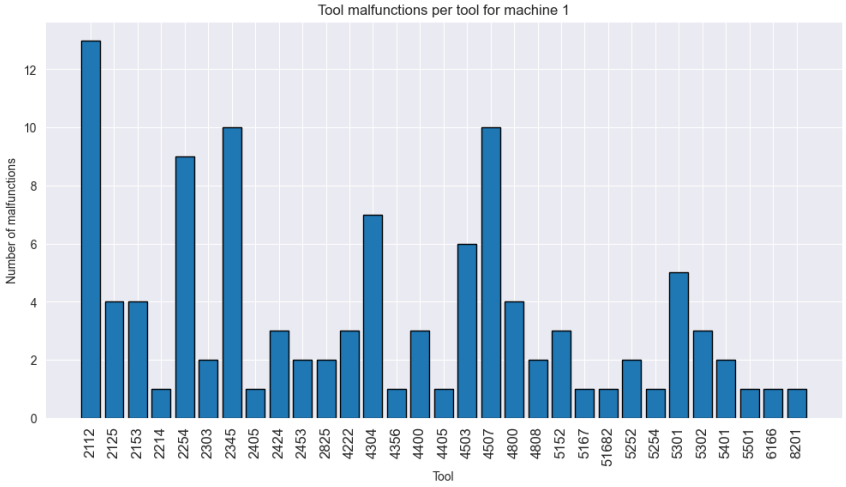
The signals in their entirety are presented in Tab. A.1 in Appendix A. According to literature and expert knowledge at the company the vibrational data are the key signals.

3.1.2 Data Exploration

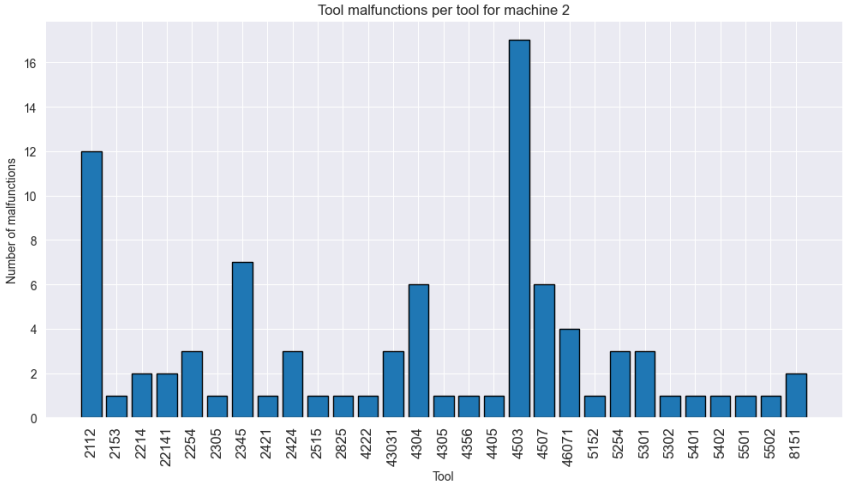
In total, 21 columns are extracted from both machines but the number of rows differs due to individual production stops.

Exploring the data on tool level, one observation reveals that for machine 1 there are 31 unique tools containing at least one tool breakage. The same number for machine 2 is 29. In Fig. 3.1 the number of malfunctions for each tool for the machines, respectively, are displayed. Tools with many malfunctions, either for a single machine or in combination, are: 2112, 2345, 4304, 4503 and 4507. These tools will from here on form a subset on which the experiments are performed. Analyzing the number of data points for each tool, in comparison with the number of malfunctions, seen in Tab 3.2, reveals a scenario of highly imbalanced data sets. Since the data set for machine 2 and tool 4503 has the most number of occurrences of malfunctions, it is from here on denoted *D1* and will be used when comparing models. Another takeaway is that the usage of the tools seems similar for both machines, since the total data points are of comparable magnitude, for a given tool.

To get an understanding of the vibrational behaviour around the malfunctions for D1, Fig. 3.2 and Fig. 3.3 visualize different patterns. In Fig. 3.2, there is a visually clear peak in close relation to the malfunction detection. This spike could illustrate the actual malfunction event or an event leading up to the malfunction. On the other hand, Fig. 3.3 displays no clear oddity before the tool breakage. This indicates that detecting such tool breakages needs to be achieved with other methods than finding point outliers. In total, 7 of the 17 malfunctions in D1 are preceded of a high spike, similar to the first figure. The remaining 10 are of different character, not preceded by abnormally large values. These characteristics are also verified when looking at a visual summary of the data in Fig. 3.4, where we see that the overall distribution remains consistent between the entire dataset and data closely related to malfunctions, in which both the most extreme outliers and normal behaviour is



(a) Machine 1.



(b) Machine 2.

Figure 3.1: Number of malfunctions per tool and machine.

Table 3.2: The number of malfunctions for each tool and machine, together with the total number of data points.

Tool	Machine	Malfunctions	Total data points
2112	1	13	49 158
2112	2	12	26 955
2345	1	10	110 523
2345	2	7	99 544
4304	1	7	8 534
4304	2	6	5 633
4503	1	6	60 608
4503	2	17	67 664
4507	1	10	124 583
4507	2	6	115 662

Table 3.3: The enumerated sequences preceded by one or multiple abnormal values.

Preceded by extreme values	Sequences
YES	4, 6, 7, 8, 11, 14, 15
NO	0, 1, 2, 3, 5, 9, 10, 12, 13, 16

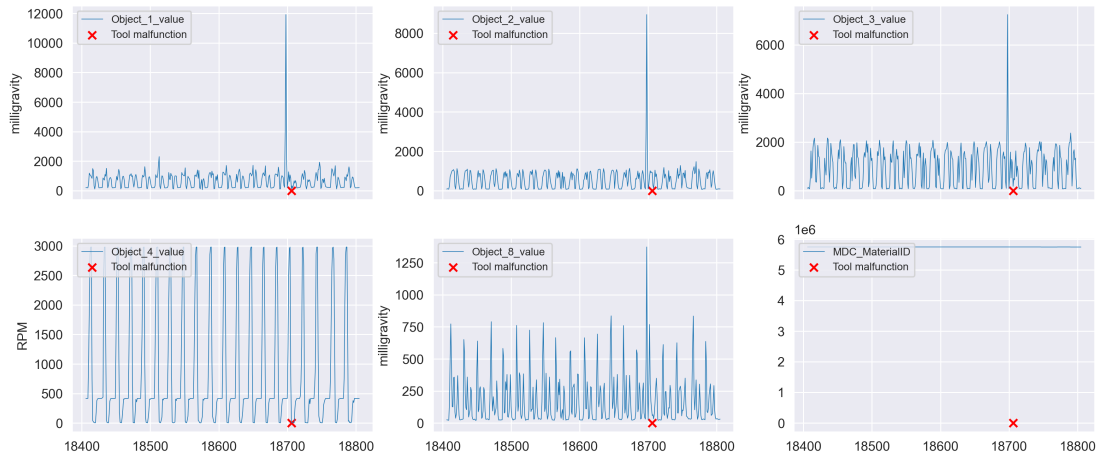


Figure 3.2: Time series data around a malfunction, with an abnormal value prior.

represented. Tab. 3.3 enumerates which characteristic each malfunction expresses. To simplify referencing, sequences in the *Yes*-row are referred to as *group 1* while the remaining constitute *group 2*.

3.2 Technique Exploration

In order to be able to use a regression model predicting RUL, a target variable needs to be created with the true remaining life for each input. Thus, it is necessary to meet certain data requirements. Specifically, accurate data is required for three main aspects: firstly, it is required to have indications for which of the master and the sibling tool is being used. Secondly, we need indications for when a tool is replaced. Lastly, the theoretical remaining life should be present in order to evaluate the performance of the model.

To investigate the feasibility of this information, relevant features are studied. Investigations are performed when the machine is in production, for a selected tool and a contiguous sequence of data points.

For an anomaly detection approach, high-quality data is necessary. Lack of quality can be estimated by analyzing the input data for erroneous entries, such as missing or faulty signals.

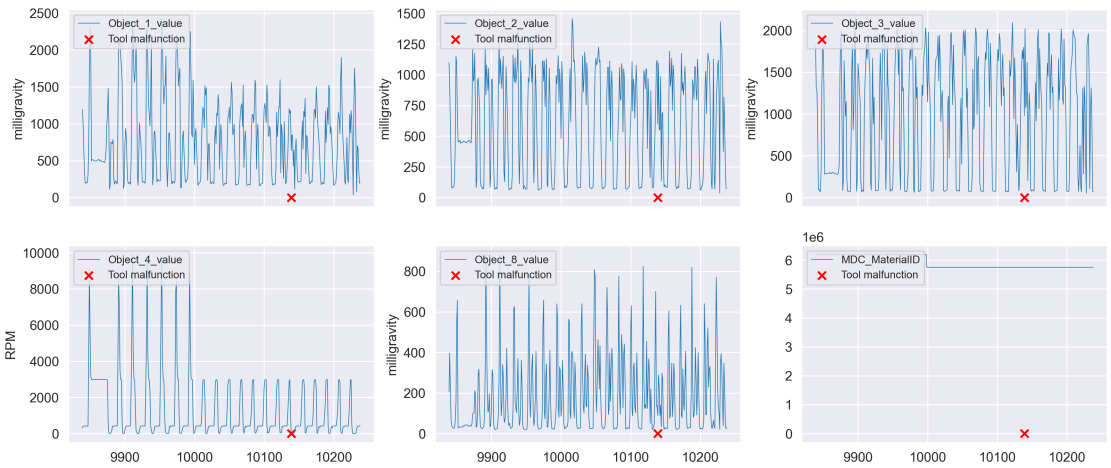
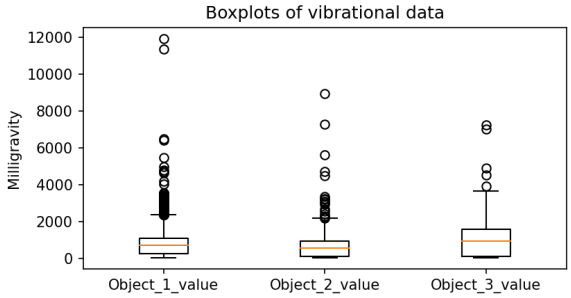
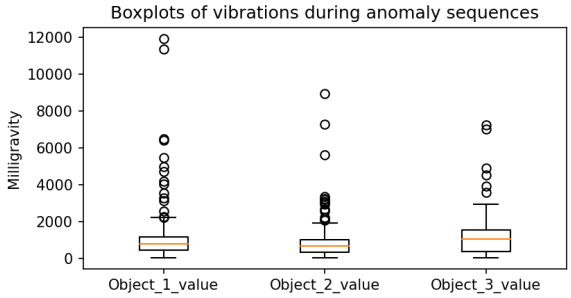


Figure 3.3: Time series data around a malfunction, without outlier.



(a) Boxplot of the vibration signals on full dataset.



(b) Boxplot of the vibration signals in close relation to malfunctions.

Figure 3.4: Boxplots for the vibration signals in the full dataset and in close relation to malfunctions.

3.3 Data Preprocessing and Feature Engineering

To effectively use the extracted data in a modeling setting, multiple preprocessing steps need to be applied. Pandas library in Python is a useful tool when working with large datasets. The preprocessing steps include data cleaning, feature extraction, feature scaling and feature selection. A description of these steps is provided in this section.

3.3.1 Data Cleaning

From the company database, only data during production or when an alarm indicates a malfunction is extracted. Also, data not corresponding to the five tools 2112, 2345, 4304, 4503 and 4507 are removed. Additionally, cleaning includes manipulating or removing faulty or uninformative data. Any row containing `NaN` value for the vibrational data is removed, corresponding to logs prior to sensor installation. The vibrational data is the core data and thus accuracy of such is important, hence the missing values are not replaced with any imputed value, such as the mean or median. However, for other features, missing or faulty data are imputed with appropriate values. For instance, a missing material ID is replaced with -1, since it is a categorical variable.

Due to the characteristics of the data extraction, coherent timestamps after a malfunction hold the same information as the first row when the malfunction is detected until the tool is replaced and production restarted. Thus, such rows are removed.

Additional cleaning is performed after the train and test split is performed. Since any anomaly detection model should be trained on normal data, it is important that the training set does not contain faulty values. Thus, extremes and periods with missing data logs, resulting in an almost constant interval, denoted *bridge*, are removed. An example is seen in Fig 3.5 where both extremes and a bridge are removed, displaying the difference between the original and cleaned data.

3.3.2 Feature Extraction

New features are created based on the original data to simplify relevant information extraction from the dataset. The column `ToolMalfunction` is created and has value 1 if any of the five alarm IDs holds the value 3190, else it is 0. Also, a feature `NewSequence` is derived based on the time gap between two consecutive rows when isolating the data on tool level. It is 1 if the gap exceeds one second, else 0. The reason is due to the condition check after each use of a tool. A larger time gap than one second would imply that another tool has been used in between or that the production was idle. Theoretically, both cases would infer that laser measurements have been performed and the condition of the tool is known. Thus, when a malfunction is detected, the interval where the actual breakage can have happened is limited to the time between the latest new sequence and the detection of the malfunction.

To evaluate anomalies detected, a column `AnomalySequence` is constructed with

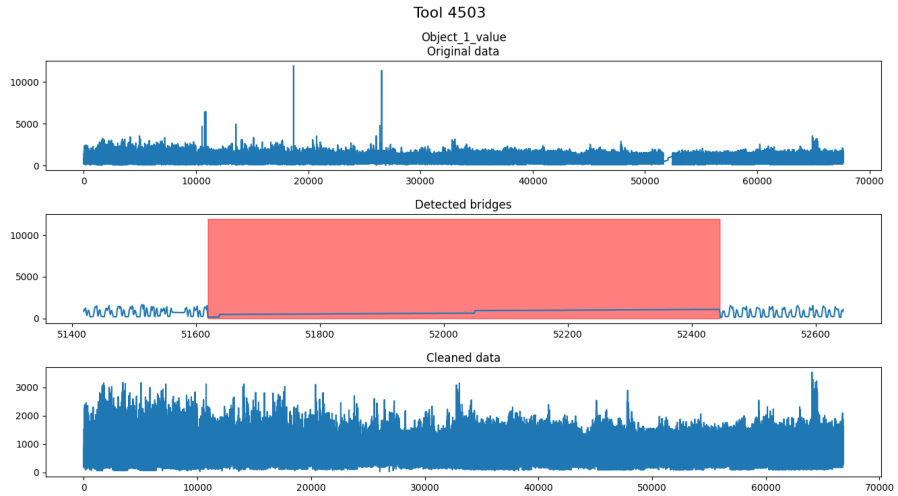


Figure 3.5: Example of how the two cleaning steps can be applied to the data, removing extreme values and bridges.

the value 1 for a sequence containing a malfunction, that is all rows between an indication of a malfunction and its corresponding new sequence. Otherwise the value is 0.

3.3.3 Feature Selection

To exploit the information stored in the data in the best way it is necessary to understand dependencies and relations between features. To reduce memory allocation and achieve computational speed-up, a subset of the features are selected. The key signals are the vibrational data, which are to be reconstructed or forecasted. In addition, features that might influence the behaviour of the vibrations are also extracted. By visualizations, material ID is observed to infer a change in behaviour characteristics of the vibrations. Fig. 3.6 displays an example of the mentioned relation. Thus, material ID is added to the selection of features. Also, the spindle RPM is selected since it determines the strain of the tool.

3.3.4 Data Transformation

The selected features are transformed in appropriate ways to suit a machine learning model. Each feature except material ID is standardized, according to $\widetilde{x}_i = \frac{x_i - \mu_i}{\sigma_i}$, where μ_i and σ_i are the mean and standard deviation of feature i.

For material ID, scaling the values is not adequate since the information does not lie in its magnitude. Material ID is a categorical variable, where each discrete value is unrelated to the other values. Thus, a binary representation is created using one-hot encoding.

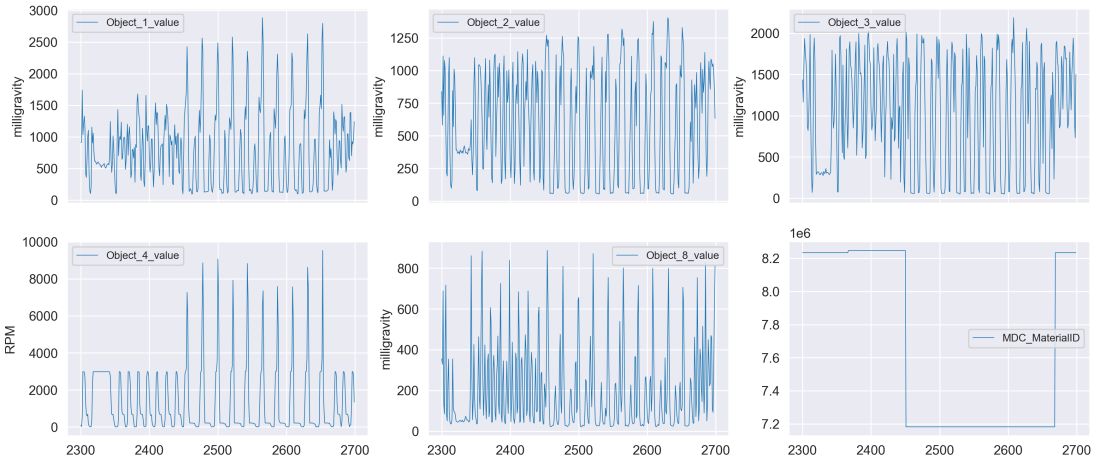


Figure 3.6: A sample sequence showcasing the impact Material ID has on the vibration signals.

3.4 Model Comparisons

The models that will be evaluated are a baseline model, an ARIMA model, an autoencoder, and MTAD-GAT. To compare the performance between different models, a general setting is needed to proceed from. In this section, the general setting is described, followed by the specific settings for each model.

3.4.1 General Description

The comparison between models is conducted on the data from a single machine and tool, specifically machine 2 tool 4503, since it has the largest record of tool breakages. The behaviour around the malfunctions is of different characters, displayed previously in Fig. 3.2 and 3.3. For this specific tool, 7 malfunctions are preceded by a high spike, similar to the first figure. The remaining 10 are of a different character, not preceded by an extremely large value. Thus, it is determined to be the best subset of the data to perform the comparison on, exposing the models to multiple malfunction scenarios. Before comparing between models, the optimal settings for each model are determined based on the mentioned dataset.

In general, the data is split into training and testing sets, with 50% of the data in each. The models are trained on a normal representation of the data. To achieve this, the data is split in such a fashion that the training data contains as few malfunctions as possible, still being coherent. Even though this can create a discontinuity in the test set, this is an efficient way to keep as many malfunctions as possible in the test set while minimizing the number of discontinuities in both sets. Any malfunctions still within the training set are removed together with the previous 300 time steps, to be sure of not keeping any data point related to a malfunction. On average, the sequence length for the selected data set is 20 seconds. This means that this tool is

being used on average 20 seconds at a time. The testing data is transformed based on the parameters obtained from transforming the training set.

For univariate models, the vibrational data in x-, y- and z-joint are used, respectively, as input and output. For multivariate models, all features are used as input features simultaneously, outputting the vibrations mentioned above. For each output feature and time step, an anomaly score is computed, as the difference between the output and the true value. In a reconstruction and forecasting scenario, the sum of the errors constitutes the score. Different approaches are investigated, one where the mean of the errors for all features within each time step constitutes a *global anomaly score* and another where the anomaly scores are kept separated for each feature. The errors are compared to a threshold, specific for the selected threshold method, and labeled as an anomaly if exceeding the threshold. The labels for the individual features can be aggregated by an AND or OR operation. For the AND operation, only time steps where the score of each feature exceed the threshold will be labeled as an anomaly. For an OR operation, it is sufficient if only one feature is classified as anomaly to classify the sample as anomalous.

The true labels are constituted of the anomaly sequences, where all instances within each sequence are labeled as anomalous. In addition to precision, recall, and F_1 scores, the *heads-up*, that is, the average time between the first detected instance in an anomaly sequence and the time of actual tool malfunction detection, are measured.

For each model, we first compare the general capabilities in ROC and PRC plots. This will indicate how the models perform in terms of classification accuracy and precision-recall trade-off. The area under the ROC curve will provide insights into the overall ability to discriminate between positive and negative samples, while the area under the PRC curve will indicate the performance in terms of precision and recall balance. Thresholds regarding the pointwise global anomaly scores and a rolling median approach, MAD, are investigated. Then, we look at the most promising models and the baseline model, evaluating them in the use-case of detecting anomaly sequences in production. For this case it is sufficient to flag one time step in a sequence as anomalous to accept the anomaly sequence as detected. Hence, in such a scenario a TP reflects a detected sequence, not an instance. The potential of each model is compared by evaluating different metrics. Primarily, a high F_1 score is wanted, but also a model able to find many anomaly sequences (high recall) with reasonably few FPs is investigated to study the behaviour of FPs in an attempt to increase the precision.

3.4.2 Baseline Model

The baseline model is created as a univariate model that identifies all values in the test set that exceeds the largest training value as anomalies.

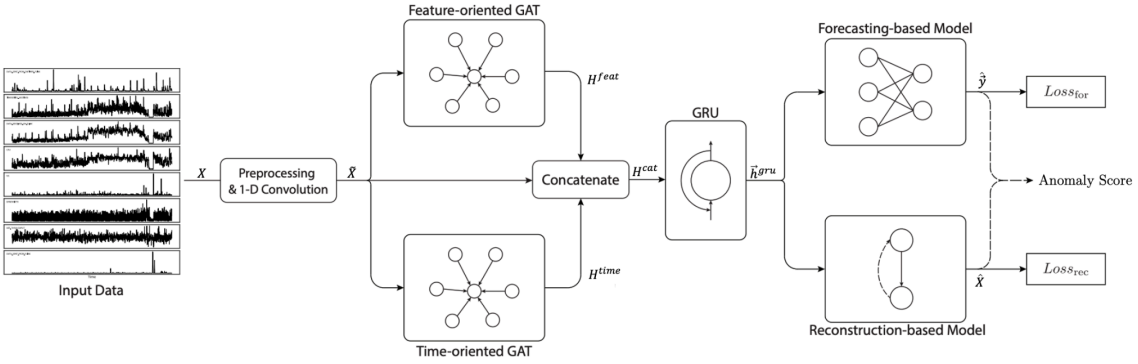


Figure 3.7: MTAD-GAT architecture, image from [14].

3.4.3 MTAD-GAT

MTAD-GAT is a deep learning-based framework for anomaly detection in multivariate time-series data, with the aim of both reconstructing and forecasting the input data [14]. The implementation of the model was obtained from an open source GitHub repository[1], with MIT license. This license grants the freedom to use, modify, and distribute the code for commercial purposes, while ensuring that the original copyright notices and disclaimers are included in any copies or modifications of the software.

Comparing to the original model architecture, the VAE is removed and instead the GRU layer is regarded as an encoder layer and an RNN decoder layer is added to reconstruct the data. Also, modifications are made to enable feeding the model with all features but only using the subset of output-features as targets for forecasting and reconstructing.

The multivariate time series data is transformed into a sliding window dataset to be fed into the model. Specifically, overlapping sub-sequences of fixed-length are created by splitting the data according to the sliding window with stride one. Each sub-sequence constitutes a sample to use during training or testing. By doing so, temporal dependencies within each window could be captured. The size of the sliding window is determined empirically.

The transformation is made using utilities in `PyTorch` using the `Dataset` and `DataLoader` classes. First, the time series data is loaded into a `PyTorch` tensor. Then, we use the `Dataset` class to create a dataset of sliding windows. We use the `DataLoader` class to create an iterator over the dataset, which allows to efficiently load and process the data in batches during training.

The model expects the data to be on sliding window form, with shape (batch size, window size, number of features). Going from left to right, a filter convolves over the input data layer filtering out noisy and irrelevant information. Then in parallel, the input data is transformed to a time oriented and feature oriented graph respectively, where the nodes in the graph represent data points and the edges represent relationships or connections between the data points. The resulting dependencies

[1]https://github.com/ML4ITS/mtad-gat-pytorch

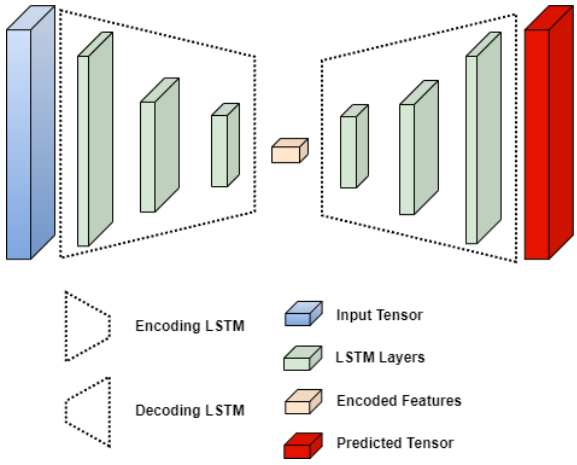


Figure 3.8: A symmetrical LSTM autoencoder architecture with 3 LSTM layers with sizes 64, 32 and 16.

are concatenated together with regular input data and fed into a GRU layer to capture long term dependencies. The output from the long term dependencies are fed into forecasting and reconstruction based modules in parallel. The reconstruction module recreates the data to the same shape as the input data, while the forecasting module predicts future values. Separate losses are computed for the reconstruction and forecasting modules in the model. The model is optimized using a joint loss function, enabling simultaneous learning of both reconstruction of the past and forecasting of the future. The selection of hyperparameters is visible in Tab. B.1, in Appendix B. With this selection of hyperparameters, the architecture has around 400 000 trainable weights.

3.4.4 LSTM Autoencoder

The autoencoder model is a reconstruction-based model and takes the same input and output features as MTAD-GAT. The shape of the input and output data also corresponds to the same shape described previously. In order to compare the performance of the models incorporating reconstruction, the same window size for the autoencoder is used as for MTAD-GAT. The autoencoder is implemented in `Keras`, consisting of an encoder with 3 LSTM layers, with sizes 64, 32 and 16. The bottleneck is represented by a `RepeatVector` layer. This layer repeats the output of the previous layer (i.e., the last LSTM layer in the encoder) multiple times, so that the resulting tensor has the same shape as the expected input of the decoder. The decoder also consists of 3 LSTM layers, with sizes 16, 32 and 64. An overview of the architecture can be seen in Fig. 3.8.

The model is trained using the Adam optimizer with a learning rate of 0.001. The training process lasts for 200 epochs. Early stopping is implemented, which means that the training will stop if the weights have not improved for 25 consecutive epochs. In such cases, the best weights achieved so far will be saved as the final model. Furthermore, this architecture has roughly 80 000 learnable parameters.

3.4.5 ARIMA

To build the ARIMA model, the Python packages `pdmarima` and `statsmodels` are used.

One model per feature is fitted using the training data. The parameters (p, d, q) are empirically found to be $(1, 0, 1)$, $(2, 0, 1)$ and $(3, 0, 3)$ for each feature, respectively. For each model, a rolling forecasting approach is utilized, meaning that for each time step in the test set, the value of the next time step is forecasted and the true value is appended to the model. This process is repeated iteratively until all the values in the test set have been predicted.

3.5 Evaluation

After comparing the performance of the different models, we select the best performing one to further investigate its generalization capabilities. Specifically, we train the selected model on data from one tool and evaluate its performance on data from another tool or machine. This shows the ability of the model to detect anomalies across different equipment and operating conditions.

The results obtained from this analysis will be evaluated in the context of the current workflow and decision-making processes of the company. For example, we will assess the potential benefits of using the proposed model to detect malfunctions earlier than the current approach and how this may translate into improved equipment reliability, reduced downtime, and cost savings.

3.6 Potential Pitfalls in the Data

When working with data from a real-world scenario, it is important to be aware of potential pitfalls that may affect the performance of the models. For this project, pitfalls can be related to the available data or the interpretation and implementation of production procedures.

One potential source of error is related to the material ID. As shown in Fig. 3.6, material ID affects vibrations. However, analyzing the training and test sets, only between 23% to 63% of material IDs within the test sets have been seen during training. For unseen material ID, the models can not utilize any relation between that material and the vibrations, possibly having a negative impact on the performance.

Another potential issue is if the data contains inaccuracies such as outliers or bridges that are not related to malfunctions. Mentioned data are not removed from the test set since the test data should reflect a real-world scenario and removing it would falsify such an objective. However, such data will reasonably raise false alarms, affecting the results of the model.

Regarding the interpretation of production procedures, the actual break point for a tool should only be able to occur within the last sequence before a detected malfunction. However, suspicious behaviour can be seen in Fig. 3.9 where a vibration signal

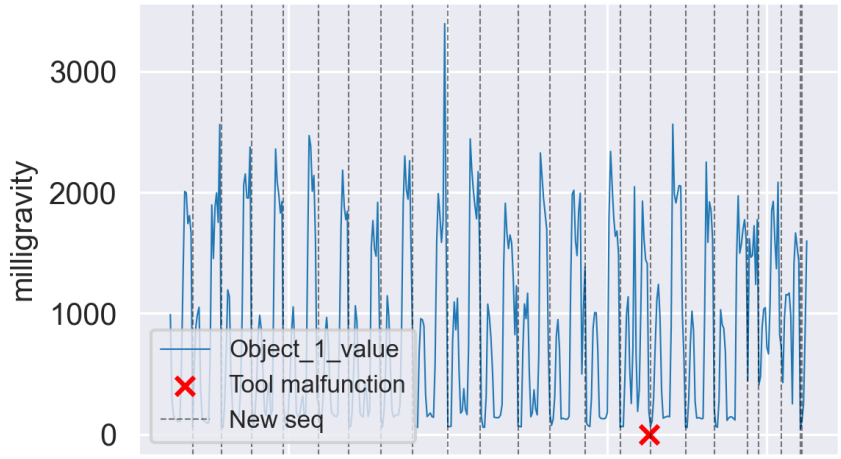


Figure 3.9: A sample from a vibration signal acting suspicious a couple of sequences before the malfunction was found. The dashed lines indicates a new sequence.

has an abnormally high value a couple of running cycles before the actual malfunction was found. If this behaviour in fact is related to the upcoming breakage, the prerequisites change.

4

Results

In this chapter, the results regarding the prospects of each technique, metrics outputted from the models and differences between the current workflow and a scenario where the model is used, are presented.

4.1 Comparison of Approaches

In this section the analysis of which approach should be used to try to predict malfunctions is presented. To build a regression model and predict RUL, labels of the true remaining life are needed. For anomaly detection, high quality data is required.

4.1.1 Predicting RUL

The labeled data needed for a supervised scenario should be the true remaining life, that is the time until malfunction, for any tool at any given point. Such data does not exist and needs to be manufactured. To achieve this, data of the exact time of malfunction, when a new tool is inserted, and which of the master and sibling tool is operating are needed. For evaluation purposes, values of the current estimate of RUL are wanted. At our disposal we have the features `PDA_Tool_Life_Counter`, `PDA_Maximum_Tool_Life` and `PDA_Remaining_Tool_Life`, from here on denoted t_{lc}, t_{ml} and t_{rl}, respectively.

The t_{ml} holds the expected life length, assigned for each tool. The t_{lc} counts the time that a tool has been used in production. The difference between t_{ml} and t_{lc} constitutes t_{rl}, All values are presented in seconds. From discussions with the company, it is assumed that a new tool should be indicated by a reset of t_{lc}, that is $t_{lc} = 0$. Also, switching between the sibling and master tool should be indicated by a discontinuity, from where the life counter then increases from. By starting from each malfunction and calculating backward until the insertion point of that tool is encountered, taking any switches between tools into consideration, the true RUL can be obtained.

However, this approach is infeasible. To start with, the exact time of malfunction is unknown. Only the time for detecting the malfunction is collected, hence only approximations of the true remaining life can be manufactured. In addition, the t_{lc} is too unpredictable to even make useful approximations. It does not incorporate

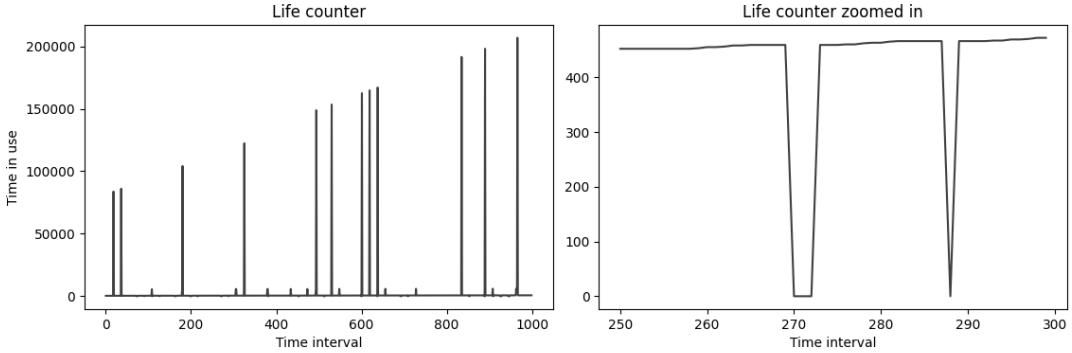


Figure 4.1: The life counter for a contiguous period when the machine is in production, for a specific tool. The left figure portrays the extremely large values and the right figure demonstrates temporary resets of the counter.

Table 4.1: Frequency for different values of maximum life time for the chosen, contiguous sequence of 1000 points.

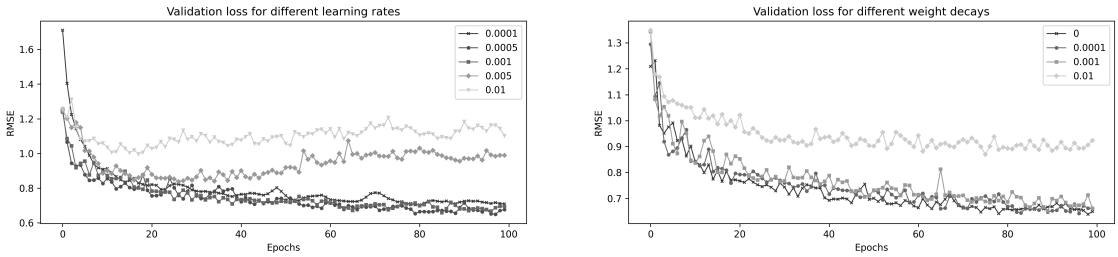
Value	Frequency
1800	965
0	15
9000	12
18000000	8

the expected, piece-wise linearity and t_{ml} is not constant, for a specific tool. To demonstrate this, data from an active production, sorted on tool level, is extracted.

For a representative, contiguous sequence of 1000 seconds, t_{lc} contains multiple abnormally high values, displayed in Fig. 4.1. These high values could correspond to faulty data recordings. During the selected sequence, the counter is also zero multiple times, which would indicate a new tool. However, Fig. 4.1 proves that the counter is only temporarily reset, since the following values correspond to those prior the reset. By the construction of the signal, this infers that invalid data has occurred. Even removing such data would not result in the expected behaviour, since t_{lc} only changes when the tool is actively operating on the material and not when production signal is high.

Despite the expectation of a constant t_{ml}, it shows unexpected fluctuations and its frequency is presented in Tab. 4.1. According to expert knowledge, the true value should be 1800 and the remaining values could be due to unsynchronized data recordings, inheriting the values from preceding tools, or manual overrides performed by operators.

Following from the demonstrated behaviour of t_{lc} and t_{ml}, t_{rl} has equal flaws. Altogether, manufacturing labels of high quality is infeasible, disqualifying the possibility of RUL prediction.



(a) Learning rates.

(b) Weight decays.

Figure 4.2: Validation loss for MTAD-GAT trained on tool 4503 and machine 2 for different learning rates and weight decays.

4.1.2 Anomaly Detection

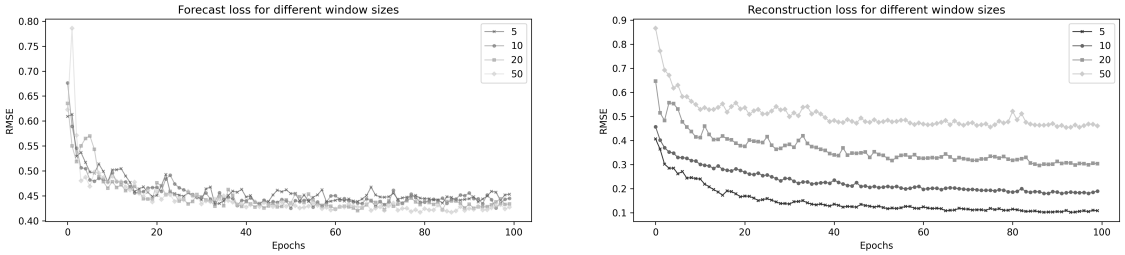
The requirements to successfully implement an anomaly detection model are not as extensive as for a model predicting RUL. The necessary time-series data exists in terms of vibrational data. This data should be reliable and complete. After processing the data as described in chapter 3, this is assumed to be fulfilled. Data regarding the actual life of a tool and the estimated remaining life is not necessary. A true label of anomalies eases the evaluation and can be constructed by utilizing the stored points when a tool malfunction is discovered together with the created feature `NewSequence`, as described in 3.3.2. By acting accordingly, all requirements for using anomaly detection are fulfilled.

4.2 Model Comparison

In this section, the performance of MTAD-GAT, Autoencoder, ARIMA and a baseline model in detecting anomalies will be compared on D1. After tuning parameters, we first compare the general capabilities of the models in ROC and PRC plots. Then the most promising models are evaluated from a use-case scenario.

4.2.1 Hyperparameter Tuning

For MTAD-GAT, empirical studies are performed to determine the values for learning rate, weight decay and window size. The loss is compared for different parameter values and generally the value resulting in the lowest loss is chosen. The learning rate and weight decay parameters are set to 0.001 and 0, respectively, based on the findings displayed in Fig. 4.2. For different window sizes the reconstruction and forecast loss are displayed individually. Using a smaller window size does not influence the forecast ability, illustrated by Fig. 4.3a. However, using fewer points during reconstruction reasonably generates a lower loss, verified in Fig. 4.3b. Even though a window size of 5 provides an overall lower loss, the results for window size 10 are also included to avoid filtering out any distributional properties detected with a larger window size.



(a) Forecast loss.

(b) Reconstruction loss.

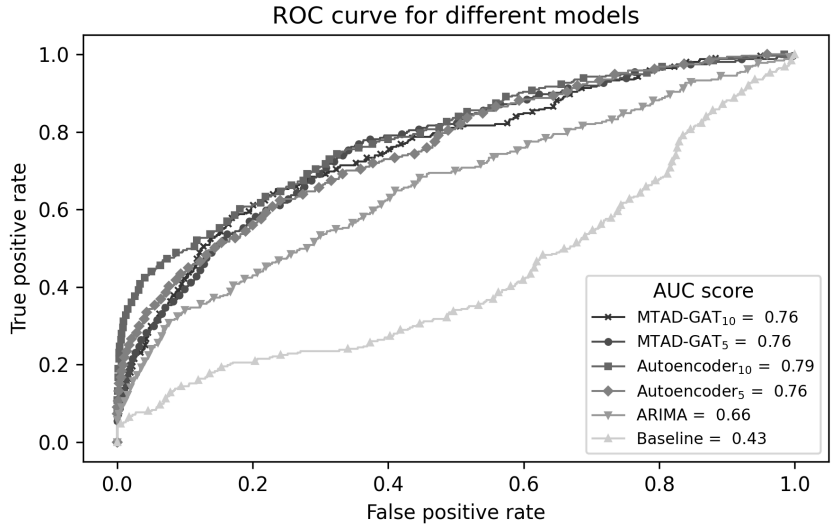
Figure 4.3: Forecast and reconstruction loss, respectively, for MTAD-GAT trained on tool 4503 and machine 2 for different window sizes.

4.2.2 ROC and PRC

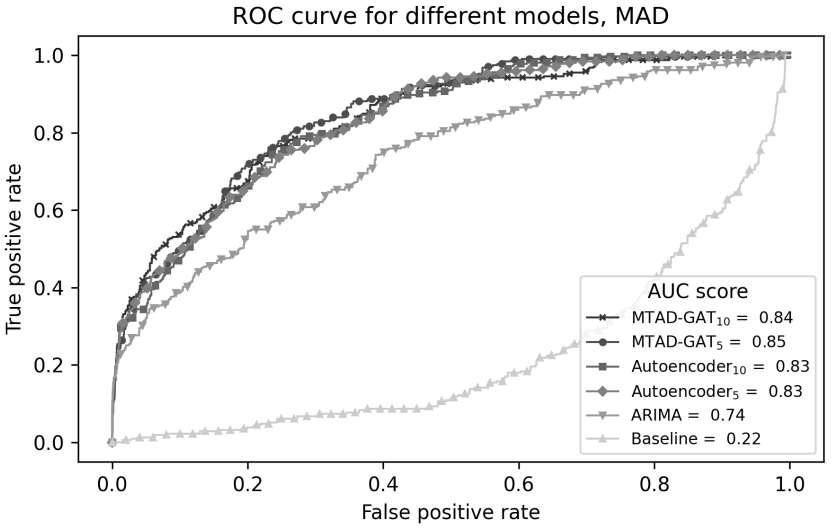
For ROC and PRC plots, the global anomaly scores and rolling MAD scores are evaluated respectively. Each point is classified as positive if its error exceeds the threshold, regardless of whether other points within the same anomaly sequence have been detected or not. In total, there are 311 points corresponding to the 17 anomaly sequences. For ROC, this provides a general investigation of how accurately the models can separate between the points corresponding to the anomaly sequences and normal points. Observe that this approach implies an assumption that all points within the anomaly sequence are anomalous in order to enable detection.

Fig. 4.4 displays the results for both global errors and MAD errors. Closer inspection of 4.4a indicates that autoencoder$_{10}$ is most profound in separating the classes for the global errors, even though the performance between all MTAD-GAT and autoencoder models are comparable. The results indicate that a fairly high share of TPs can be detected at a low FP rate, using these models. The baseline, comparing the actual vibration values to the maximum value seen in training, is outperformed by all other models, indicating that solely relying on large vibrational values is not sufficient. Focusing on the ROC for MAD errors, displayed in Fig. 4.4b, the same models are eminent, with an increased AUC score for all models except the baseline. This implies that forecasts and reconstructions during anomaly sequences are generally exposed to higher distributional errors. On the other hand, the median values of the vibrations are not explicitly increasing during anomaly sequences, as proved by the reduced AUC score for the baseline. However, it should be noted that the good performance of the ROC plots should be attributed to the highly imbalanced nature of the dataset, proved by the PRC plots.

The PRC plots, visualized in Fig 4.5, show the trade-off between precision and recall for the same errors as in ROC. Here the AUC scores are clearly reduced, proving the effect the imbalanced dataset provides. Since the number of negative (normal) samples are much larger than the positive ones (anomalous), even a small fraction of FPss will heavily punish the precision rate since there are so few TPs. This is illustrated in both PRC cases, for global anomaly scores (Fig. 4.5a) and MAD errors (Fig. 4.5b), where both precision rates approach zero for a low recall score. Overall, autoencoder$_{10}$ demonstrates the best performance amongst the models. It yields a



(a) For global anomaly score.



(b) For MAD errors.

Figure 4.4: ROC for all models. The subscripts in the legends indicate the window size for reconstruction.

higher precision rate for recall scores up until approximately 0.4. However, from the figure it is impossible to validate which sequences are detected.

ROC and PRC provide general transparency to the models' capability, but it is also interesting to evaluate the models from a use-case perspective, analyzing how well and which anomaly sequences are detected by the models.

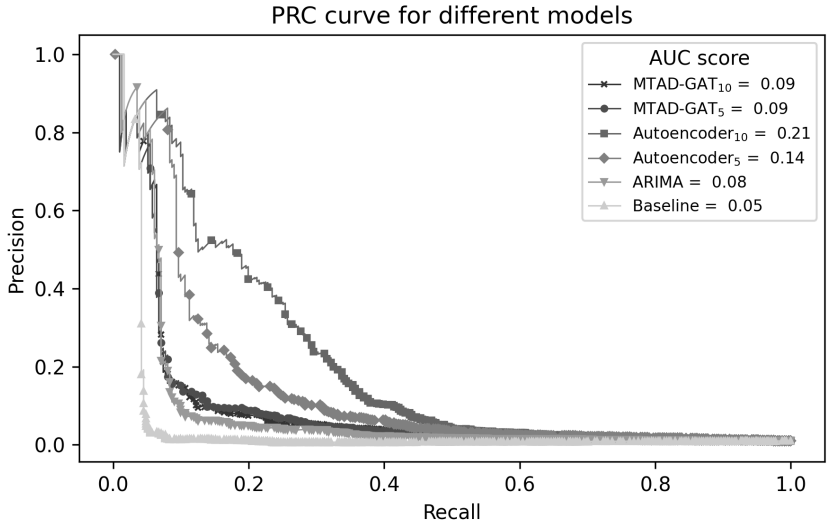
4.2.3 Use-case Scenario

From a use-case perspective, the number of detected anomaly sequences are of interest rather than individual instances. In Tab. 4.2 a comparison between MTAD-GAT$_5$, autoencoder$_{10}$ and the baseline is presented. The settings for each threshold method are distinguished by the subscripts. A threshold with subscript $feat$ classifies the instances for each feature, aggregating the final classification by an AND or OR operation. The *global* subscript indicates that the classifications are obtained by using the chosen threshold method on the global anomaly score. For the ϵ-method, the subscripted number denotes how many standard deviations are being used. The table also presents an enumeration of the detected sequences, to evaluate which type of sequences can be detected, in comparison with group 1 and 2.

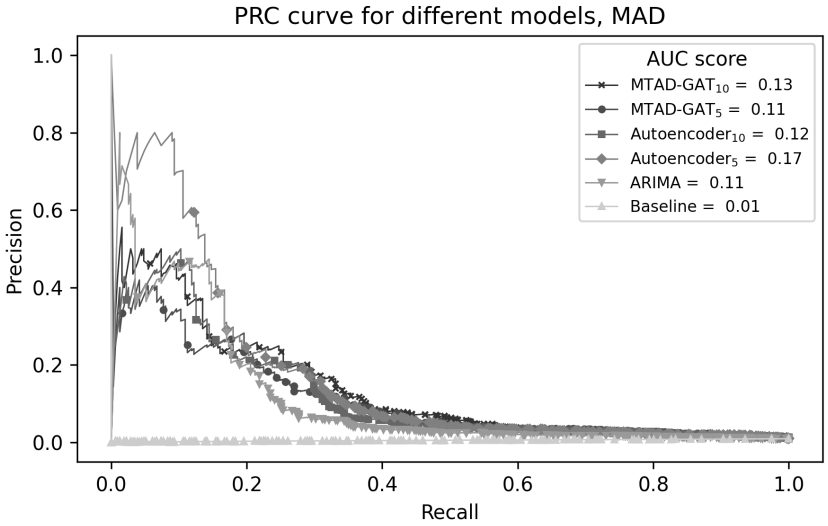
For the thresholds yielding the highest F_1 score for each model, respectively, both the baseline and MTAD-GAT only detect sequences in group 1. The heads-up score is the same across both models, indicating that both probably detect the same values within the anomaly sequences. MTAD-GAT avoids some of the FPs that the baseline classifies, but in general the performance is comparable. The autoencoder$_{10}$ using max$_{global}$ obtains the highest F_1 score, managing to detect 6 anomaly sequences from group 2, in addition to all in group 1. This could indicate that there are large enough distributional errors within anomaly sequences belonging to group 2 that the model is capable of detecting.

Lowering the threshold to detect all sequences results generally in an increment in FPs. Also, such an action could incorrectly classify instances in the training set as anomalies. The baseline has to use 1 standard deviation, while the same number for MTAD-GAT and the autoencoder are 6 and 9, respectively. The low threshold needed for the baseline proves that the anomaly sequences in group 2 does not contain any relatively large values. As a result, the baseline misclassifies thousands of instances. The higher ϵ-values for MTAD-GAT and the autoencoder informs us that larger errors within all anomaly sequences exist. However, in terms of FPs, the autoencoder outperforms MTAD-GAT, classifying only 93 points wrongly as positive. However, using this threshold yields multiple FPs in the training data, disqualifying it from further investigation.

Using MAD$_{global}$ as threshold ensures that no FP within the training data is detected. In the test set, 12 sequences are detected together with 45 FPs. Worth noticing is that sequence 8, belonging to group 1, is not detected. This could indicate that sequence 8 consists of short, if any, sequences of succeeding unexpected values. Like the performance using max$_{global}$, this threshold can successfully identify multiple anomalies belonging to group 2 without an explosion of FPs.



(a) For global anomaly scores.



(b) For MAD errors.

Figure 4.5: PRC for all models. The subscript in the models' name indicates the window size for reconstruction.

Table 4.2: Model evaluation in a use-case scenario, for different thresholds.

Model	Threshold	Agg. operation	TP	FP	Precision	Recall	F_1	Heads-up	Detected seq.
Baseline	$\max_{global}$	-	7	5	0.58	0.41	0.48	8.57	4, 6-8, 11, 14, 15
	$\varepsilon_{1,global}$	-	17	2880	0.01	1.00	0.01	12.00	0-16
MTAD-GAT$_5$	$\max_{feat}$	AND	7	2	0.78	0.41	0.54	8.57	4, 6-8, 11, 14, 15
	$\varepsilon_{6,global}$	-	17	630	0.03	1.00	0.05	13.65	0-16
Autoencoder$_{10}$	$\max_{global}$	-	13	10	0.57	0.76	0.65	9.38	0, 4-15
	$\varepsilon_{9,global}$	-	17	93	0.15	1.0	0.27	9.88	0-16
	MAD_{global}	-	12	45	0.21	0.71	0.32	4.67	0, 4-7, 9-15

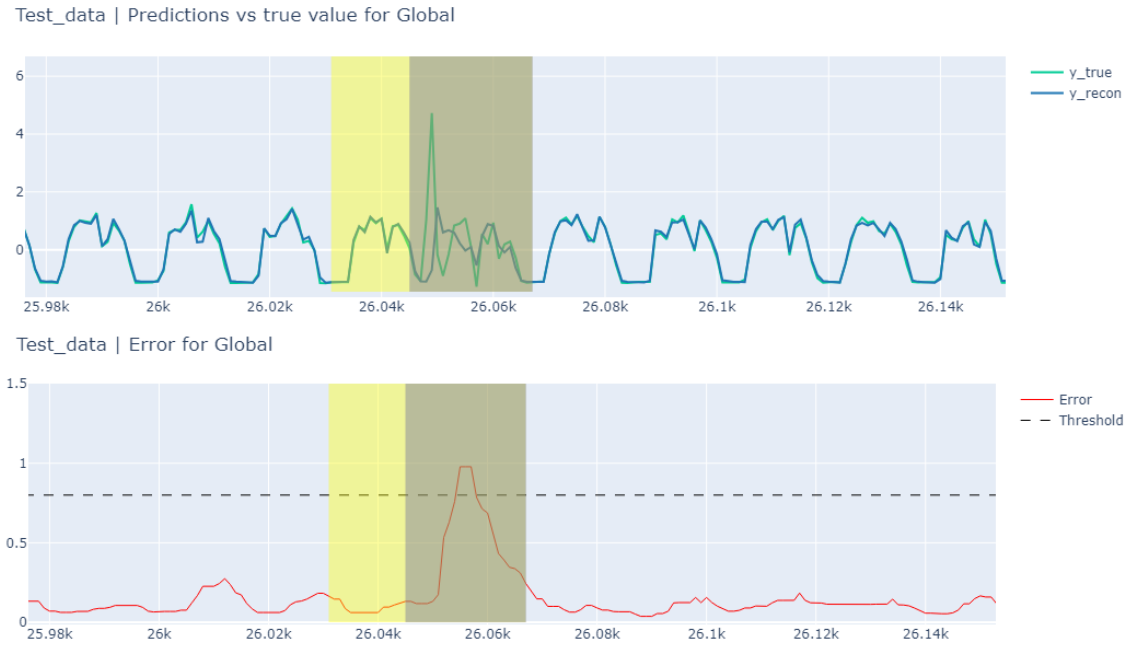


Figure 4.6: True positive classification.

Visualizing the reconstructions for cases where TP, FN and FP occur with the autoencoder can further explain performance and areas of improvement. In Fig. 4.6, the reconstructed values and the true values, along with the corresponding error between them, are presented. The yellow rectangle represents the true anomaly sequence, while the blue rectangle indicates predicted anomalies. Hence, overlapping rectangles constitute a detected anomaly. The rectangle areas are extended by 10 points on both sides for visual clarity. It is evident that the autoencoder accurately reconstructs the data, and deviations from the expected pattern result in higher errors yielding positive predictions. However, some of the sequences are not detected, especially sequences belonging to group 2. The reconstruction errors of such sequences are visualized in Fig. 4.7. Despite not surpassing the threshold, the errors for some of the sequences are noticeably different from the rest.

When depicting FP classifications, there are cases where these instances occur related to a true anomaly sequence. Examples can be seen in Fig. 4.8. Such behaviour can be found in both MAD and pointwise errors, raising the cause for further FP analysis in the use-case scenario.

4.2.4 False Positive Analysis

In the use-case scenario a collective approach is used, treating every cluster of FPs as a singular entity since TPs within anomaly sequences are counted likewise. In addition, FPs that directly coincide with true anomaly sequences should be disregarded. Based on these premises, precision, recall and F_1 score are recomputed for the autoencoder with $\max_{global}$ and MAD_{global} as chosen threshold methods. The results can be seen in Tab. 4.3. With this interpretation, a substantial decrease in

Figure 4.7: False positive classification.

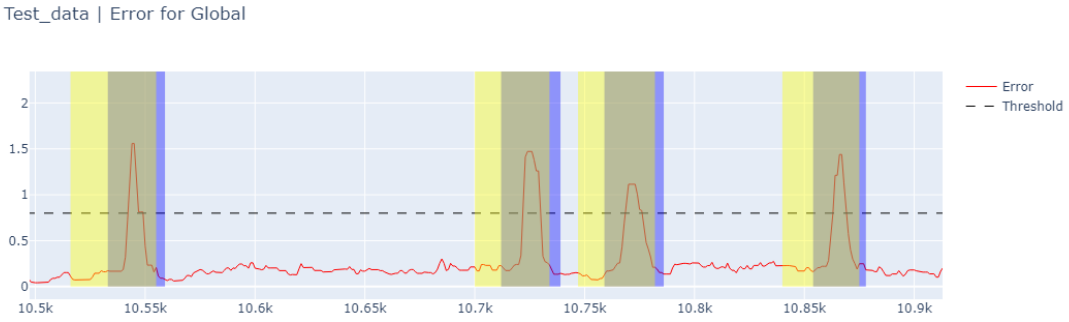


Figure 4.8: Reconstruction errors and threshold. FP classification near true anomaly sequences.

Table 4.3: Re-evaluating performance, disregarding FP sequences and FPs in close relation to true anomaly sequences.

Model	Threshold	TP	FP	Precision	Recall	F_1	Heads-up	Detected seq.
Autoencoder_{10}	$\max_{global}$	13	6	0.68	0.76	0.72	9.38	0, 4-15
	MAD_{global}	12	3	0.80	0.71	0.75	4.67	0, 4-7, 9-15

Table 4.4: False positives (true positives) for each feature for autoencoder_{10}.

Threshold	Vibration x-joint	Vibration y-joint	Vibration z-joint
$\max_{global}$	19 (13)	1 (10)	6 (11)
MAD_{global}	3 (11)	1 (12)	11 (11)

falsely predicted anomalies is shown for both errors resulting in an improvement in F_1 score with 9 and 43 percentage units for $\max_{global}$ and MAD_{global}, respectively.

Both thresholds are based on the mean error from all features. Refining the search to a more granular analysis displays the number of FPs detected individually for each vibration signal. The results are presented in Tab. 4.4. For both thresholds the vibrations in the y-joint result in fewer FPs than its companions. For MAD, solely that signal is sufficient for achieving the same recall as taking all signals into consideration. Even if this is not the case for $\max_{global}$, this might indicate that vibrations in the y-axis are generally more stable during normal data and hence the errors are more significant in contrast.

4.3 Generalization

To further investigate the appropriateness of the model, we trained autoencoder_{10} specifically for the selected subset of tools on both machine 1 and machine 2. Subsequently, the precision, recall, and F_1 score are computed for the aforementioned use-case. The corresponding scores can be found in Tab. 4.5, which highlights the best configuration only.

The various results presented in the table describe the complexity of the problem.

Table 4.5: Performance evaluation for the subset of tools selected for machines 1 and 2.

Machine	Tool	FN	TP	FP	Precision	Recall	F_1
1 (2)	2112	9 (2)	1 (5)	3 (39)	0.25 (0.11)	0.10 (0.71)	0.14 (0.20)
1 (2)	2345	8 (6)	0 (0)	7 (1)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
1 (2)	4304	3 (3)	3 (2)	0 (0)	1 (1)	0.50 (0.40)	0.67 (0.57)
1 (2)	4503	4 (5)	2 (12)	3 (3)	0.29 (0.80)	0.33 (0.71)	0.31 (0.75)
1 (2)	4507	3 (0)	5 (1)	1 (3)	0.83 (0.25)	0.63 (1)	0.71 (0.33)

The characteristics of the data can vary heavily even on tool level, being the reason for not detecting any of the anomaly sequences for tool 2345. However, FPs are held at a relatively low level for all combinations. Seeing that the model performs well on tools 4503 and 4507 we investigate the generalization capabilities of the model.

Training on machine 1 tool 4503 and testing on machine 2, tool 4503 (D1), yields only predictions in group 1. Similarly, training on the largest dataset (machine 1, tool 4507) and testing on D1, the model finds only sequences from group 1. Clearly, an approach like this is not sufficient to detect distributional errors similar to the ones in group 2.

4.4 Model versus the Current Workflow

One crucial aspect that is measured during the evaluation of the performance is the heads-up time. This metric captures the time duration between the moment the model detects an anomaly and when the machine detects the malfunction.

The heads up are different for each tool since the usage of each tool is different resulting in various sequence lengths. Therefore, these numbers are not comparable between tools since they do not operate under the same conditions. However, the largest heads-up is seen for machine 2 and tool 2112 where we find the anomalies on average 100 seconds prior with respect to the current workflow. For 4503, where it is shown that a sequence length is on average 20 seconds, the heads up is on average 9.5 seconds for max threshold and 4.67 seconds for MAD.

In total 31 out of 74 malfunctions are detected in combination with 60 FPs, correctly classifying more than 300 000 instances as non-anomalous.

5

Discussion

In this chapter we discuss the results that were obtained, possible improvements and future work. The discussion is held in the light of the main research question: how the production at Sandvik Coromant AB can be improved by using collected time-series data. It ends with a conclusion.

The main research question was further divided into three sub-questions regarding the feasible strategy to approach an industry 4.0 scenario, which model is most prominent for the selected approach, and finally the improvements that are achievable. Each sub-question is separately answered following the related results. However, all aspects have not been possible to cover, leaving room for further studies and potential improvements.

5.1 Approaching Data-Driven Decision Making

Based on the properties of the data stored by the company, the feasibility of two different strategies to reach a more data driven production environment has been investigated: regression-based RUL prediction and anomaly detection. The requirement for supervised learning in RUL prediction imposes stricter constraints on the data. Even though the prerequisites were stated to exist for a RUL scenario, the data quality did not align with the expectations. Consequently, RUL prediction was determined not to be a possible approach as of the current state. Therefore, the study concludes that anomaly detection is the feasible strategy for leveraging the available time-series data.

RUL prediction would enable a constant monitoring possibility, hence it is an interesting approach. To create a situation where the prerequisites for such a model are fulfilled, specific requirements on the data need to be addressed. To manufacture a label, the time for the actual tool breakage is required (the time of detection can only serve as an approximation). In addition it needs to be clearly stated which version of the tool is currently utilized. One suggestion is to add an additional signal or modify the tool ID to state which of the sibling and master tool is being used. The reason for this is that when manufacturing the label, it has to be constituted from a malfunction backwards to the time of insertion, counting the time in usage. Usage is not solely in production, but should coincide with the actual time the tool is operating on the material. Thus, an alternative approach could be to have a reliable counter, avoiding sudden extreme values or resets. The success of a supervised

model is determined by the quality of data mentioned above, but also by the amount of it. To get a sufficient sample size, it might not be sufficient to only extract data relevant to malfunctions, especially if a RUL prediction is intended to be performed on tool-level. As done in previous studies, collecting data from isolated runs until breakage is a possible strategy to increase the sample size.

5.2 Anomaly Detection Model

Compared to other models, such as simple thresholding, MTAD-GAT, ARIMA, the autoencoder showed superior performance on the anomaly detection dataset D1. This observation holds both when using the AUC score for the PRC as a measure, and when considering the practical use-case scenario. Avoiding FPs is a crucial aspect in order to not raise false alarms in a deployment scenario, possibly causing an unnecessary production shutdown. Autoencoder_{10} performed better in terms of FPs, outperforming the other models when lowering the threshold to detect all anomaly sequences. Setting the thresholds based on errors in the training set still results in more than 70% detected anomalies, from both groups. This suggests that the model is capable of learning the normal behavior of the data. The low number of FPs from both thresholds implies that the model can effectively identify both substantial deviating single values as well as a series of smaller deviating values during many of the anomaly sequences.

Results from the baseline model imply that to detect anomalies of the same type as represented in group 1, it is sufficient to set a maximum value corresponding to the highest value seen in the training set. While this method results in a few FPs, it is an easy and effective approach to detect some malfunctions earlier. The baseline results also demonstrate that a large value is not sufficient to detect anomaly sequences from group 2. The ROC and PRC also show that the baseline is strongly limited in separating positive from negative instances.

The ARIMA model surpasses the baseline model but falls short in comparison with the deep learning models. This could be due to the fact, as stated in the literature, that reconstruction models are better when analyzing unpredictable time series. This is also supported by the fact that MTAD-GAT, which combines reconstruction and forecasting, also is outperformed by the autoencoder. Comparing the loss for reconstruction and forecasting verifies that forecasting is more difficult than reconstructing the data. This might imply that forecasting modules have a negative impact on the classification, and reconstruction-based models handle the unpredictable time series in a better way. However, it is possible that another version of MTAD-GAT could exceed the performance of the autoencoder. Since the number of parameters in MTAD-GAT is roughly 5 times larger, the training sets might be too small to enable its full potential. The more complex the model is, the larger dataset is required. Otherwise the risk for overfitting is increased, where the model learns specific patterns instead of general ones, reducing its capability to deal with new, unseen data. Current possible improvements for MTAD-GAT could be to tweak the weights of reconstruction and forecasting parts in the error computation, giving more importance to reconstruction. Another possibility for improvement is to

dynamically adjust the weight assigned to each feature, considering the variability and noise levels. As seen for the dataset D1, solely the vibration in Y-axis proved equally good results as for the combined vibrations. We leave a more extensive search for the optimal settings for future work.

Evaluating the model from a use-case perspective is important in order to investigate its performance in a future deployment scenario. From such an inspection many FPs related to anomaly sequences are found. Since the current median value is affected by previously large errors in the selected sequence, the motivation for not counting FPs occurring right after a true malfunction when using MAD is valid. Considering that a new tool can give rise to large vibrations explains such findings when using the $\max_{global}$ threshold. In the case of deployment, avoiding such false alarms can be achieved by ignoring any alarms risen related to a new tool insertion. This demonstrates the importance of studying the occurrence of FPs. A future implementation of a new tool signal might reveal that the remaining FPs are related to such events. It should also be kept in mind that erroneous data is found and removed from the training set, while similar data might appear in the testing data, possibly being the cause for FPs.

Studying the FNs displayed in Fig. 4.7 indicates that some of the undetected sequences incorporate some characteristics that might be detectable. The two first sequences are constituted of multiple relatively high errors, indicating that a threshold based on the area under the curve might distinguish these as anomalies.

When testing the autoencoder on more tools, the performance is varying. This shows how each dataset carries individual properties and states that one model fits all is a naive assumption. Further investigations for each dataset could provide more insights helpful for creating an accurate model. Anomaly sequences in group 1 can still be detected when trained on another data, but more complex anomalies are not detected. Finding a model that generalizes well is a huge challenge.

5.3 Improvements from a Model Perspective

Detecting anomalies with the model gives on average a few seconds heads-up, compared with the current workflow. Using MAD generally gives a lower heads-up, intuitively since it requires multiple large errors to report an anomaly. Thus, a trade-off between heads-up and accuracy may be relevant. The impact the heads-up provides is determined by two things: what is the reason behind the signals detected and the action taken on a risen alarm.

If the reason for the alarm is the true event of malfunction it requires shorter reaction time to avoid material damage, while if it represents events leading up to a malfunction more time is available to prevent the actual damage. To determine this, more fine grained annotation of when the actual breakage occurs is needed. This probably has to be manufactured in a test environment.

The possible actions to take are of different severity levels, from warning messages to machine emergency stops. Due to the available response time presented by the heads-

up, it is probably only the emergency stop-action that will result in a prevention of possible tool breakage. However, a shutdown as responding action needs to be determined in the light of costs for pausing production for tool inspection versus damaged products. A shutdown due to a false alarm correlates with fewer produced articles and repeated shutdowns might also affect the reliability and sentiment of the operators. On the other hand, missing a tool breakage will result in loss of revenue due to discarded products in combination with operators manually having to verify the condition of recently produced products.

Ultimately, it needs to be determined what the detected anomaly represents. Detecting 31 malfunctions in combination with 60 false alarms indicates that savings for early malfunction detection should double the costs of false alarms, in combination with being endurable by the operators, in order to be acted on.

5.4 Conclusion

To conclude the findings of this thesis, anomaly detection was found to be the feasible approach for reaching a data driven decision-making scenario based on the current characteristics of the data stored at Sandvik Coromant AB. For the selected dataset, the utilization of a reconstruction-based LSTM autoencoder has demonstrated superior performance compared to other statistical and deep learning models. Specifically, its reconstruction capabilities drastically reduce the FPs compared to other models while still detecting all anomaly sequences. Evaluating the model from a use-case perspective and avoiding any FPs in the training data resulted in an F_1 score of 0.75 finding 12 of 17 malfunctions for the selected dataset. However, each dataset has its own characteristics. This means that extending the evaluation to multiple tools exposed the difficulties inherent in time-series analysis. In total, the model successfully identifies 31 out of 74 malfunctions. Using an anomaly detection model enables the detection of malfunctions prior to the current workflow. The exact enhancements need to be evaluated from a cost perspective comparing the savings of an avoided equipment failure to a shutdown caused by a false alarm.

5.4.1 Future Works

To make more informed decisions regarding malfunction detection, it is essential to consider the concept of RUL. Investigating and integrating RUL methods into the malfunction prediction framework would allow for continuous monitoring of when a system or component is likely to fail. Therefore, it is of high interest to adapt the state of data collection to incorporate relevant RUL information, by more fine grained annotation.

Although our study involved tuning the complexity models to the best of our abilities, further model tuning is of interest. Fine-tuning the hyperparameters and exploring alternative model architectures could potentially enhance the performance of the selected models or reveal the superiority of other models that were not included in our comparison. In order to further refine the model and optimize its performance, an iterative process of experimentation and evaluation should be undertaken.

In order to achieve higher accuracy and reduce FPs, it is important to conduct further analysis and leverage domain knowledge. By investigating specific cases and studying the characteristics of FPs, we can identify potential limitations or biases in the current models. This analysis could involve collaborating with domain experts to gain insights into the underlying causes of FPs and guide the development of more accurate anomaly detection approaches. To increase the understanding of the model performance, future studies should investigate how it operates on unexplored tools and machines.

6

Takeaways

In this chapter, we summarize the key insights and lessons learned from our research and project.

This project has been an exciting and educational journey, encompassing all aspects of a project pipeline, facing challenges within each one of them. From collecting and understanding data from a production environment to the potential end user's decision-making. It has been a bumpy ride, where each challenge provided some useful insight.

One thing we were constantly reminded of is the importance of planning and keeping a structured work process. Both regarding the big picture in terms of knowing the data characteristics to set the guidelines for the project before going down any path that needs to be revised later, as much as the everyday work where it is easy to dig deep immediately instead of widening the approach. In addition, we have learned the importance of planning the work, both high and low level. Discovered the hard way, these plans do rarely unfold exactly as anticipated. Adaptability and flexibility have proven to be crucial in navigating unexpected challenges or changes, allowing us to adjust our approach and still achieve our and the company's objectives.

For the theoretical aspect of the project, we have gained further understanding in working with time-series data and how to approach an anomaly detection scenario by reconstructing or forecasting time-series. We have studied multiple models and extended our knowledge of key building blocks and first-handed experienced the false assumption that higher complexity yields better results. From this demanding but more so exciting project, we can't help but agree to the *no solution fits all*-statement within the field of PdM. Finally, being Sandvik Coromant's and the team of COPDs first master thesis, we take pride in paving the way for future projects and setting a strong foundation for continued innovation and success within this area.

Bibliography

[1] M. Haarman, M. Mulders, and C. Vassiliadis, "Predictive maintenance 4.0: Predict the unpredictable," in *PwC documents, no. PWC & mainnovation (p. 31)*, vol. PwC and Mainnovation, 2017.

[2] S. Ayvaz and K. Alpay, "Predictive maintenance system for production lines in manufacturing: A machine learning approach using IoT data in real-time," *Expert Systems with Applications*, vol. 173, p. 114 598, Jul. 2021, ISSN: 09574174. DOI: 10.1016/j.eswa.2021.114598. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417421000397 (visited on 02/15/2023).

[3] C. A. Escobar, M. E. McGovern, and R. Morales-Menendez, "Quality 4.0: A review of big data challenges in manufacturing," *Journal of Intelligent Manufacturing*, vol. 32, no. 8, pp. 2319–2334, Dec. 2021, ISSN: 0956-5515, 1572-8145. DOI: 10.1007/s10845-021-01765-4. [Online]. Available: https://link.springer.com/10.1007/s10845-021-01765-4 (visited on 02/02/2023).

[4] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106 024, Nov. 2019, ISSN: 03608352. DOI: 10.1016/j.cie.2019.106024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360835219304838 (visited on 02/02/2023).

[5] T. Zonta, C. A. da Costa, R. da Rosa Righi, M. J. de Lima, E. S. da Trindade, and G. P. Li, "Predictive maintenance in the industry 4.0: A systematic literature review," *Computers & Industrial Engineering*, vol. 150, p. 106 889, Dec. 2020, ISSN: 03608352. DOI: 10.1016/j.cie.2020.106889. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360835220305787 (visited on 02/03/2023).

[6] W. J. Lee, H. Wu, H. Yun, H. Kim, M. B. Jun, and J. W. Sutherland, "Predictive maintenance of machine tool systems using artificial intelligence techniques applied to machine condition data," *Procedia CIRP*, vol. 80, pp. 506–511, 2019, ISSN: 22128271. DOI: 10.1016/j.procir.2018.12.019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2212827118312988 (visited on 02/21/2023).

[7] L. Ren, Y. Sun, J. Cui, and L. Zhang, "Bearing remaining useful life prediction based on deep autoencoder and deep neural networks," *Journal of Manufacturing Systems*, vol. 48, pp. 71–77, Jul. 2018, ISSN: 02786125. DOI: 10.1016/j.jmsy.2018.04.008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0278612518300475 (visited on 02/08/2023).

[8] A. Kanawaday and A. Sane, "Machine learning for predictive maintenance of industrial machines using IoT sensor data," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China: IEEE, Nov. 2017, pp. 87–90, ISBN: 978-1-5386-0497-7. DOI: 10.1109/ICSESS.2017.8342870. [Online]. Available: http://ieeexplore.ieee.org/document/8342870/ (visited on 02/10/2023).

[9] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 19, 2018, pp. 387–395. DOI: 10.1145/3219819.3219845. arXiv: 1802.04431[cs,stat]. [Online]. Available: http://arxiv.org/abs/1802.04431 (visited on 04/03/2023).

[10] R.-J. Hsieh, J. Chou, and C.-H. Ho, "Unsupervised online anomaly detection on multivariate sensing time series data for smart manufacturing," in *2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)*, Kaohsiung, Taiwan: IEEE, Nov. 2019, pp. 90–97, ISBN: 978-1-72815-411-4. DOI: 10.1109/SOCA.2019.00021. [Online]. Available: https://ieeexplore.ieee.org/document/8953015/ (visited on 05/17/2023).

[11] S. Liu, B. Zhou, Q. Ding, *et al.*, "Time series anomaly detection with adversarial reconstruction networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 4293–4306, Apr. 1, 2023, ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2021.3140058. [Online]. Available: https://ieeexplore.ieee.org/document/9669010/ (visited on 05/17/2023).

[12] D. Li, D. Chen, L. Shi, B. Jin, J. Goh, and S.-K. Ng, *MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks*, Jan. 15, 2019. arXiv: 1901.04997[cs,stat]. [Online]. Available: http://arxiv.org/abs/1901.04997 (visited on 05/17/2023).

[13] L. Wong, D. Liu, L. Berti-Equille, S. Alnegheimish, and K. Veeramachaneni, "AER: Auto-encoder with regression for time series anomaly detection," in *2022 IEEE International Conference on Big Data (Big Data)*, Osaka, Japan: IEEE, Dec. 17, 2022, pp. 1152–1161, ISBN: 978-1-66548-045-1. DOI: 10.1109/BigData55660.2022.10020857. [Online]. Available: https://ieeexplore.ieee.org/document/10020857/ (visited on 04/03/2023).

[14] H. Zhao, Y. Wang, J. Duan, *et al.*, *Multivariate time-series anomaly detection via graph attention network*, Sep. 4, 2020. arXiv: 2009.02040[cs,stat]. [Online]. Available: http://arxiv.org/abs/2009.02040 (visited on 04/03/2023).

[15] S. Alnegheimish, D. Liu, C. Sala, L. Berti-Equille, and K. Veeramachaneni, "Sintel: A machine learning framework to extract insights from signals," in *Proceedings of the 2022 International Conference on Management of Data*, Jun. 10, 2022, pp. 1855–1865. DOI: 10.1145/3514221.3517910. arXiv: 2204.09108[cs]. [Online]. Available: http://arxiv.org/abs/2204.09108 (visited on 05/17/2023).

[16] S. Ho, M. Xie, and T. Goh, "A comparative study of neural network and box-jenkins ARIMA modeling in time series prediction," *Computers & Industrial Engineering*, vol. 42, no. 2, pp. 371–375, Apr. 2002, ISSN: 03608352. DOI: 10.

1016/S0360-8352(02)00036-0. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360835202000360 (visited on 05/17/2023).

[17] S. Makridakis, E. Spiliotis, V. Assimakopoulos, A.-A. Semenoglou, G. Mulder, and K. Nikolopoulos, "Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward," *Journal of the Operational Research Society*, vol. 74, no. 3, pp. 840–859, Mar. 4, 2023, ISSN: 0160-5682, 1476-9360. DOI: 10.1080/01605682.2022.2118629. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/01605682.2022.2118629 (visited on 04/18/2023).

[18] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, *LSTM-based encoder-decoder for multi-sensor anomaly detection*, Jul. 11, 2016. arXiv: 1607.00148[cs,stat]. [Online]. Available: http://arxiv.org/abs/1607.00148 (visited on 04/03/2023).

[19] A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, and K. Veeramachaneni, *TadGAN: Time series anomaly detection using generative adversarial networks*, Nov. 14, 2020. arXiv: 2009.07769[cs,stat]. [Online]. Available: http://arxiv.org/abs/2009.07769 (visited on 05/17/2023).

[20] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage AK USA: ACM, Jul. 25, 2019, pp. 2828–2837, ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330672. [Online]. Available: https://dl.acm.org/doi/10.1145/3292500.3330672 (visited on 05/23/2023).

[21] D. E. Rumelhart, G. E. Hintont, and R. J. Williams, "Learning representations by back-propagating errors," 1986.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1, 1997, ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://direct.mit.edu/neco/article/9/8/1735-1780/6109 (visited on 03/21/2023).

[23] K. Cho, B. van Merrienboer, C. Gulcehre, *et al.*, "Learning phrase representations using RNN encoderdecoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. [Online]. Available: http://aclweb.org/anthology/D14-1179 (visited on 04/25/2023).

[24] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, *Graph attention networks*, Feb. 4, 2018. arXiv: 1710.10903[cs,stat]. [Online]. Available: http://arxiv.org/abs/1710.10903 (visited on 04/25/2023).

[25] S. Brody, U. Alon, and E. Yahav, *How attentive are graph attention networks?* Jan. 31, 2022. arXiv: 2105.14491[cs]. [Online]. Available: http://arxiv.org/abs/2105.14491 (visited on 04/12/2023).

[26] B. Xu, N. Wang, T. Chen, and M. Li, *Empirical evaluation of rectified activations in convolutional network*, Nov. 27, 2015. arXiv: 1505.00853[cs,

stat]. [Online]. Available: http://arxiv.org/abs/1505.00853 (visited on 04/25/2023).

[27] D. Bank, N. Koenigstein, and R. Giryes, *Autoencoders*, Apr. 3, 2021. arXiv: 2003.05991[cs,stat]. [Online]. Available: http://arxiv.org/abs/2003.05991 (visited on 06/06/2023).

[28] G. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. Holden-Day, 1976.

[29] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano, "A review on outlier/anomaly detection in time series data," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–33, Apr. 30, 2022, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3444690. [Online]. Available: https://dl.acm.org/doi/10.1145/3444690 (visited on 02/01/2023).

[30] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2886457. [Online]. Available: https://ieeexplore.ieee.org/document/8581424/ (visited on 02/02/2023).

[31] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/1541880.1541882. [Online]. Available: https://dl.acm.org/doi/10.1145/1541880.1541882 (visited on 04/25/2023).

A

Appendix - Features

Table A.1: Extracted signals: name and description.

Signal name	Description [unit]
TimeStamp	Date and time for data entry
MDC_AlarmID1	Machine alarm [-]
MDC_AlarmID2	Machine alarm [-]
MDC_AlarmID3	Machine alarm [-]
MDC_AlarmID4	Machine alarm [-]
MDC_AlarmID5	Machine alarm [-]
MDC_MaterialID	Material ID [-]
MDC_OperationID	Operation ID [-]
MDC_ProductionSignal	Production signal [-]
MDC_ToolNbr1	Current tool in spindle [-]
MDC_WorkOrderNumber	Article number for product [-]
Object_1_value	Vibration in x-joint [milligravity]
Object_2_value	Vibration in y-joint [milligravity]
Object_3_value	Vibration in z-joint [milligravity]
Object_4_value	Rotational speed of spindle [rpm]
Object_8_value	Vibration in x-joint [milligravity]
PDA_Actual_Feed_Rate	Machine throttle [-]
PDA_Actual_Spindel_RPM	Rotational speed of spindle [rpm]
PDA_Tool_Life_Counter	Current working tool life counter [s]
PDA_Maximum_Tool_Life	Current working tool theoretical life length [s]
PDA_Remaining_Tool_Life	Maximum tool life - Tool life counter [s]

B

Appendix - Model Parameters

Parameter	**Value**	**Description**
Learning Rate	0.001	Step size for each update
Batch Size	256	Samples per batch during training
Epochs	100	The number of times to iterate over the entire dataset
Optimizer	Adam	The optimization algorithm used
Dropout	0.3	Rate of neurons dropped out at each update
Convolutional Parameter	**Value**	**Description**
Kernel Size	7	Size of the filter
GAT Parameter	**Value**	**Description**
Feature Embedded Dimension	20	Size of feature vector representing each node
Temporal Embedded Dimension	20	Size of time vector representing each node
GRU Parameter	**Value**	**Description**
Number of Layers	1	Number of GRU layers
Hidden dimension	150	Number of neurons
Forecasting Parameter	**Value**	**Description**
Number of Layers	3	Number of fully connected layers
Hidden dimension	150	Number of neurons
Reconstruction Parameter	**Value**	**Description**
Number of Layers	1	Number of fully connected layers
Hidden dimension	150	Number of neurons

Table B.1: MTAD-GAT Hyperparameters

C

Appendix - Model Implementation

A sketch of an integration to the existing system can be seen in Fig. C.1. The data historian database, which stores the signals obtained from machines and sensors, forwards the subset of signals needed (vibrational signals, spindle RPM, material ID and tool ID) to the model. If needed, an intermediate layer can be used to extract the signals, from which the model query the data. Since the model operates on tool level, the tool id is required. The whole set of the needed signals should be sent in the same time stamp, so that they are corresponding (related). The predictions are computed for the next time step while reconstruction is performed on the current data. Hence, for each updated signal, the model compares the values with the previously predicted ones and with ones currently reconstructed.

The model should output an indication if the input is an anomaly or not, represented by a 1 or 0, respectively. The time from which input signals are sent until output, including all computational steps within the model, should not exceed the delta between two input signals. As of current configuration, that delta is 1 second. Thus, demands are high on selecting efficient methods for data extraction and migration. To achieve high computational speed, one option is to install an NVIDIA graphic processor unit (GPU), enabling parallel computations. The faster these steps are performed, the more efficient predictions can be made, enabling a smaller time delta between data extraction and possible actions.

Depending on the reliability to the model, different actions are possible. As a first step, the labels can be stored and an indication can be sent to notify any person monitoring the output. A further step is to let the detected anomalies be alerted to operators via their screen monitoring the machines. At a level where the output of the model is completely trusted, a detected anomaly renders in a machine emergency stop.

Increasing the interpretability of the model classifications and behaviour can be achieved by also storing the predicted and reconstructed values. This would make visualization and other comparable analysis available, and possibly give insights to further improvements or threshold tuning.

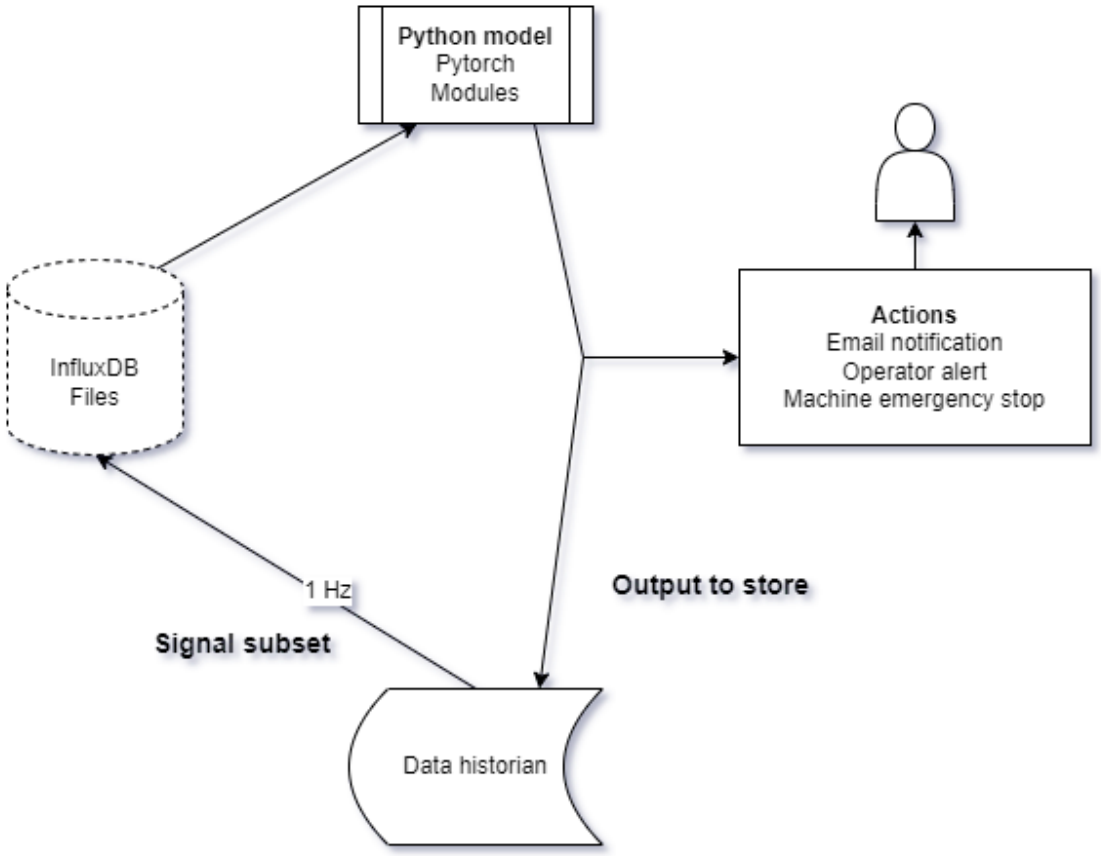


Figure C.1: High level flowchart of model implementation.