# ANOMALY DETECTION FOR PUMP RELATED TIME SERIES DATA

## A GUIDE FOR ANOMALY DETECTION IN DATA GENERATED FROM SENSORS PRESENT IN PUMP

CHAHAT GUPTA

STUDENT ID 202101681

MASTER'S THESIS

September 2023

Supervisor: Panagiotis Karras
Co-Supervisor: Henrik Pederson
Company Supervisors: Dávid Nagy, Rasmus Engholm

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

# ANOMALY DETECTION FOR PUMP RELATED TIME SERIES DATA

*A Guide for Anomaly detection in data generated from sensors present in Pump*




Master's Thesis

Department of Computer Science
Faculty of Natural Sciences
Aarhus University

September 2023

*Impermanence* is the law of universe.

Whatever has happened, has happened for good
Whatever is happening is also for good
Whatever will happen, shall also be good

— The Bhagavad Gita

# ABSTRACT

In this thesis I explore multiple Distance based, Deep learning as well as Machine Learning techniques for anomaly detection in the time series data recorded from the pump sensors manufactured by the world leading pump maker. The data, obtained was used to train various models to predict and classify if the upcoming pressure time sequence data in anomalous or not. In a window if the sequence of the points prediction score exceed the threshold they were grouped and classified as anomalous. I explore multiple models which includes FastDTW, Autoencoders, Long Term Short Memory Networks $(LSTM)$, Temporal Convolution Network $(TCN)$, Long Term Short Memory AutoEncoder $(LSTM-AE)$, and Temporal Convolution Network $(TCN-AE)$ AutoEncoder.

I reviewed multiple papers which have implemented these state-of-the-art models and was able to achieve $F_1$ score of 0.99 on the considered dataset.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES



# ACRONYMS

IPC Intelligent Pump Control

PLC Programming Logic Controller

# 1

# INTRODUCTION

Anomaly detection in the context of time data, often referred to as time series anomaly detection, involves identifying unexpected or unusual patterns within sequential data points collected over time. These patterns, or anomalies, can deviate from expected behaviors based on historical trends, seasonality, or cyclical patterns. Such deviations might indicate system malfunctions, security breaches, or other significant events. In terms of time series data, input data could be either univariate, which is the focus of this work, or multivariate type. There has been extensive research and active research ongoing on detecting the anomalies to ensure the protection of the Critical Infrastructure (CI) involved in them.

Time series data, such as stock prices, web traffic, sensor readings, or sales numbers, inherently contain temporal dependencies, meaning the value at a given time depends, in part, on previous values. Anomaly detection techniques, like moving averages, autoregressive models, and more deep learning methods like Long Short-Term Memory (LSTM) networks, are designed to model these dependencies. Once a model is trained on 'normal' data, it can then be used to flag any data points or sequences that deviate significantly from the expected patterns as potential anomalies. The identification of these irregularities can be critical for proactive system maintenance, fraud detection, and other applications where real-time insights can lead to timely interventions.

Pumps are critical infrastructures, that need to operate consistently to ensure smooth processes, optimal efficiency, and extended machine's lifespan. Time series data from these pumps can include metrics like pressure, flow rate, temperature, vibration levels, and energy consumption. Pumps are critical to a wide array of settings like industries ranging from manufacturing to cooling machinery, water treatment plants, hydrological systems, and households. Changes in the metrics produced by the pump might indicate a pump malfunction, degradation, or other operational issues which could lead to a shutdown of the operations in industries and high maintenance costs. Detecting anomalies in time series data produced by the pump.

Analyzing this data over time enables the identification of patterns and trends indicative of the pump's normal operating conditions. By

applying the computational algorithms, machine learning, or deep learning approaches, the aim is to develop a model that learns the underlying features and seasonalities in generated sequence data over time. The results produced by these models can help industries preemptively identify and address potential pump issues before they escalate into major breakdowns, ensuring continuous operation, minimizing downtime, and reducing maintenance costs. For instance, a sudden spike in vibration levels could suggest mechanical wear or misalignment, while an unexpected drop in flow rate could be indicative of blockages or pump cavitation.

## 1.1 ANOMALY DETECTION FOR PUMPS

Pumps are often integral to the operation of a system or process and anomaly detection in pumps by analyzing the time series data can help with:

- *Overall System Stability*, For e.g., in a water treatment plant, pumps are used in various stages to move water through the system. If a pump fails, it can disrupt the entire process, leading to operational downtime and potential financial loss. Anomaly detection helps monitor pump health proactively and continuous running of operation
- To reduce *Costs*, pumps are often categorized as high-cost assets. Unexpected behavior or pump failure can lead to expensive repair or replacement. Additionally, the downtime associated with pump failure can result in significant operational costs.
- For *Risk Reduction*, in industries like oil and gas or chemical processing, pump failure could result in hazardous situations such as leaks or spills that pose a risk to human safety and the environment.
- To *Minimize downtime*, some pumps have sensors built into them for leak detection, and failure and can activate a pump shutdown to save the impeller or propeller from damage. This minimizes downtime as we are only required to replace the damaged parts or take corrective measures.

Overall, anomaly detection in Pumps or abnormal pump behavior pointing to failures can address any hazardous situations before they emerge. Personnel safety and environmental protection are ensured. Figure 1 references various other benefits of anomaly detection in pumps by applying various techniques.

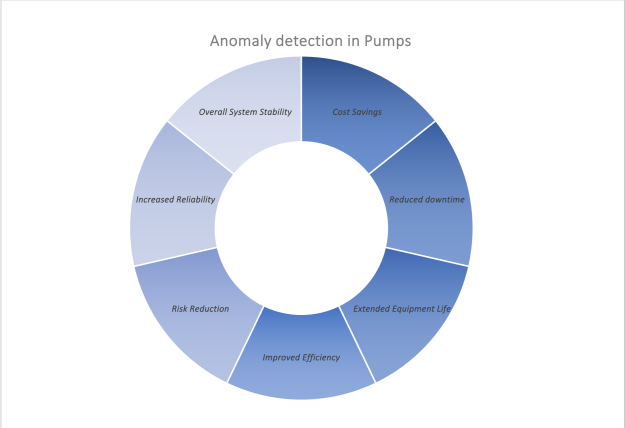



Figure 1: Benefits of Anomaly Detection for Pumps.

## 1.2 PREDICTIVE MAINTENANCE FOR PUMPS FOR ANOMALY DETECTION

Predictive Maintenance entails monitoring of the key equipment properties, which in this case are data from pump sensors to predict equipment faults before they occur based on patterns in operational data. Predictive maintenance using Machine learning or Deep learning approaches can lead to cost savings, extended equipment lifespan, and improved efficiency which is also one of the crucial aspects in any manufacturing industry. The goal is to identify early signs of potential issues or anomalies in the data before failures happen. Detecting anomalies early through predictive maintenance allows maintenance to be scheduled proactively before a failure occurs which in turn brings down the Total Cost. In Figure 2, it can be visualized how Predictive Maintenance tries to find a balance i.e., *Optimum* in regards to the Number of Failures and Cost over time [12, 19]. Predictive maintenance relies on condition monitoring through the measurement and analysis of physical characteristics such as sound, visual inspection results, temperature, and vibration. This is useful as it helps avoid unexpected equipment breakdowns and the associated costs of production downtime. Unexpected failures disrupt operations. By predicting issues in advance, maintenance can be carried out at a convenient time during regular maintenance schedules instead of urgently in response to breakdowns [8].

For Predictive Maintenance, computational, machine learning and deep learning algorithms can be applied to time-series sensor data

collected from industrial machinery to learn these patterns. Features are extracted from raw sensor readings that are indicative of equipment health. Sensor readings like vibration, temperature, noise, pressure etc. Labeled datasets contain these features along with the classification of data points as normal or faulty. The models learn the normal operating patterns and boundaries of the feature space. New unlabeled operational data is then fed to these trained models. The models detect if new data points lie within the normal boundaries learned during training or indicate anomalies. These normal boundaries are considered to set the *threshold*. As different models use different ways to learn, the threshold value is set accordingly. The right threshold keeps a balance between the number of False Positives/-False Negatives reported by the model. Any anomalies predicted by the models before an actual fault occurs enable predictive maintenance by scheduling works in advance. This avoids expensive unplanned breakdowns and improves equipment availability through preventative monitoring using machine learning algorithms applied to operational sensor data [21].

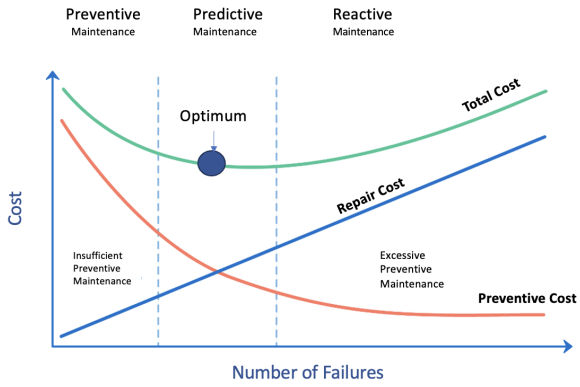


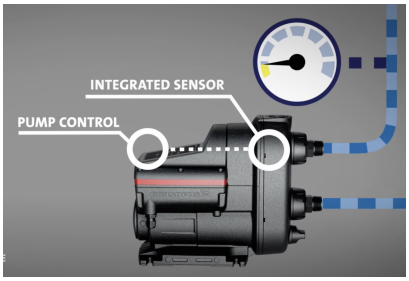

Figure 2: Predictive Maintenance for Pumps. The image is an adaptation from [12, 19]

## 1.3 ANOMALY DETECTION FOR SCALA2 PUMP

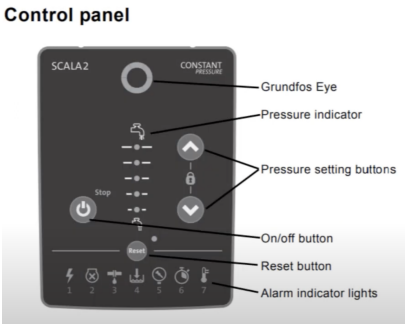
*SCALA2 Pump* by *Grundfos* has been a game changer in domestic households for boosting water pressure. It's a water booster pump that can help maintain the set pressure in all taps at all times irrespective of varying water pressure emerging from varying inlet pressure

from the city mains/reservoirs or multiple open taps. Varying inlet water pressure from the entire city systems affects homeowners during the day, in the mornings and evenings the pressure is low since the water usage is more because of showering, cleaning, and, cooking and when they're away during the day the water pressure increases. This variation can be felt in the taps around the house whenever you turn on a tap.

(a) Grundfos SCALA2 Pump




(b) Pressure Sensor along with pressure gauge




(c) Control Panel to operate Pump


(d) Digital Panel to adjust pressure

Figure 3: Water Boosting Pump used to collect data.

The SCALA2 system is equipped with a sensor that gauges water pressure from the pump to outlets like showers. If the detected pressure diverges from the user-set desired level, the system auto-adjusts the pump speed via its frequency converter, ensuring consistent pressure. This functionality of maintaining the outlet water pressure by the pump comes through the internal pressure tank which is called *precharge pressure* 3. Then tank precharge has to be 70% of the required

outlet pressure. This setup ensures optimal pressure across all taps, regardless of inlet pressure or concurrent tap usage. The pump also consists of *Intelligent Pump Control* IPC which monitors the pressure and flow of the water in the system and accordingly runs the pump or puts it into an ideal mode where there is no need to raise or lower the pressure. In cases, where the pump keeps on operating for more than 2 minutes, the IPC will try to put the pump in ideal mode to prolong the pump's life and lessen the energy consumption. For the experiments, the pressure was set to *3.5 bar*. The tank precharge pressure was maintained at 70% of the set 3.5 bars which was *2.45 bars*.

For more details on how the data has been generated and used, users can refer to *Chapter 4* for the details.

## 1.4 MOTIVATION FOR ANOMALY DETECTION

The motivation for anomaly detection in pumps is driven by a multifaceted pursuit of efficiency, safety, and cost-effectiveness in industries heavily reliant on fluid transport and machinery. Early anomaly detection is crucial for preventing catastrophic failures, minimizing downtime, and avoiding costly disruptions in sectors like manufacturing and energy production. It significantly reduces maintenance costs and optimizes resource allocation by enabling proactive, data-driven maintenance strategies.

Anomaly detection in a broader spectrum also helps in enhancing energy efficiency, contributing to sustainability goals, and reducing energy consumption. Moreover, this approach aligns with the digitalization trend, integrating advanced analytics and machine learning into pump monitoring for operational efficiency and innovation.
Anomaly detection in household pumps involves the application of data-driven techniques to monitor and identify deviations from normal operating conditions in residential water and heating systems. This typically requires the installation of sensors to collect relevant data, such as flow rates, pressure, and temperature. Machine learning algorithms, including statistical methods and neural networks, can then be employed to analyze this data in real-time. When anomalies are detected, such as a sudden drop in water pressure or irregular pump behavior, the system can trigger alerts or automatic corrective actions to prevent potential issues, ensuring the smooth and efficient operation of household pumps and enhancing overall convenience and safety for homeowners.

In a typical residential water system, water pressure can vary within certain expected ranges. E.g., during normal operation, the water

pressure might be around 2.75 to 4.0 bars. However, if there's a sudden and significant drop in water pressure to, let's say, 0.5 bar, it could indicate a leak in the plumbing system, a malfunctioning pump, or other issues. An anomaly detection system for residential plumbing could continuously monitor water pressure data. When it identifies sudden or unexpected changes in the values it can alert homeowners to take preventive measures before the damage is irreparable. This early detection can help prevent water damage, conserve water resources, and reduce repair costs by addressing issues promptly. Like with any other anomaly detection system, the goal is to strike a balance, avoiding excessive false alarms while reliably identifying anomalies in water pressure to maintain the integrity of the household plumbing system.
In essence, anomaly detection in pumps represents a strategic investment for improved reliability, cost reduction, energy efficiency, safety, and alignment with modern industrial practices, positioning industries for smarter, more resilient, and sustainable operations.

## 1.5 PROBLEM STATEMENT

This thesis is centered on the development of an anomaly detection system tailored for pump sensor data, utilizing a diverse range of models hailing from distinct domains, encompassing computational algorithms, machine learning techniques, and deep learning paradigms.

Our underlying hypothesis posits, that the deep learning Model *Temporal Convolution Networks (TCNs)* which are a variant of Convolution Neural Networks (CNN) will exhibit notably superior accuracy in detecting anomalies than other baseline models such as FastDTW (Fast Dynamic Time Warping), Autoencoders, Multilayer perceptron (MLP) and LSTM models. This assertion is founded on TCNs' intrinsic design for processing sequential data, as well as their capacity to retain information over extended temporal horizons. More information on the structure and evaluation of these models can be found in *Chapter 5*.

The experimental dataset has been procured through collaboration with *Grundfos*, a preeminent global pump manufacturer, originating from their Leakage Detection System. This dataset comprises univariate time series data, meticulously logging pressure values derived from within the system. In the course of our research, we employ cutting-edge models for anomaly detection, adopting an unsupervised methodology in analyzing the recorded data. Our approach

incorporates the application of the Window Sliding technique, as elucidated in the works referenced in [7, 13], to preprocess the data for training. Additionally, we employ diverse threshold techniques contingent on the model's outputs to identify and flag anomalies within the dataset.

## 1.6 KEY FINDINGS & OUTLINE

Some of the findings from the experiment are summarised below:

- Tested out 1 distance-based, 3 predictions, and 3 reconstruction-based architectures (FastDTW, MLP, TCN, LSTM, AutoEncoder, LSTM AutoEncoder, and TCN AutoEncoder).
- We found using the fixed window size which matches the periodic signal in the data yielded better results in comparison to large/small window-size.
- It took more time to train the TCN and TCN-AE models due to their architecture involving dilation and skip connections.
- The LSTM got the lowest accuracy of 33%, while all other architecture achieved an accuracy of more than 85% accuracy.

This thesis is organized as follows.

- **Chapter 2** is a review of various anomaly detection techniques that have been applied to time series data.
  We discuss the state-of-the-art models in finding anomalies and categorize the techniques depending upon the method used for detecting anomalies and also, what transformation techniques they use before anomaly detection.
- **Chapter 3** discusses the approaches for anomaly detection and the motivation and methodology for the Window Size selection and Rolling window-based transformation applied to the data.
- **Chapter 4** discusses the overview of the system, data acquisition, and data preparation for the experiments.
- **Chapter 5** discusses the architecture of the baseline models and TCN model used in the experiments and the evaluation of these models.
- **Chapter 6** discusses the results and future direction of research.

# 2

# RELATED WORK

In this chapter, we will discuss the extensive research that has been done on Anomaly detection and the state-of-the-art models that have been implemented to detect the anomalies. We also talk about the inspiration that has been drawn from the related work for choosing the various models.

Anomaly detection in time series data can be achieved using various techniques, including statistical methods and machine learning algorithms. Traditional statistical methods, such as moving averages (MA), time-series decomposition (FFT), and autoregressive models (ARIMA), have been widely used in anomaly detection for decades [9]. These methods are often based on the assumption that the time series data is stationary, meaning that the mean, variance, and autocorrelation structure of the data do not change over time [14]. Most of the time a lot of pre-processing is required to use these statistical models, including determining data. Moreover, these models also lose accuracy with large amounts of data. Anomaly detection on multivariate time series doesn't produce well-defined results.

However, machine learning algorithms, including deep learning methods, have gained popularity in recent years for anomaly detection in time series data. Machine learning techniques, such as neural networks (NN) and support vector machines (SVM), can be used to learn patterns and relationships in the sensor data and detect anomalies that deviate from the learned patterns [4, 20]. These methods can consider the temporal aspect of the data and can learn from the data without the need for explicit modeling of the underlying statistical distribution. Deep learning methods, such as Recurrent Neural Networks (RNNs) [18, 24], Convolutional Neural Networks (CNNs), a modification of RNNs that overcome the issue of memory loss, and Long Short Term Memory( LSTM) have also been widely implemented.

Other approaches include decision trees, random forests, and Autoencoder, Hierarchical temporal memory (HTM) [3]. TCN (Temporal Convolutional Network) offers a promising approach to anomaly detection in time series data since it can capture long-range dependencies in the data which is especially useful for detecting anomalies that occur over a long range and can be trained to detect anomalies in real-time [2]. Various hybrid models have also been proposed based on the best result-producing models [27]. LSTM along with Autoen-

coders and LSTM with Variational autoencoders (LSTM-VAE) are effective in detecting anomalies in time series data [1, 15, 26].

## 2.1 OVERVIEW OF EXISTING TECHNIQUES

The authors of the paper [23] mention the importance of Preventive maintenance in the circulating pumps to avoid breakdown and malfunctions. Sanayha et al. in their paper proposed a two-stage model that forecast for failures one day in advance. The authors have collected data for a Circulating water pump (CWP) for 8 sensors over a year. They have used the Auto Regression Integrated Moving Average (ARIMA) to forecast trends based on the historical data and further in the process, they classify the predicted value as Outliers. The model will forecast on the normalized data using the ARIMA and Regression Artificial Neural Network. This predicted data is de-normalized and classified by the model as an Outlier or not. They used the Auto correlation Function (ACF) and Passive Auto correlation Function (PACF) to determine the hyperparameters for the ARIMA model. The author used Mahalanobis distance to determine outliers for multivariate data. The data was collected hourly from the sensors in year 2015. They got F1 scores of 0.85 for the different events that occurred in the data over a year.

There have been multiple techniques applied for anomaly detection, these include distance based models like Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Isolation forest, and Dynamic Time Warping (DTW) [5]. Most of these models require a lot of training data for the models to train properly and produce reliable results but DTW has proved effective in case of periodic data and requires no training. Diab et al. introduced a novel approach to using DTW, which is a time series alignment algorithm for anomaly detection [6]. DTW is effective in capturing temporal dependencies and has been successfully applied to identify anomalies related to pump performance degradation. Since then there have been multiple improvements done to the DTW algorithm. In 2007, Salvador et al. introduced an improvement to the DTW algorithms which is referred to as FastDTW [22]. With the introduction of Coarsening, Projection, and Refinement in the original DTW, they were able to run the algorithm in Linear Time and Space. This improvement has made FastDTW one of the promising distance-based algorithms in anomaly detection.

Bai et al. proposed a novel approach for sequence modeling and the use of Convolution Neural Networks with dilation layers and skip connections [2]. Upon this building block, several research papers have been published using TCNs to model data and detect anoma-

lies.
He et al. proposed of using Convolution Networks, specifically Temporal Convolution Networks on different real-world datasets and modeling the prediction errors using a Gaussian distribution to detect anomalies. to extract high-level features from the structural data [25]. The paper presents a novel approach for anomaly detection in time series data using unsupervised learning, specifically applying the principles of Temporal Convolutional Networks (TCN) and autoencoders. The model learns to recognize normal behavior, assigning a low score to normal data and a high score to anomalous data. The model is compared with several other state-of-the-art anomaly detection algorithms, and it is shown to outperform them on a Mackey-Glass (MG) anomaly benchmark. After training, only a small amount of labeled data is needed to adjust the anomaly threshold. The paper also introduces a new synthetic benchmark based on the Mackey-Glass (MG) time series, called the Mackey-Glass Anomaly Benchmark (MGAB), which is a chaotic time series with injected anomalies. The authors demonstrate that their approach performs well on a new synthetic benchmark and is a promising method for real-world applications. TCN Models has been used for anomaly detection on other real-world datasets and have been effective in predicting the anomalies [11, 16, 28].

Gopali et al. in their paper did a comparative study of detecting Anomalies using Temporal Convolutional Networks (TCN) and Long Short-Term Memory (LSTM) models in time series data. They have explored - LSTM, a RNN that remembers past data, and TCN, a CNN that uses dilated convolutions to capture long-term trends. They used an algorithm to detect anomalies using these models by comparing predicted and actual values. The models are tested on multivariate sensor data from a water treatment testbed containing normal operations and attack incidents. The multivariate sensor comes from 3 sensors which are used as inputs and one as the output. The data is then pre-processed and split for training, validation, and testing. In the paper, they have build and trained multiple versions of LSTM and TCN models with 1, and 3 hidden layers. TCN uses dilated convolutions with 84 filters and 'tanh' activation. LSTM uses 'tanh' activation and dropout. MAE loss and Adam optimizer are used. The results obtained by them are, that TCN and 3-layer LSTM perform best with F1 scores of 0.920 and 0.901 respectively. TCN has a slightly higher recall (0.903 vs 0.864) and builds a stable model faster. 1-layer LSTM has the lowest training time but also the lowest F1 score. They conclude both these models are effective in detecting anomalies but TCN models train faster and perform marginally better than the LSTM models.

Wei et al. introduced an approach to detect anomalies in indoor air quality time using a long short-term memory (LSTM) autoencoder [26]. The proposed approach has the potential to be used for real-time anomaly detection in indoor environments. The authors start by collecting time series data of indoor air quality parameters, including CO2 and PM2.5 concentrations, from a real-world indoor environment, detecting anomalies is critical for ensuring the health and a safe environment. They pre-process the data by training their models on normal data by removing outliers and normalizing it to a range of 0 to 1. They split this data into training and testing sets with the training set consisting of normal indoor air quality data and the testing set consisting of both normal and anomalous data. In their approach, they propose an LSTM autoencoder to learn a compressed representation of the input data and then reconstruct the input from this representation. After training on the training set, they detect anomalies in the testing set. The anomaly score is calculated as the difference between the input and the reconstructed output, with a higher score indicating a higher likelihood of an anomaly. They evaluated their approach using several metrics, including precision, recall, F-score, and area under the receiver operating characteristic curve (AUC-ROC). Their approach effectively works to detect anomalies in the indoor air quality data, with an AUC-ROC score of 0.90 for CO2 and 0.87 for PM2.5. Their approach outperforms, approaches like K-Nearest Neighbor, including the moving average method. The limitation of their approach is it requires a large amount of training data and changes in the indoor environment could result in a large number of false positives that do not necessarily indicate an anomaly.

Malhotra et al. proposed an LSTM-AD for anomaly detection in Time Series data [17]. Since LSTM is a type of recurrent neural network capable of learning long-term dependencies in sequence data, it is well-suited for modeling time series where past values are correlated to current and future values. Traditional methods for time series anomaly detection like statistical tests often fail to capture these long-range dependencies. The proposed approach is evaluated on 3 different time series datasets, demonstrating it can discover anomalies while preserving the original temporal dependencies in the data. They achieved F1 scores of 0.84, 0.90 & 0.89 for the Space Shuttle, Power, and Engine Dataset respectively.

This paper proposes a hybrid deep learning model that combines Long Short-Term Memory (LSTM) and Autoencoder for anomaly detection in indoor air quality (IAQ) time series data [26]. The LSTM encoder learns the long-term dependencies in the time series input sequences as it reduces the high-dimensional input into a lower-dimensional encoded representation. The decoder contains another LSTM network

that reconstructs the input sequence from the encoded representation. During testing, sequences containing all $CO_2$ values are reconstructed and individual errors are compared to the threshold. Values with higher errors are labeled as anomalies. The proposed model achieves 99.5% accuracy, 100% precision, 89.9% recall, and 94.68% F1-score - outperforming other LSTM and autoencoder models. Sensitivity analysis shows the best performance with a time window length of 10 samples and single hidden layer architecture. Overall, the authors were able to show the combination of LSTM and autoencoder architectures leverages their strengths to learn long-term dependencies and reconstruct sequences for anomaly identification, and detect anomalies in real-world datasets.

The field of anomaly detection in pumps has seen significant advancements in recent years. Researchers have explored distance-based approaches, machine learning-based techniques, and deep learning-based models to address the unique challenges posed by pump systems. These studies provide valuable insights and methodologies for the development of effective anomaly detection solutions in both industrial and residential settings.

In the following chapters, we will build upon the insights from these related works to develop and evaluate various approaches for anomaly detection in pump systems.

# 3

# DETECTION TECHNIQUES AND WINDOW-BASED TECHNIQUES

In this chapter, we dive into the key concepts and techniques employed for detecting anomalies in pump time series data. We will also explore the use of Periodic Window-Based and Sliding Window techniques in this context. Time series data collected from sensors monitoring industrial pumps contains valuable information to detect anomalies and predict maintenance needs. Various machine learning models have been proposed for this task which can be broadly categorized into distance-based models, prediction models, and reconstruction models [5].

## 3.1 DISTANCE-BASED MODELS

Distance-based models are commonly used for anomaly detection in pump time series data. These models assess the dissimilarity between data points and employ distance measures to identify anomalies. Some widely used distance-based techniques include Euclidean distance, Mahalanobis distance, and dynamic time warping (DTW). The core idea is to measure the distance between data points and identify those that deviate significantly from the norm.
A commonly used distance-based algorithm is k-Nearest Neighbors (k-NN) which finds the k closest training samples to a test point based on a distance measure, usually Euclidean distance. If the average distance of the test point to its neighbors exceeds a pre-defined threshold, then it is predicted as an anomaly.

## 3.2 PREDICTION MODELS

Prediction-based models leverage historical pump time series data to forecast future observations. Anomalies are detected when actual observations deviate significantly from the predicted values. The prediction error or residual between actual and predicted values is analyzed. If the prediction error exceeds a certain threshold, the data point is flagged as an anomaly. They are capable of handling data with missing values or irregular sampling intervals. They can capture complex temporal dependencies better than the distance-based models.
Widely adopted prediction models include Auto-Regressive integrated moving average (ARIMA) models, recurrent neural networks (RNNs), and long short-term memory (LSTM) networks. These models might struggle with sudden shifts or drastic changes in the data distribution.

Complexity in choosing hyperparameters and handling multi-modal data to predict the next values from the previous ones.

## 3.3 RECONSTRUCTION MODELS

Reconstruction-based models involve training models to accurately reproduce input data. Anomalies are identified when the reconstruction error is high. Reconstruction models like Autoencoders often used in pump time series data, compress the high-dimensional input data into a lower-dimensional latent space representation before reconstructing the output.
They are generally capable of detecting a wider range of anomalies, including those that don't exhibit predictable patterns. Reconstruction-based models are more robust to sudden changes in data distribution. These models do have some limitations, as they might not be able to capture complex temporal relationships in sequential data as effectively as prediction models. Selecting appropriate model architectures and reconstruction error metrics can be challenging since training these types of models can be more computationally intensive compared to prediction models.

## 3.4 PERIODIC WINDOW-SIZE SELECTION

In pump time series data, periodicity is a common characteristic due to the cyclical nature of pump operations. The Periodic Window-Based technique involves dividing the time series data into fixed time windows, where each window corresponds to a complete cycle. This allows for the analysis of cyclic patterns and the identification of anomalies within each window [7].
The window size is an important hyperparameter when applying window-based techniques to time series data. The size determines the number of temporal data points included in each window segment. An optimal size balances factors like windows having sufficient context. Each window would become a single sample for training models like LSTM or CNNs to learn temporal patterns within that scope. The size with the highest accuracy on validation data could be selected for the model.

## 3.5 ROLLING WINDOW TECHNIQUE FOR MODEL TRAINING

The rolling window technique is a common approach for training models on time series data. It involves repeatedly selecting a fixed-size window of data and using it to train or update the model [13]. The window "rolls" or moves forward through the time series data, allowing the model to learn from different segments of the data over time. This technique is particularly useful when dealing with time-

dependent patterns and when you want to capture changing relationships in the data.

### 3.5.1 *Data Representation*

Suppose you have a time series dataset D with N sequential time points represented as $X = [x_1, x_2, x_3, \ldots, x_N]$, where $x_i$ represents the value at time $i$ in the time series.

### 3.5.2 *Fixed-Size Window*

Define a fixed-size window of length $W$ (e.g., $W = 132$ time points) that you will use for training. This window represents a segment of your time series data.

### 3.5.3 *Training and Prediction*

The rolling window technique involves the following steps:

1. **Initialization**: Start with the first $W$ data points, $X_1 = [x_1, x_2, \ldots, x_W]$, to initialize your model.
2. **Training**: Train your model using the data within the current window $X_t$. The model learns from this segment of data to make predictions.

$$M_t = f(X_t)$$

3. **Prediction**: After training, the model can make predictions for the next time point $x_{t+1}$ based on its learning from the current window.

$$\hat{x}_{t+1} = M_t(x_t, x_{t-1}, \ldots, x_{t-W+1})$$

4. **Rolling the Window**: Move the window one time step forward by removing the oldest data point and adding the next data point. For e.g., to move from $X_t$ to $X_{t+1}$, you shift the window as follows:

$$X_{t+1} = [x_2, x_3, \ldots, x_{W+1}]$$

5. **Repeat**: Repeat the training and prediction steps with the new window $X_{t+1}$. Continue this process until you have processed the entire time series.

Where:

$M_t$ represents the model's state or parameters at time $t$.

$\hat{x}_{t+1}$ is the predicted value for the next time point.

$X_t$ is the current window of data for training.

By using the rolling window technique, your model adapts to changing patterns in the time series data, making it suitable for tasks like anomaly detection where patterns may evolve over time. However, for distance-based models like the FastDTW, concept of a rolling window is only used to predict whether the sequence is anomalous or not.

## 3.6 CONCLUSION

Anomaly detection in pump time series data is crucial for maintaining the reliability of industrial systems. Distance-based models, prediction models, and reconstruction models offer various ways to detect anomalies. For instance, you could use a prediction model to capture the expected behavior of the data and a reconstruction model to flag anomalies that deviate significantly from the expected reconstruction.
In practice, the choice between these three approaches depends on factors such as the nature of the data, the types of anomalies you're trying to detect, the available computational resources, and the specific requirements of your application. The choice of technique depends on the specific characteristics of the data and the desired level of accuracy. Moreover, the Periodic Window-Size selection and Sliding Window techniques provide effective ways to handle the periodic nature of pump data and enable real-time monitoring.

# 4

# DATA

In this chapter, we divulge into the core component on which all the research is based. The data was obtained from a system resembling a household setup, with a pivotal component: the *SCALA2* pump. The subsequent sections in this chapter focus on unraveling the intricacies of the system, the data acquisition process, and the central focus of this thesis - anomaly detection on the abnormal data using algorithms in the Univariate Time series.

## 4.1 OVERVIEW OF THE SYSTEM

The data is acquired from a Lab setup that emulates a household environment, housing an important element - the pump. This pump plays a pivotal role in pushing water into the system and maintaining the system's equilibrium. Much like a circulatory system, the pump regulates pressure, ensuring optimal functionality within the confines of the system.
Moreover, the system has a reservoir from where water is pumped into the system and returned back for the optimal reserves. The reservoir, intricately linked with the pump, serves as a safeguard against pressure fluctuations.

The water pressure from the pump is detected using a pressure gauge, which is mounted right after the water leaves to pump to enter the system. The integrated sensor which maintains the checks on pressure readings before it leaves the system is unavailable to the user, hence a pressure gauge has been used to measure those pressure readings. In Fig 4, the blue box refers to the pressure gauge from which the pressure readings are being recorded and sent to the PLC. The data is recorded 24x7 by the systems and stored on a PLC. From the figure, it can be seen how the pump is connected to the Water Heater and a separate connection for cold water which is interconnected with various taps strategically positioned throughout the system to mimic a typical household environment. These taps, similar to the taps we find in our kitchens, bathrooms, and showers, represent essential outlets where water is accessed for diverse household needs. The *Grundfos SCALA2 Pump* serves as the heart of this water supply network, ensuring a consistent flow to each tap, mirroring the synchronized delivery of water to any residents.

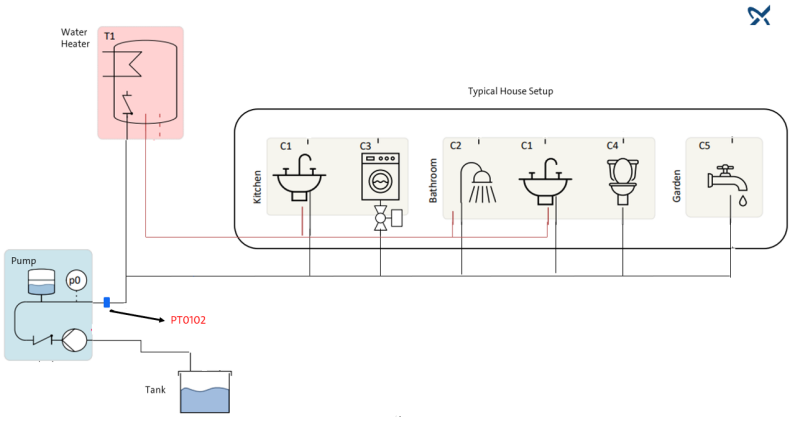



Figure 4: Lab setup fitted with Pressure Gauge.

## 4.2 DATA GENERATION

The data from this pump was gathered through a lab experiment which was performed in *Grundfos* Lab. The data that has been used to train the models is from where we are running the taps for around 2 hours. Multiple metrics from different sensors was recorded, few of these readings were from vibration, outlet pressure, and water flow sensor. Since the anomalies are present only in the Pressure sensor readings, other metrics were not used for training the models.
As the pressure gauge is mounted right after the water leaves the pump, the outlet pressure sensor values don't have any noise in them. We didn't have any issues with missing values and no outliers were detected during the running of the experiments. The user can refer to Table 1 for details on data collected from the system and the number of Anomalies present in the data.

Table 1: Data Summary

| | |
|---|---|
| Periodic Window size | 132-137 |
| Number of data points | 26800 |
| Number of Normal data Points | 19475 |
| Total number of Anomalies | 7325 |

## 4.3 DATA ACQUISITION

The data acquired is a time series dataset that involves multiple readings from a laboratory setup that simulates a typical household water usage system. This dataset is a multivariate time series, as it contains readings from various pressure, water flow, and pump speed sensors at different locations within the system.
The readings from these sensors are taken at regular intervals of 100ms, and we are specifically interested in analyzing the outlet pressure that is being maintained by a *Grundfos SCALA2 Pump* [10]. *SCALA2 Pump* is built to provide trouble-free operation, increasing the homeowner's peace of mind and comfort. To obtain accurate readings for this outlet pressure, we are focusing on the "*PT0102[Bar]*" sensor value(recorded by the pressure gauge 4), which is located right next to the pump outlet.

The readings recorded in the dataset by all kind of sensors provides a lot of information on the behavior of a household water system, but my experiments are based on identifying anomalies in the outlet pressure that is being maintained by the *SCALA2 Pump*. The reason we are interested in this data is because we are keen to understand how a pump would behave under conditions that are not normal. Using this data recorded from the pressure sensor, and applying various models we should be able to detect the anomalies. The anomaly detection on this data would prolong the life of the pump and necessary preventive measures can be taken to reduce the downtime of the system, additional repairing costs and also avoid any hassle to the homeowner.

The *SCALA2 Pump* is responsible for maintaining the outlet pressure in the household water system, and there are nine different settings with fixed intervals of 0.5 bars that range from 0.5 bar to 5.5 bar. The recommended outlet pressure under normal conditions is 3 bars, but higher settings can be used if the incoming water flow in the taps is low.
To simulate *real-world* systems, higher outlet pressure is required as in most households, , given the low incoming pressure from the city system or reservoir. The data has been collected at set points of *3.5 bars* or higher (refer to 5), which could provide additional insights into how the system might behave in different conditions.

The pump contains a precharge air pressure tank which allows the pump to maintain the pressure in the home system. Separate buffer air pressure tanks are used to maintain this tank's pressure. There is


Figure 5: Pressure Gauge reading at the time of the experiment, the pressure value reads 3.5 bars.

no sensor is present to monitor and measure the pressure of this tank. Before the start of the pump, the precharge pressure is maintained at 70% of the set discharge pressure. This pressure is referred to as $P_{out}$ pressure. In our experiments, since the discharge pressure was set to 3.5 bars, the precharge pressure i.e., $P_{out}$ was set to 2.45 bars. During this operation, normal data is generated. This data is a periodic signal that repeats around every 2 minutes with a window size from 132 to 137 data points, the same can be seen in figure 6.

Over time, this air pressure gradually vanishes from inside the pump, which plays one of the factors for anomalies to occur. The loss of the pressure could take days/months in the real-world depending upon

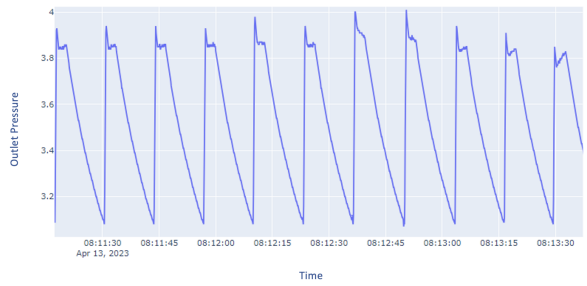



Figure 6: Normal Outlet Pressure readings from the Pressure Gauge.

the usage of the pump. To create the dataset with anomalies, the case of loss of air pressure was introduced by opening the valve which makes it to 0 bars. The loss of the precharge tank pressure forces the pump to run without stopping at an interval of 2 minutes. The data generated from this run looks like noise but shows the continuous running of the pump. In Figure 7, it can be seen Outlet pressure is constantly around 3.5 bars, only dropping in between for 2000ms because the IPC tries to take control of the system and stop the pump from running continuously.

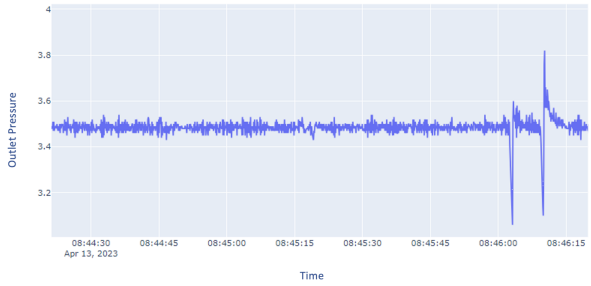



Figure 7: Abnormal data as recorded during the experiment.

In Figure 8, it is visible, that once the precharge air pressure from inside the pump starts to vanish, the periodic cycles are also lost. Once, the precharge air pressure vanishes completely, the pump starts to operate continually. This leads to an abnormal flow of water in the system and aging of the pump parts.

There are 2 different flows, low and normal flow for which anomalies are being considered. For my analysis, we focused on low flow which means the water flowing through various taps is less than 1.5 L. The irregular working of the pump, to maintain the pressure makes it a probable case for anomalies. During low flow of water from the taps

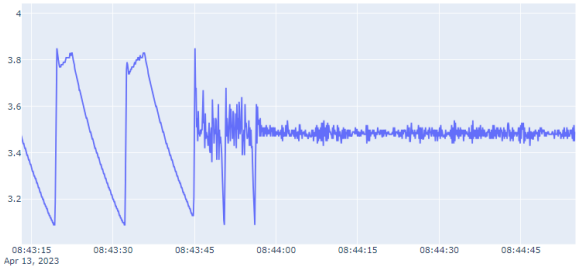



Figure 8: Abnormal data as recorded during the experiment.

and with the precharge air pressure missing, anomalies could be seen. The type of anomaly is context anomalies. The contextual anomalies show that the pump runs for a continuous time period to get to the set pressure but fails to do that.

To evaluate all the models on this data, we have used the F1 score and the accuracy metrics in *Chapter 5*.

# 5

# SYSTEM ARCHITECTURE AND EVALUATION

The application domain for anomaly detection using distance-based prediction and reconstruction models is diverse. In this chapter, we discuss the problem of anomaly detection in the pump using the existing techniques based on their methodology and the data transformation prior to anomaly detection and evaluation of these models.

## 5.1 OVERVIEW

Implementing these models for anomaly detection involves several practical considerations. Data pre-processing, including normalization and window creation, is crucial to ensure the model's optimal performance. Training the models involves selecting appropriate hyperparameters, radius, distance metrics, loss functions, and optimization techniques. Additionally, model evaluation requires defining anomaly detection thresholds, choosing performance metrics, and conducting cross-validation to assess generalization.

In each of the aforementioned sections, we provide a comprehensive exposition of the implemented algorithms, including a detailed breakdown of their inputs and outputs. We emphasize the models that exhibited superior efficacy and talk about the techniques employed for error thresholding that yielded optimal practical results.

Each model discussed in this chapter has its own set of advantages and limitations. Distance-based models, such as FastDTW, are better suited for capturing the dissimilarity between data points in the given 2 two time series. Prediction models, such as TCN and LSTM, excel at capturing sequential dependencies and are well-suited for scenarios where forecasting anomalies is essential but might struggle with long-range dependencies or sudden shifts in data distribution. Reconstruction models, like Autoencoders and LSTM Autoencoders, offer interpretability by focusing on data reconstruction but may struggle with complex temporal patterns.

With the exception of FastDTW, all the models were trained on 60% of the entire dataset, and further utilizing an 80:20 training-to-validation data split. The remaining 40% of the entire dataset is being utilized as a test set. During the training process, the models were compiled using the Adam optimizer with a learning rate of 0.001

and mean squared error (MSE) as the loss function. The training was conducted using the provided training data, denoted as X_train and y_train, for a duration of 20 epochs with a batch size of 64, a choice determined to yield optimal results for the anomaly detection task. The training procedure incorporated an early stopping callback that monitored the validation loss, halting training if there was no improvement for 5 consecutive epochs. This measure was taken to prevent overfitting and to retain the model with the best validation performance. Although the models were initially set to train for more than 20 epochs, they often terminated early due to the onset of overfitting during the training process.

For the Prediction-based models, we have used *MinMaxScaler* from *scikit-learn* to preprocess data for machine learning. It begins by creating a *MinMaxScaler* instance called "scaler". The training data, stored in the DataFrame df_training_value, is then scaled using the fit_transform method, which computes the minimum and maximum values for each feature in the training data and scales them to a range typically between 0 and 1. The scaled training data is saved in the variable scaled_train. Similarly, the test data in df_test_value is scaled using the same scaling parameters determined from the training data, and the scaled test data is stored in the variable scaled_test. Scaling ensures that all features have consistent scales, preventing any individual feature from dominating the learning process, which is essential for many machines as well as deep learning algorithms.

On the other hand, for reconstruction models, the data normalization is performed through standardization. This means it scales the data in a way that ensures it has a mean of 0 and a standard deviation of 1. Standardization ensures that the data has a well-defined mean and standard deviation, making it easier to interpret the scaled values for reconstruction models. It provides a more interpretable scaling, where a value of 2, for instance, indicates that a feature is two standard deviations away from the mean. Moreover, it is robust to outliers, making it suitable for data with extreme values that could otherwise skew results. Standardization also preserves the original distribution shape, which can be crucial since the data has a non-Gaussian distribution.

## 5.2 PERFORMANCE METRICS

To evaluate the performance of all the baseline models and TCN model, we adopt the Precision, Recall, F1 Score and Accuracy as metrics. These metrics are used to evaluate the quality of a model.
To calculate these metrics, we need to define the following terms:

- **True Positive (TP)** indicate the number of time points correctly identified as anomalous points.
- **True Negative (TN)** indicates the number of time points correctly identified as normal points.
- **False Positive (FP)** indicates the number of time points incorrectly identified as anomalous points which are actually normal points.
- **False Negative (FN)** indicates the number of time points incorrectly identified as normal points which are actually anomalous points.

From all these aforementioned terms, below we will explain each metric along with their respective formulas:

**Accuracy** is the ratio of the number of correct predictions to the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

**Precision** is the ratio of the number of true positives to the total number of true positives and false positives.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

**Recall** is the ratio of the number of true positives to the total number of actual positives.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

**F1 Score** is the harmonic mean of precision and recall, which tries to balance these two metrics. It ranges between 0 and 1, where 1 indicates perfect precision and recall.

$$F1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4}$$

**Confusion Matrix** A confusion matrix is a table through which all the above metrics can be calculated. The matrix is created to compare TP, FP, TN, FN, in cases where we are dealing with data which can classified into 2 categories like in our case, Normal and Anomaly. The matrix displays the true vs. predicted classifications for the dataset on which the evaluation is performed upon.

The ROC (Receiver Operating Characteristic) curve one of the metrics for evalauation of threshold, although valuable for assessing a binary classification model's overall discrimination ability, is not typically employed to choose the threshold when calculating metrics like

| | | Predicted | |
|---|---|---|---|
| | | Normal | Anomaly |
| True | Normal | TN | FP |
| | Anomaly | FN | TP |

Table 2: Confusion Matrix

Precion and Recall. Since the procured dataset is imbalanced (i.e., one class significantly outnumbers the other), optimizing precision and recall becomes critical. The ROC curve alone doesn't consider class imbalance and doesn't guide on selecting a threshold that maximizes the performance of the abnormal class.

## 5.3 DISTANCE BASED MODELS

### 5.3.1 *FastDTW*

Fast Dynamic Time Warping (FastDTW) model employs a sliding window approach for efficient time series similarity measurement. The distance-based model is often used in time series analysis to find an optimal alignment between two sequences. The "fast" in FastDTW indicates that it is an approximation or an optimized version of the original DTW algorithm designed to reduce its computational complexity. It aims to provide a quicker way to calculate the DTW distance while still maintaining a reasonable level of accuracy. The dataset was divided into 2 segments, "normal" which contains only normal data (60% of the whole pressure time series data), and "mixed" segment which has normal as well as abnormal data segments. We are extracting a "reference" segment which is labeled as normal as it only contains normal periodic data points. This segment serves as the benchmark against which other segments are compared. The window segment contains 132 data points, which contain a single periodic signal from the pressure time series data.

We are using the *fastdtw* Python package. FastDTW uses a distance metric to compute the "cost" of aligning two points from the sequences. We used *Euclidean distance* as the distance metric to calculate the cost of the warping path. To calculate the threshold, the reference window was compared to one of the other normal segments through the sliding window technique. We slide the normal segment also of 132 data points, one point in time, and calculate the mean of all the points in the reference window. The threshold is applied to the distance values to classify each segment as normal or abnormal. Distances larger than the threshold suggest that the current segment deviates significantly from the reference and is therefore flagged as an anomaly.

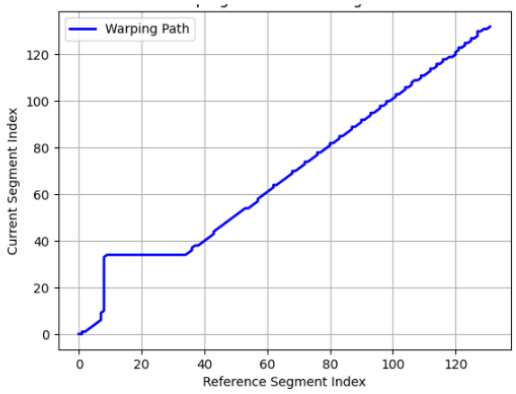



Figure 9: Warping Path for radius 12 in FastDTW.

To control the flexibility of the warping path, "radius" is another primary parameter for finding the warping path. Through multiple experiments for the selection of the right radius, we got the highest accuracy with radius value as *12*. With a smaller radius, the algorithm will consider fewer potential alignments, making it faster but resulting in more False Positive (FP). With a larger radius, the algorithm becomes slower making it similar to DTW and more False Negative (FN). In the figure 9, we can review the warping path computed by the fastdtw algorithm.

Once the threshold is calculated, through the rolling window, we calculate the "cost" for each window segment in the mixed signal with the reference segment. If the cost for the whole window is more than the set threshold, the data point is predicted as an anomaly otherwise normal. We computed a confusion matrix to compute the Precision, Recall, and F1 score for the test set which constitutes 40% of the whole dataset containing around 3200 normal points and 7300 anomalous points. The metrics require actual labels for the mixed data to compare against the predicted labels, which in our case is available. Figure 10 shows the confusion matrix for FastDTW model.

*Results:*

With this distance-based model, we achieved a *F1 Score* of 0.903 and an accuracy of 86.1% of the test data.

We have used FastDTW-based as a baseline model which offers a method to identify deviations or anomalies in a time series dataset by comparing segments of data against a predefined reference segment.

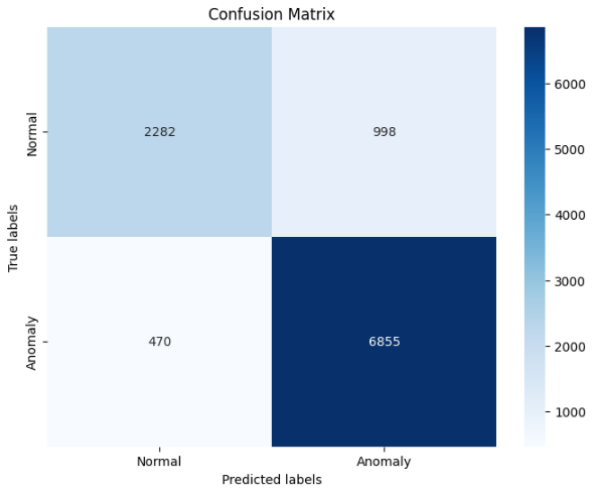



Figure 10: Confusion Matrix for the FastDTW.

The model performs well 86% accuracy and provides a starting point for anomaly detection, fine-tuning and validation against known data are crucial for optimal performance for these distance-based models. FastDTW's efficiency arises from its ability to approximate DTW by breaking the problem into smaller subseries. It recursively divides the time series data, computes DTW distances within smaller windows, and merges these distances to construct the final warping path. This divide-and-conquer strategy drastically reduces the computational burden, making it practical for large-scale pump time series anomaly detection.

## 5.4 PREDICTION MODELS

*Data Pre-processing*

We define a function for prediction-based models, `create_sequences`, used to create sequences for training models. This function takes as input a list of numerical values, denoted as `values`, and an optional parameter `time_steps` representing the length of the input sequences, which is set to 132 representing the length of the periodic signal in the pressure data. Within the function, two empty lists, X and `y`, are initialized to store the input-output pairs for training. A loop iterates through the `values` list, and for each iteration, it forms an input sequence X of length `time_steps` by extracting a slice of values from the current index `i` to `i + time_steps`. Simultaneously, it records the corresponding output value `y` at the position `i + time_steps`, effectively creating a supervised learning dataset where X contains sequences and `y` contains the values to be predicted. Finally, the function returns these input-output pairs as NumPy arrays, denoted as `X_train` and `y_train`, which can be used to train the prediction-based learn-

ing models, such as TCN, MLP and LSTM networks, for sequence prediction tasks.

*Threshold Selection*

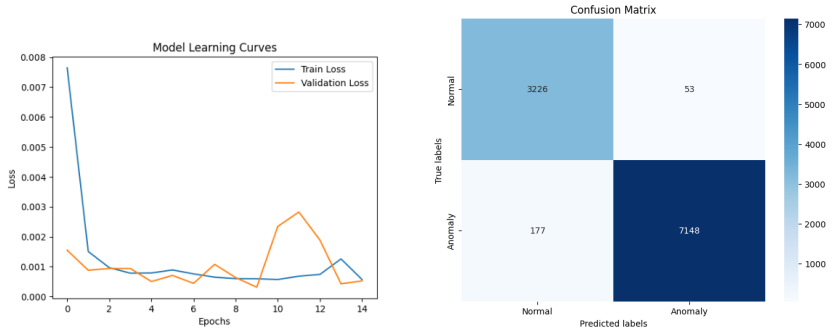
The thresholding technique, particularly for anomaly detection is helpful in the context of regression tasks such as prediction. It aims to identify instances where the model's predictions significantly deviate from the actual target values by considering the distribution of errors. We predict the models output as $y\_train\_pred$, which is then transformed back to their original scale using the inverse transformation provided by $scaler.inverse\_transform$. Similarly, the original target values $y\_train\_original$ are also inverse-transformed using the scikit-learn python library. The absolute errors between these predicted and original values are then computed as $train\_errors$. The threshold is set using statistical properties of these errors. In this case, it is calculated as the mean $np.mean(train\_errors)$ plus three times the standard deviation $3 * np.std(train\_errors)$ of the errors. This threshold effectively represents a range within which most of the errors fall. Predictions with errors exceeding this threshold are considered outliers or anomalies. This technique is valuable because it adapts to the data's characteristics, providing a dynamic threshold that accounts for variations and uncertainties in the prediction errors. It offers a more robust way to detect unusual predictions or potential model failures compared to fixed threshold values.

### 5.4.1 *Multilayer Perceptron model*

One particular configuration of the neural networks is feed forward neural networks (FFNN) more specifically the architecture called as Multilayer perceptron (MLP). It consists of a minimum of 3 layers: input, hidden and the output layer. It takes the time series sequence vector as an input and outputs a number. It can be a regression or a classification model depending on the activation function in the final, third, output layer. The neural network can be graphically defined as a combination of multiple connected units in one or more layers shown as a circle. The inbound arrow represents the input to that logical unit whereas the outbound arrow represents the output of the unit. There can be multiple inbound and outbound arrows coming to and from a single logical unit. The hidden layer of the network performs a mathematical operation on the input time series sequence vector and introduce non-linearity using an activation function such as ReLU (rectified linear unit) or tanh and the output layer has a single output unit which forecast the next time step. This FFNN being the vanilla

network does not include any convolution, skip, gated connections which other complex deep learning models adapt. In multilayer perceptron all outputs of one layer are connected to each input of the succeeding layer. This kind of architecture is called fully-connected.

The dense layer which is the output layer has one node (unit) which outputs value in range from [0,1], inverse transformation is performed to get the value in the same range as the original data because we normalize that data using *scikit-learn Min-MaxScaler* transformation function. Usually called as feature scaling performs a linear transformation and preserves the original distribution of data. The feature scaling avoid issues like model learning large weight values. A model with large weight values is often unstable. The target variable which we are trying to forecast if has large spread of values, may result in large error gradient values causing the weight values to change significantly in return making learning process unstable.




(a) Training & Validation Loss while training

(b) Confusion Matrix for MLP model

Figure 11: MLP Mean Square Error (MSE) Loss while training and Confusion Matrix.

The system architecture for the MLP (Multi-Layer Perceptron) model has been designed for prediction task and is constructed using the TensorFlow and Keras libraries. The model architecture consists of several layers, each serving a specific purpose to enable accurate predictions. The input layer is initialized with 132 units, which corresponds to the input shape defined by the variable TIME_STEPS(132, representing a periodic window). This layer employs the rectified linear unit (ReLU) activation function, facilitating the capture of nonlinear patterns within the data. To increase the model's capacity to learn intricate relationships in the data, two hidden layers, each containing a customizable number of hidden units (64 units in this case), are added consecutively. These layers are also activated with ReLU functions. Following the hidden layers, a third hidden layer with 32

units and ReLU activation is introduced, further compressing the data representation before reaching the output layer. The output layer consists of a single unit, suitable for this prediction task of anomaly detection where a continuous numerical prediction is required. Figure 11a, shows the training and the validation loss during the training.

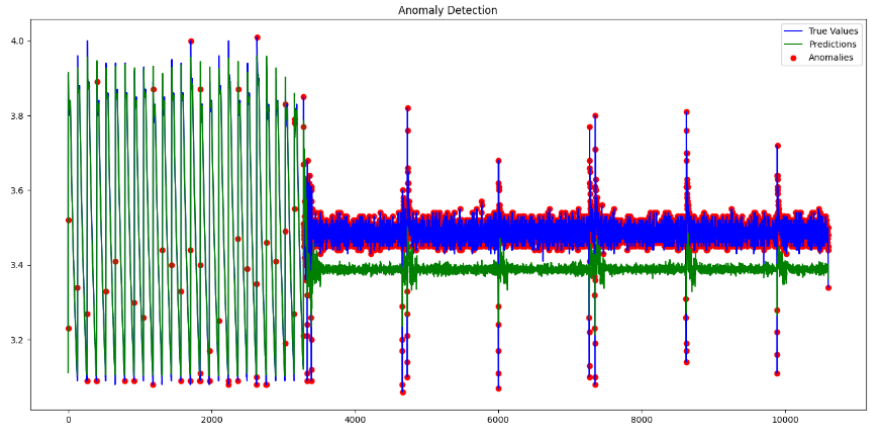



Figure 12: Prediction on Test dataset for MLP model.

*Results:*

Based on the aforementioned strategy for setting the Threshold, if the prediction error of the output is more than the set threshold, the data point is marked as an anomaly. In Fig 12, anomalies have been marked in "red" and fig 11b shows the confusion matrix. The MLP is able to achieve good results in separating out the abnormal data from the normal ones. With this prediction-based model, we achieved a *F1 Score* of 0.984 and an accuracy of 97.83% on the test data.

In summary, this MLP prediction model is a feedforward neural network architecture capable of learning complex relationships in the input data. It's a machine learning model without any kernels, strides and can be customized by adjusting the number of hidden units, dropout rates, and learning rates to optimize its performance for anomaly detection in various pump related sensor data.

### 5.4.2 *Temporal Convolution Network(TCN)*

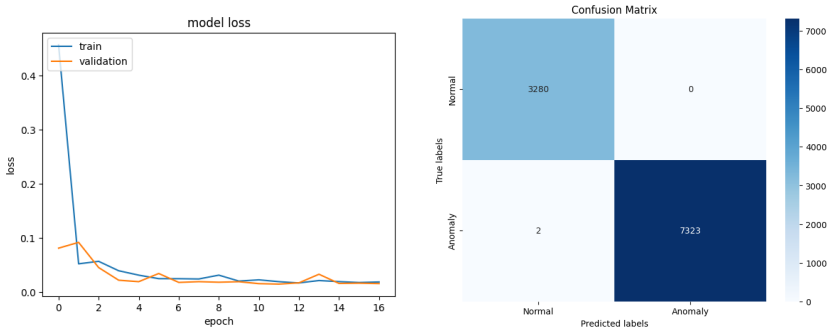
Temporal Convolutional Networks (TCN) is a deep learning architecture that leverages 1D convolutions to capture temporal dependencies within sequences effectively. They are specialized deep learning models designed for tasks like anomaly detection in sequential data, such as time series. TCNs utilize dilated convolutions and skip con-

nections to enhance their predictive capabilities. Dilated convolutions enable TCNs to efficiently capture information across various time scales, crucial for detecting anomalies occurring at different frequencies. With dilated convolutions, TCNs can maintain a broad receptive field without an excessive increase in model parameters. Additionally, skip connections address the vanishing gradient problem and facilitate the training of deep TCN models, enabling the capture of both high-frequency and low-frequency patterns effectively.

The architecture of the TCN model has been designed with the goal of making single-step predictions with the $return\_sequences = False$. The $nb\_stacks$ parameter is set to 1, which means there is only one stack of convolutional layers. Therefore, it's a single-layer TCN model. The model takes as input a sequence of data with multiple features, where the number of features is determined by the shape of the training data. This input layer is responsible for processing the temporal information within the data. The TCN model employs a stack of convolutional layers with dilated convolutions. The `nb_filters` parameter controls the number of filters in each convolutional layer which is fixed at 24, and the `kernel_size` parameter defines the size of the convolutional kernel, set to 7 in our case.
Importantly, the model uses dilations of increasing powers of 2, to capture temporal dependencies at different scales. The dilation parameter list corresponds to the number of convolutional layers in the model and has been set to

```
dilations=[2 ** i for i in range(5)]
```

meaning there are five convolutional layers in the TCN model, each with a different dilation rate: 1, 2, 4, 8, and 16. These dilation rates control the receptive field of each convolutional layer, allowing the model to capture temporal dependencies at different scales.
To facilitate gradient flow during training and capture both short-term and long-term dependencies, skip connections are introduced between convolutional layers. These connections enable the model to bypass certain layers, allowing for the fusion of features from different depths in the network. The `use_skip_connections` parameter controls the inclusion of these connections. The model is configured for a prediction task, so we have set the feature (`regression=True`). The TCN's output length matches the number of features in the input data, ensuring that the model produces predictions of the appropriate dimensionality. Figure 13a, shows the training and the validation loss during the training.




(a) Training & Validation Loss while training

(b) Confusion Matrix for TCN model

Figure 13: TCN Mean Square Error (MSE) Loss while training and Confusion Matrix.

*Results:*

Based on the aforementioned strategy for setting the Threshold, if the prediction error of the output is more than the set threshold, the data point is marked as an anomaly. In Fig 14, anomalies have been marked in "red" and fig 13b shows the confusion matrix. TCN performs really well on the test data, reporting only 2 False Negatives. With this prediction-based model, we achieved a *F1 Score* of 0.9999 and an accuracy of 99.98% on the test data.

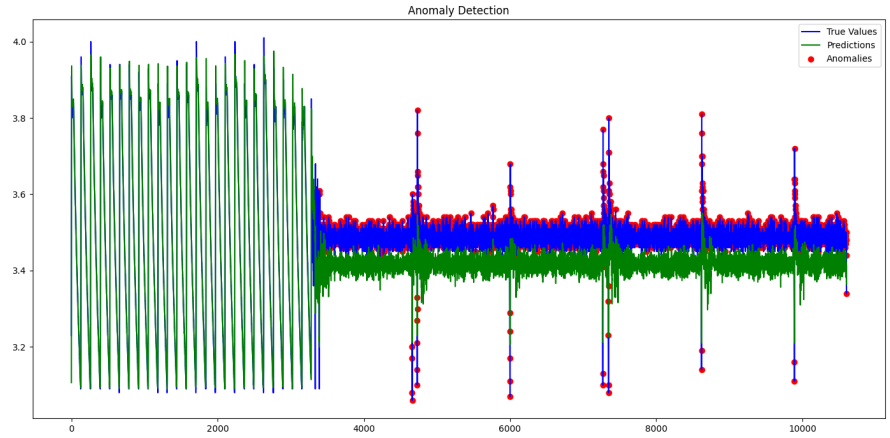


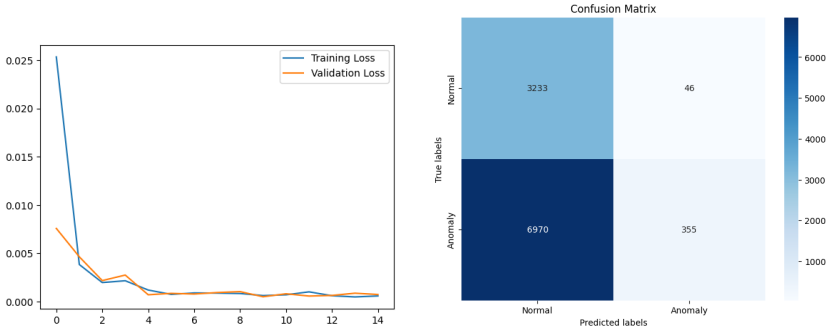

Figure 14: Prediction on Test dataset for TCN model.

In summary, this TCN model is designed as a deep learning architecture as it leverages dilated convolutions, skip connections, and configurable hyperparameters to effectively capture temporal dependencies and produce real-valued predictions.

### 5.4.3 *Long Short-Term Memory(LSTM)*

LSTM (Long Short-Term Memory) prediction models are a type of recurrent neural network (RNN) architecture designed for sequence prediction tasks. Unlike traditional RNNs, LSTMs are capable of capturing long-term dependencies in sequential data, making them well-suited for applications such as time series forecasting, and anomaly detection. These models consist of LSTM layers that process input sequences, with each layer capable of retaining and updating information over time. The use of LSTM layers enables the model to learn and predict patterns within sequential data, making them a powerful tool for tasks where understanding temporal relationships and context is essential.

The model architecture for this experiment, consisted of 3 LSTM layers and the final single Dense Layer. The first LSTM layer with 64 units takes input sequences with a shape of $(\mathrm{TIME\_STEPS}, 1)$, where input sequence is of 132 data points. The second LSTM layer also consisted of 64 hidden units. The $\mathtt{return\_sequences} = \mathtt{True}$, setting in the first two LSTM layers allows the model to process variable-length sequences, which is essential when dealing with data of varying lengths. The third and the final LSTM layer, consisted of 32 hidden units and the $\mathtt{return\_sequences} = \mathtt{False}$ for this layer, this produces a fixed-length representation of the sequence. The final Dense layer with one unit is used for prediction task. This is the model's output, which will be compared to a threshold for anomaly detection. Figure 15a, shows the training and the validation loss during the training.




(a) Training & Validation Loss while training

(b) Confusion Matrix for LSTM model

Figure 15: LSTM Mean Square Error (MSE) Loss while training and Confusion Matrix.

*Results:*

Based on the aforementioned strategy for setting the Threshold, if the prediction error of the output is more than the set threshold, the data point is marked as an anomaly. In Fig 16, anomalies have been marked in "red" and fig 15b shows the confusion matrix for LSTM. Being the right choice for processing the sequenctial data, LSTM produced poor results. With this prediction-based model, we achieved a *F1 Score* of 0.091 and an accuracy of 33.84% on the test data.

In summary, the LSTM model is technically configured to process sequential data efficiently and to capture long-term dependencies but in this dataset, the model reported the most number of False Negatives and wouldn't be the right choice in case of periodic data sequences.

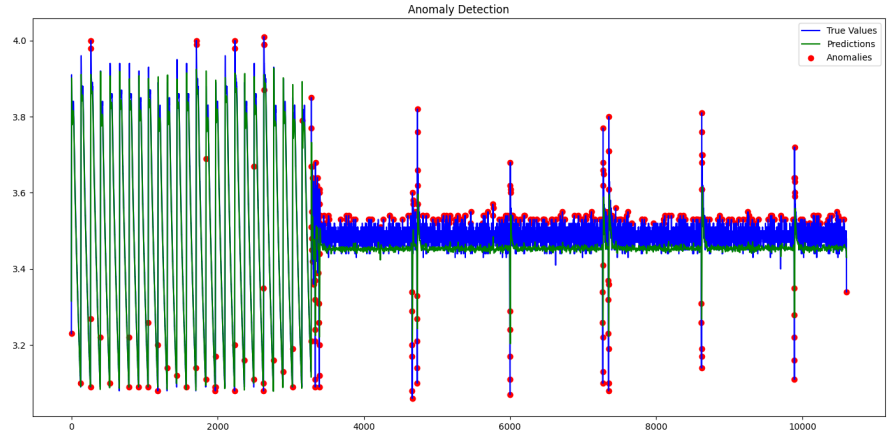



Figure 16: Prediction on Test dataset for LSTM model.

## 5.5 RECONSTRUCTION MODELS

*Data Pre-processing*

For the reconstruction-based models, the creation of training sequences is emphasized for the model to learn and reproduce sequential patterns. We set the TIME_STEPS = 132, which defines the length of the sequences to be created. A function `create_sequences` takes a list of numerical values, denoted as `values`, and an optional parameter `time_steps`, which specifies the length of the input sequences. Within the function, a loop iterates through the `values` list. For each iteration, it creates an input sequence of length `time_steps` by extracting a subsequence of values from the current index `i` to `i + time_steps`, effectively sliding a window over the data. Each of these subsequences is then appended to the `output` list. Finally, the `output` list, which contains these input sequences, is converted into a NumPy array using

*np.stack(output)*. The resulting X_train variable represents the collection of input sequences that can be used to train a reconstruction-based model.

*Threshold Selection*

The threshold selection criteria is different than the prediction-based models. We calculate maximum mean absolute error (MAE) loss of the training data when it is reconstructed by an autoencoder. It involves two main steps:

- First, the MAE loss is computed for each data point in the training set by taking the absolute element-wise difference between the original training data ($X_{train}$) and its corresponding reconstruction ($x_{train\_pred}$) using the formula:

$$\text{train\_mae\_loss} = \frac{1}{N}\sum_{i=1}^{N}\left|X_{train}^{(i)} - x_{train\_pred}^{(i)}\right|$$

  Here, $N$ represents the number of data points in the training set.

- After computing the MAE loss for each data point, the threshold is set to the maximum of these individual MAE losses:

$$\text{threshold} = \max(\text{train\_mae\_loss})$$

  This thresholding technique helps in detecting unusual or unexpected observations in the data.

### 5.5.1 *AutoEncoder*

AutoEncoders are a type of neural network architecture used for unsupervised learning and dimensionality reduction. They are primarily designed for feature learning and data compression. AutoEncoders consist of an encoder and a decoder, and their main objective is to reconstruct the input data from a lower-dimensional representation. The training of an AutoEncoder is based on minimizing a reconstruction loss. A common choice for the loss function is the Mean Squared Error (MSE). The encoder maps the input data $x$ to a lower-dimensional latent representation $z$:

$$z = f_{encoder}(x)$$

Here, $f_{encoder}$ represents the encoding function, which typically consists of multiple layers of neurons (commonly feedforward neural networks) with activation functions like ReLU or sigmoid. The decoder reconstructs the data from the latent representation:

$$\hat{x} = f_{decoder}(z)$$

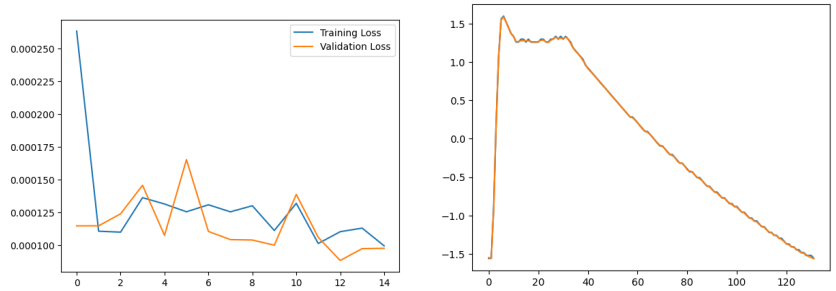
Similarly, $f_{decoder}$ is the decoding function, which aims to recover the input data. It also consists of one or more layers with appropriate activation functions.

The AutoEncoder model architecture is implemented using the Keras library and comprises several layers designed to encode and subsequently decode input data. The input layer is defined with a shape corresponding to the dimensions of the training data. This serves as the entry point for the data. Two Convolutional 1D layers are then added successively. The first layer applies 32 filters with a kernel size of 7, using a stride of 2 and the Rectified Linear Unit (ReLU) activation function. The second layer follows a similar structure but with 16 filters. These layers aim to capture and extract meaningful features from the input data.

Next, two Convolutional 1D Transpose layers are introduced for the decoding phase. These layers perform the reverse operation of the previous convolutional layers. They upscale the features learned by the encoder, starting with 16 filters and then 32 filters, both using a kernel size of 7, stride of 2, and ReLU activation. These layers are responsible for generating a reconstructed version of the input data. Finally, a Convolutional 1D Transpose layer with a single filter and kernel size of 7, along with no activation function, serves as the output layer. It produces the final output of the AutoEncoder, representing the reconstructed data. The loss function MSE measures the difference between the original and reconstructed data, providing a quantifiable measure of how well the AutoEncoder can replicate the input. Figure 17a, shows the training and the validation loss during the training. The reconstructed window segment of 132 points from the TCN-AE model can be visualized In Fig 17b.

*Results:*

Based on the aforementioned strategy of setting the Threshold, we calculate the reconstruction error for the output, if the error is more than the threshold are marked as an anomaly otherwise normal. In Fig 18, anomalies have been marked in "red" and fig 19 shows the confusion matrix. The AutoEncoder achieves exceptional results in separating out the abnormal data from the normal ones. With this reconstruction-based model, we achieved a *F1 Score* of 0.9999 and an




(a) Training & Validation Loss while training

(b) Prediction for first window from the AutoEncoder Model

Figure 17: AutoEncoder Mean Square Error (MSE) Loss while training and Prediction on the first normal segment.

accuracy of 99.98% on the test data.

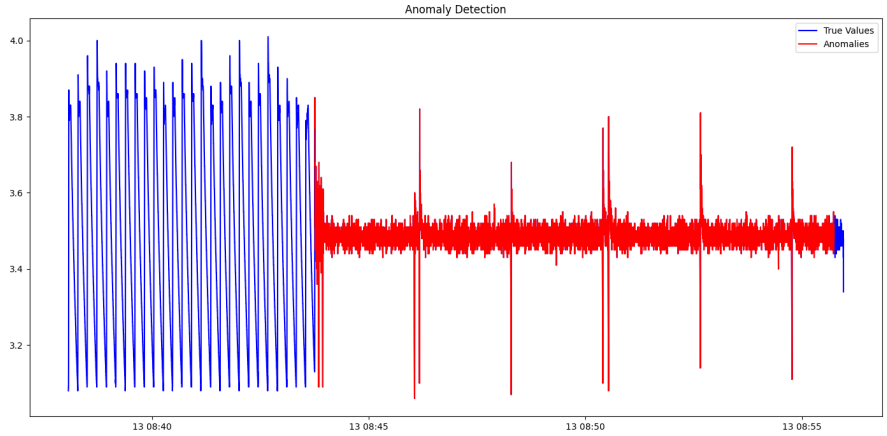



Figure 18: Prediction on Test dataset for AutoEncoder model.

In summary, the AutoEncoder model encodes input data through convolutional layers, compresses it into a lower-dimensional representation, and then reconstructs the original data through transposed convolutional layers. The model's architecture and parameters are configured to optimize the reconstruction of the input data while reducing its dimensionality.

### 5.5.2 *Temporal Convolution Network AutoEncoder(TCN-AE)*

The TCN-AE (Temporal Convolutional Network Autoencoder) is a neural network architecture designed for sequence data processing and reconstruction. It combines the principles of autoencoders and Temporal Convolutional Networks (TCNs). The encoder part of the TCN-AE takes an input sequence X of length T, which in our case is set to 132 data points and processes it through a stack of TCN layers.

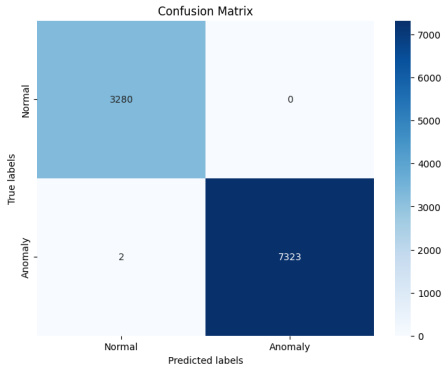



Figure 19: Confusion Matrix for AutoEncoder model.

Each TCN layer applies dilated convolutions to capture temporal dependencies at different scales. The dilated convolution operation for a TCN layer with dilation factor $d$ can be represented as:

$$y_i = \sum_{k=1}^{K} w_k \cdot x_{i-d \cdot k}$$

where $x_i$ is the input at position $i$, $y_i$ is the output at position $i$, $w_k$ are the learned weights, and $K$ (set as 3) is the kernel size.

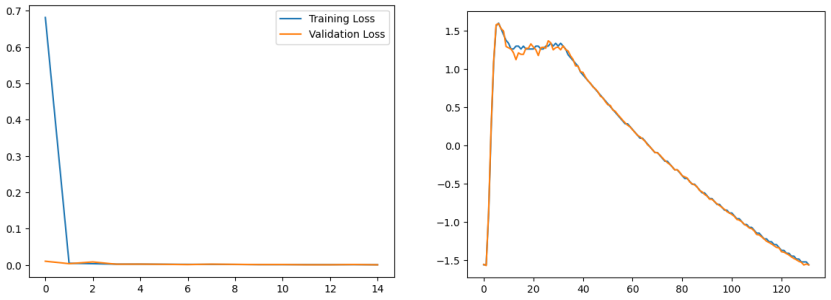
The decoder part of the TCN-AE aims to reconstruct the input sequence from the encoded representation. Similar to the encoder, it consists of TCN layers with dilated convolutions. The goal is to minimize the reconstruction error, typically measured as the mean squared error (MSE), between the input $X$ and the reconstructed output $X'$:

$$\text{MSE} = \frac{1}{T} \sum_{i=1}^{T} (X_i - X_i')^2$$

The architecture for the model during the training consists of the encoder section, the input sequences are processed through two TCN layers with 64 and 32 units, respectively, where the TCN layers have specific dilations of $[1, 2]$ and use the ReLU activation function. The dilations parameter helps the TCN capture both local and global dependencies in the input sequences. The latent space, represents a compressed representation of the input data obtained from the encoder layers. This latent representation captures the most important features of the input sequence.

In the decoder section, the encoded data is reconstructed by passing it through two TCN layers with 32 and 64 units with the similar dilations used in the encoder layer, matching the architecture of the encoder. Finally, a 1D convolutional layer with one filter and linear activation is applied to produce the output sequence with the same shape as the input. Figure 20a, shows the training and the validation

loss during the training. The reconstructed window segment of 132 points from the TCN-AE model can be visualized In Fig 20b.




(a) Training & Validation Loss while training

(b) Prediction for first window from the TCN-AE Model

Figure 20: TCN-AE Mean Square Error (MSE) Loss while training and Prediction on the first normal segment.

*Results:*

Based on the aforementioned strategy of setting the Threshold, we calculate the reconstruction error for the output, if the error is more than the threshold are marked as an anomaly otherwise normal. In Fig 21, anomalies have been marked in "red" and fig 22 shows the confusion matrix. The TCN-AE achieves exceptional results in separating out the abnormal data from the normal ones. With this reconstruction-based model, we achieved a *F1 Score* of 0.9999 and an accuracy of 99.98% on the test data.

Overall, the TCN-AE is trained to encode the input sequence into a lower-dimensional latent space and then decode it back to reconstruct the original sequence. This architecture is effective for anomaly detection in the current sequential data.

### 5.5.3 *Long Short-Term Memory AutoEncoder(LSTM-AE)*

An Long Short-Term Memory AutoEncoder (LSTM-AE) also a reconstruction model, consists of two main parts: the encoder and the decoder which consists of typically one or more LSTM layers. The LSTM layers are chosen for their ability to capture temporal dependencies and patterns within sequential data, making them well-suited for anomaly detection tasks where anomalies often manifest as deviations from expected sequential patterns. These parts are used to compress input data into a lower-dimensional representation and then reconstruct the original data. Anomalies that deviate significantly from this learned representation are more likely to result in reconstruction

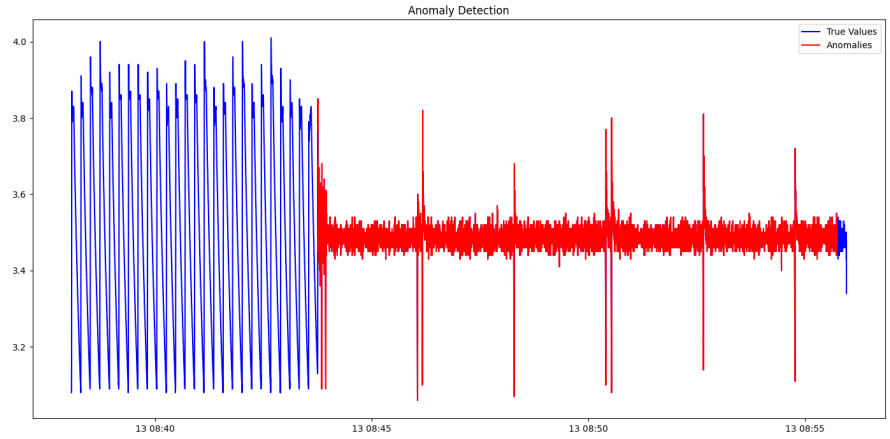



Figure 21: Prediction on Test dataset for TCN-AE model.

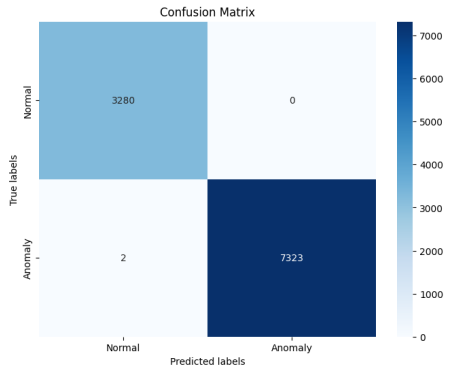
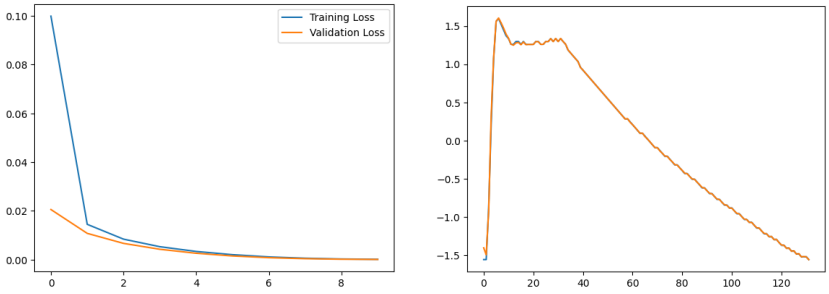



Figure 22: Confusion Matrix for TCN-AE model.

errors, enabling their detection. The encoder part of the LSTM-AE typically consists of at least one or more LSTM layers. LSTM cells in these layers have three gates: the input gate (i), the forget gate (f), and the output gate (o). The LSTM cell updates its internal state (c) and produces an output (h) based on the input data (x), the previous cell state ($c_{prev}$), and the previous hidden state ($h_{prev}$). By encoding and then decoding the input data sequences, the model learns a compact representation of normal data.

The architecture for the model during the training consists of the encoder section which has a single LSTM layers consisting of 64 hidden units and the return sequences is configured to True. The encoder takes input sequences with the space specified in $input\_shape$. This LSTM layers captures relevant features and patterns from the input sequences. The next layer is another LSTM layer consisting of 32 hidden units and is also set to $return\_sequences = True$ which act as a latent-space representation of the input sequences. It continues the encoding process by learning a lower-dimensional representation of the data while maintaining the sequence structure. This layer serves as an input to the decoder. The third LSTM layer has 64 LSTM units

and is configured to return sequences as well. This layer serves as the decoder, attempting to reconstruct the input sequences from the lower-dimensional representation learned by the previous layers. The final layer is a Dense layer with the number of units equal to the width of the input data (specified by `input_shape[1]`). It uses a linear activation function ('linear') to produce continuous output values. This layer acts as the decoder's output layer, aiming to reconstruct the input data sequences. Figure 23a, shows the training and the validation loss during the training. The reconstructed window segment of 132 points from the LSTM-AE model can be visualized In Fig 23b.




(a) Training & Validation Loss while training

(b) Prediction for first window from the LSTM-AE Model

Figure 23: LSTM-AE Mean Square Error (MSE) Loss while training and Prediction on the first normal segment.

*Results:*

Based on the aforementioned strategy of setting the Threshold, we calculate the reconstruction error for the output, if the error is more than the threshold are marked as an anomaly otherwise normal. In Fig 24, anomalies have been marked in "red" and fig 25 shows the confusion matrix. The LSTM-AE achieves exceptional results in separating out the abnormal data from the normal ones. With this reconstruction-based model, we achieved a *F1 Score* of 0.9992 and an accuracy of 99.89% on the test data.

Overall, the LSTM-AE is trained to encode the input sequence into a lower-dimensional latent space and then decode it back to reconstruct the original sequence. This architecture is effective for anomaly detection in the current sequential data.

The results obtained for the performance metrics, have been summarised and discussed in Chapter 6, for the all the considered models.

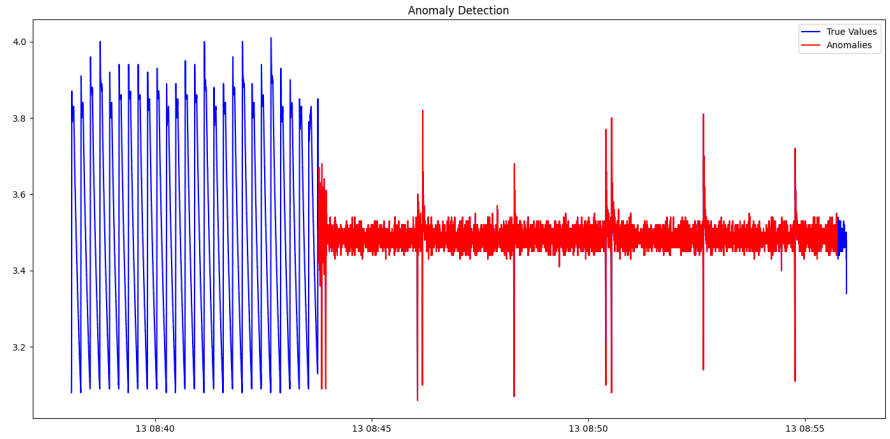



Figure 24: Prediction on Test dataset for LSTM-AE model.

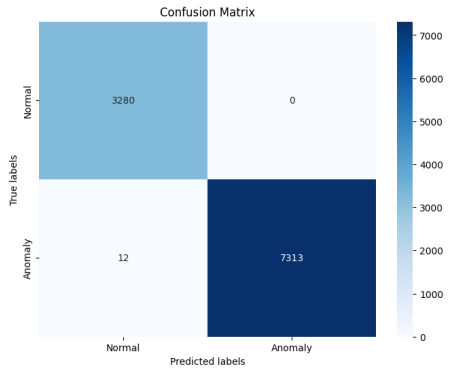



Figure 25: Confusion Matrix for LSTM-AE model.

## 5.6 CONCLUSION

In this chapter, we delved into the application of distance-based, prediction, and reconstruction models for the task of anomaly detection in univariate time series data. Specifically, we explored the utilization of FastDTW, MLP, TCN, Autoencoders, LSTM, and LSTM Autoencoder and TCN Autoencoder. These models leverage distinct approaches, such as warping distance, prediction, and reconstruction, to identify anomalies and capture intricate temporal patterns within pump data. The incorporation of a window-based technique further amplifies their performance by facilitating detection of localized anomalies. An in-depth comprehension of the underlying theoretical principles, practical implementations, as well as the inherent advantages and constraints of these models equips researchers and industry professionals with the requisite knowledge to judiciously

choose and apply anomaly detection methodologies tailored to the demands of their specific pump-related use cases.

# 6 RESULTS AND FUTURE WORK

In this work, we analyzed pressure signals from *Grundfos SCALA2 Pump* to detect anomalies in the time series data. We tested out various models based on their techniques to predict output and set the chosen threshold to detect the anomalies.

We evaluated different models for anomaly detection, including distance-based model, FastDTW, Prediction Models, like MLP (MultiLayer Perceptron), TCN (Temporal Convolutional Network), LSTM (Long Short-Term Memory) and Reconstruction models, like AutoEncoder, LSTM-AE (LSTM-based AutoEncoder), and TCN-AE (TCN-based AutoEncoder). Among these models, TCN and TCN-AE achieved exceptionally high F1 scores of 0.9999, indicating their outstanding ability to detect anomalies in the pump data. Table 3, shows all the stats from different experiments performed on the Pump time series data.

As stated in the hypothesis in Chapter 1, TCN would emerge as the most effective model for pump anomaly detection has been validated by the results. TCN not only outperformed other models but exceeded our expectations, achieving an exceptional F1 score of 0.9999, accuracy of 99.98%, and perfect precision. This reaffirms the suitability of TCN for the crucial task of pump anomaly detection.

| **Model** | **F1 Score** | **Accuracy %** | **Precision** | **Recall** | **True Positive (TP)** | **False Negative (FP)** | **True Negative (TN)** | **False Positive (FN)** |
|---|---|---|---|---|---|---|---|---|
| FastDTW | 0.903 | 86.16 | 0.872 | 0.935 | 6855 | 470 | 2282 | 998 |
| MLP | 0.984 | 97.83 | 0.992 | 0.975 | 7148 | 177 | 3226 | 53 |
| TCN | **0.9999** | **99.98** | **1.000** | **0.9997** | 7323 | 2 | 3280 | 0 |
| LSTM | 0.091 | 33.84 | 0.885 | 0.048 | 355 | 6970 | 3233 | 46 |
| AutoEncoder | **0.9999** | **99.98** | **1.000** | **0.9997** | 7323 | 2 | 3280 | 0 |
| LSTM-AE | 0.9992 | 99.89 | **1.000** | 0.9984 | 7313 | 12 | 3280 | 0 |
| TCN-AE | **0.9999** | **99.98** | **1.000** | **0.9997** | 7323 | 2 | 3280 | 0 |

Table 3: Anomaly Detection Models Comparison for Techniques like Distance, Prediction, and Reconstruction.

The reported instances of False Positives in the TCN-AE (Temporal Convolutional Network AutoEncoder) models were observed during

periods when the pressure in the precharge tank began to decrease. This occurred because the models were unable to accurately detect anomalies in these situations. Notably, even as the pressure decreased, there remained a periodic signal in the data, which posed a challenge for the models in distinguishing anomalies.
Similarly, the MLP (Multilayer Perceptron) model also had difficulty identifying data points as anomalies when they occurred during periods when the IPC (Integrated Process Control) system assumed control and attempted to raise the outlet pressure.
In contrast, both the LSTM (Long Short-Term Memory) model and FastDTW (Fast Dynamic Time Warping) method successfully detected anomalies in cases when the IPC system assumed control and when the precharge tank pressure started to decrease. However, there were still instances of False Positives during the phase when the outlet pressure($P_{out}$) began to diminish.
To mitigate these False Positives during the decrease of $P_{out}$ it may be beneficial to adjust the anomaly detection threshold during this specific period. This adjustment could help reduce the number of erroneous anomaly alerts during this phase.

In terms of Accuracy and Precision, TCN, AutoEncoder, and TCN-AE models all achieved accuracy levels above 99.98%, demonstrating their capacity to accurately classify pump behavior as normal or anomalous. Additionally, these models including the LSTM-AE attained perfect precision, indicating that they rarely misclassify normal instances as anomalies (no False Positives). LSTM was the worst model with the least accuracy and the lowest F1 score. LSTM reported 90% of the anomalies as False Negative meaning as missed anomalies. TCN was the only Prediction model, able to attain accuracy higher than 99%

In terms of Recall, TCN and TCN-AE models achieved a very high recall score, indicating their ability to effectively identify true positives (anomalies) while minimizing false negatives (missed anomalies). This is crucial for pump anomaly detection as it ensures that few anomalies go undetected. In comparison with other baseline models, FastDTW and MLP, while achieving relatively good F1 scores and accuracy, lag behind TCN and TCN-AE in terms of precision and recall. LSTM, while having a high precision, has a low recall, indicating that it tends to miss many anomalies.

In summary, the results suggest that TCN and TCN-AE models are the top performers for pump anomaly detection, providing a robust solution for identifying and addressing pump abnormalities with high accuracy and minimal false alarms. These models are capable of accurately identifying anomalies, making them suitable for

real-time monitoring and maintenance of pumps. These models offer promising prospects for improving pump reliability and reducing maintenance costs in at least the given household settings.

In terms of future research, we would like to explore more alternative architectures using different Anomaly detection approaches. Further improvement, lies in the fine-tuning of the threshold used for anomaly detection. While our study achieved the desired results, further optimization of the thresholding technique would potentially enhance the model's performance. Investigating adaptive thresholding methods or exploring dynamic thresholds based on specific pump operational conditions would be a promising direction for future work.

In conclusion, our study not only affirms the superiority of TCN for pump anomaly detection but also suggests that refining the thresholding strategy could lead to even more accurate and robust anomaly detection systems.

# BIBLIOGRAPHY


[1] SM Abdullah Al Mamun and Mehmet Beyaz. "LSTM recurrent neural network (RNN) for anomaly detection in cellular mobile networks." In: *Machine Learning for Networking: First International Conference, MLN 2018, Paris, France, November 27–29, 2018, Revised Selected Papers 1*. Springer. 2019, pp. 222–237.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." In: *arXiv preprint arXiv:1803.01271* (2018).

[3] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A Lozano. "A review on outlier/anomaly detection in time series data." In: *ACM Computing Surveys (CSUR)* 54.3 (2021), pp. 1–33.

[4] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. "Deep learning for anomaly detection in time-series data: review, analysis, and guidelines." In: *IEEE Access* 9 (2021), pp. 120043–120065.

[5] Zahra Zamanzadeh Darban, Geoffrey I Webb, Shirui Pan, Charu C Aggarwal, and Mahsa Salehi. "Deep learning for time series anomaly detection: A survey." In: *arXiv preprint arXiv:2211.05244* (2022).

[6] Diab M Diab, Basil AsSadhan, Hamad Binsalleeh, Sangarapillai Lambotharan, Konstantinos G Kyriakopoulos, and Ibrahim Ghafir. "Anomaly detection using dynamic time warping." In: *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. IEEE. 2019, pp. 193–198.

[7] Arik Ermshaus, Patrick Schäfer, and Ulf Leser. "Window Size Selection in Unsupervised Time Series Analytics: A Review and Benchmark." In: *International Workshop on Advanced Analytics and Learning on Temporal Data*. Springer. 2023, pp. 83–101.

[8] Jakub Gęca. "Performance comparison of machine learning algotihms for predictive maintenance." In: *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska* 10 (2020).

[9] Saroj Gopali, Faranak Abri, Sima Siami-Namini, and Akbar Siami Namin. "A comparative study of detecting anomalies in time series data using lstm and tcn models." In: *arXiv preprint arXiv:2112.09293* (2021).

[10] Grundfos. *SCALA2*. https://product-selection.grundfos.com/dk/products/scala/scala2?tab=models. Online; [Accessed: (13-March-2023)]. 2022.

[11] Yangdong He and Jiabao Zhao. "Temporal convolutional networks for anomaly detection in time series." In: *Journal of Physics: Conference Series*. Vol. 1213. 4. IOP Publishing. 2019, p. 042050.

[12] Kavina Kananesan. *Basic Principles of Preventive Maintenance*. https://www.nrx.com/principles-preventive-maintenance/. Online; [Accessed: (19-July-2023)]. 2019.

[13] Lattawit Kulanuwat, Chantana Chantrapornchai, Montri Maleewong, Papis Wongchaisuwat, Supaluk Wimala, Kanoksri Sarinnapakorn, and Surajate Boonya-aroonnet. "Anomaly detection using a sliding window technique and data imputation with machine learning for hydrological time series." In: *Water* 13.13 (2021), p. 1862.

[14] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. "Revisiting time series outlier detection: Definitions and benchmarks." In: *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 1)*. 2021.

[15] Shuyu Lin, Ronald Clark, Robert Birke, Sandro Schönborn, Niki Trigoni, and Stephen Roberts. "Anomaly detection for time series using vae-lstm hybrid model." In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Ieee. 2020, pp. 4322–4326.

[16] Jianwei Liu, Hongwei Zhu, Yongxia Liu, Haobo Wu, Yunsheng Lan, and Xinyu Zhang. "Anomaly detection for time series using temporal convolutional networks and Gaussian mixture model." In: *Journal of Physics: Conference Series* 1187.4 (2019), p. 042111. DOI: 10.1088/1742-6596/1187/4/042111. URL: https://dx.doi.org/10.1088/1742-6596/1187/4/042111.

[17] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. "Long Short Term Memory Networks for Anomaly Detection in Time Series." In: *Esann*. Vol. 2015. 2015, p. 89.

[18] Anvardh Nanduri and Lance Sherry. "Anomaly detection in aircraft data using Recurrent Neural Networks (RNN)." In: *2016 Integrated Communications Navigation and Surveillance (ICNS)*. Ieee. 2016, pp. 5C2–1.

[19] Adherbal Caminada Netto, Arthur Henrique de Andrade Melani, Carlos Alberto Murad, Miguel Angelo de Carvalho Michalski, Gilberto Francisco Martha de Souza, and Silvio Ikuyo Nabeta. "A novel approach to defining maintenance significant items: A hydro generator case study." In: *Energies* 13.23 (2020), p. 6273.

[20] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. "Deep learning for anomaly detection: A review." In: *ACM computing surveys (CSUR)* 54.2 (2021), pp. 1–38.

[21] Marina Paolanti, Luca Romeo, Andrea Felicetti, Adriano Mancini, Emanuele Frontoni, and Jelena Loncarski. "Machine learning approach for predictive maintenance in industry 4.0." In: *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. IEEE. 2018, pp. 1–6.

[22] Stan Salvador and Philip Chan. "Toward accurate dynamic time warping in linear time and space." In: *Intelligent Data Analysis* 11.5 (2007), pp. 561–580.

[23] Manassakan Sanayha and Peerapon Vateekul. "Fault detection for circulating water pump using time series forecasting and outlier detection." In: *2017 9th International Conference on Knowledge and Smart Technology (KST)*. IEEE. 2017, pp. 193–198.

[24] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan 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*. 2019, pp. 2828–2837.

[25] Markus Thill, Wolfgang Konen, and Thomas Bäck. "Time series encodings with temporal convolutional networks." In: *International Conference on Bioinspired Methods and Their Applications*. Springer. 2020, pp. 161–173.

[26] Yuanyuan Wei, Julian Jang-Jaccard, Wen Xu, Fariza Sabrina, Seyit Camtepe, and Mikael Boulic. "LSTM-autoencoder-based anomaly detection for indoor air quality time-series data." In: *IEEE Sensors Journal* 23.4 (2023), pp. 3787–3800.

[27] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications." In: *Proceedings of the 2018 world wide web conference*. 2018, pp. 187–196.

[28] Yi-Fan Zhang, Peter J Thorburn, and Peter Fitch. "Multi-task temporal convolutional network for predicting water quality sensor data." In: *International Conference on Neural Information Processing*. Springer. 2019, pp. 122–130.

# ACKNOWLEDGMENTS

Thanks to Panagiotis Karras, Henrik Pederson, Dávid Nagy, Rasmus Engholm for helping me and guiding me throughout the thesis work. I am deeply grateful for the invaluable support provided by my supervisors throughout the course of my thesis work. Their expertise and encouragement were instrumental in making this thesis possible. Thanks to my family and friends who supported me throughout the entire time.

# DECLARATION

I hereby declare that this thesis is my original work, and all sources and references have been duly acknowledged and cited. This work has not been submitted for any other degree or publication.

*Aarhus, September 2023*

Chahat Gupta