Unsupervised Anomaly Detection and Interpretation for Unstructured Sensor-Generated Multivariate Time Series Data

Zhu, Huilin

Award date:
2021

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science

Unsupervised Anomaly Detection and Interpretation for Unstructured Sensor-Generated Multivariate Time Series Data

Master Thesis

Huilin Zhu

Supervisors:
Mike Holenderski
Simon Koop
Bert van Willigen

Final Version

Eindhoven, The Netherlands
December 2021

Contents

Abstract

In industrial scenarios, many machines are monitored by multivariate time series sensor data. Therefore, detecting anomalies from this type of data is important for industries to detect machine malfunctions. But there are two obstacles preventing the usage of many state of the art machine learning anomaly detection algorithms on this type of data. First is that the time series sensor data generated from machines are not always structured, and thus difficult for machine learning models to make use of them. Second is that it is often not enough to only detect the timestamp where an anomaly has occurred, but finding out the sensors that are causing the anomaly is also important. In this research, we propose an anomaly detection pipeline that addresses these two problems. It transforms unstructured time series datasets to a structured format that is usable by most anomaly detection models. We evaluate the performance of the pipeline with a real-world dataset and show that the pipeline preserves anomaly detection precision, and it is also able to find most anomalous sensors that are responsible for the occurrence of anomalies.

1. Introduction

Anomaly detection is the identification of unusual data points. It is a highly valuable step of data analysis because anomalies can reveal crucial information, especially for time series datasets streamed from various sensors, like industrial machine sensors and hospital patient monitoring sensors. Anomalies in such datasets can be indications of machine failures and sudden worsening of patient's conditions. The presence of anomalies can also indicate measurement errors, which need to be handled before further data analysis. In addition, finding out the relevant sensors that cause an anomalous event not only narrows down the target for further examinations, but can also provide insights for engineers on how to solve the issue.

Despite the abundance of existing anomaly detection models and algorithms for time series, to the best of our knowledge, they are only applicable on a specific type of time series data, where the timestamps of all sensors are aligned, and the values within each sensor are logged at a uniform interval. However, such "well-behaved" datasets are not guaranteed in many real-world scenarios.

Time series datasets can be roughly categorized as either structured or unstructured. A structured time series is one where the rows and columns are timestamps and attributes (e.g. sensors) respectively, having the following characteristics:

- For each row in the dataset, all attribute values are logged at the same timestamp: $\mathbf{x}_{t,\cdot} = (x_{t,1}, x_{t,2}, ..., x_{t,M})$ where t is a unique timestamp and M is the number of attributes contained in the dataset.
- The values of an attribute are all logged at the same time interval: $t_{i+1} - t_i = c$ for all i where c is a constant representing a fixed time interval.
- No missing values are present.

Unlike structured datasets, in an unstructured time series, timestamps across attributes do not necessarily align, logging time intervals within an attribute can be irregular and vary over time, and the number of values logged for an attribute can differ from another attribute. Figure 1 shows an example for each of these two dataset types.

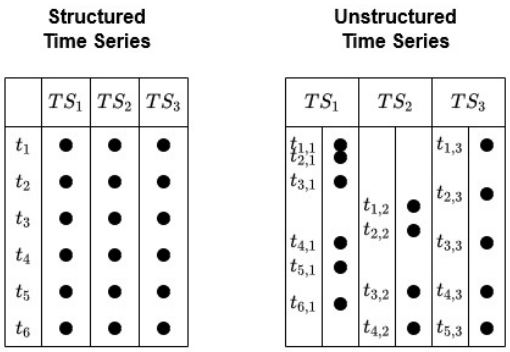


Figure 1: Structured time series (left) and unstructured time series (right). t_i is i^{th} timestamp, and TS_m is m^{th} attribute.

Direct transformation of an unstructured dataset into a structured format would not create a more applicable dataset for machine learning models, as this would create a sparse dataset filled with missing values. Figure 2 demonstrates why this would be the case. Suppose the data of two sensors, TS_A and TS_B, are stored in an XML file, where sensor TS_A records a value every 3 seconds, and sensor TS_B records a value every 1 second, then we have a data shown on the left part of the

figure. It is unstructured because logging time varies across sensors and cannot be used by machine learning models. If we were to directly transform it into any tabular format, like CSV, then we would need to align the data points with respect to timestamp, and then the data shown on the right of the figure is obtained. For this mini example, even with a small logging interval difference between the sensors and with only two sensors, the sparsity of the combined data is already 50%. If we were to directly transform real-world unstructured sensor data with hundreds of sensors and with a vast difference between logging intervals, then an even more sparse dataset would be produced.

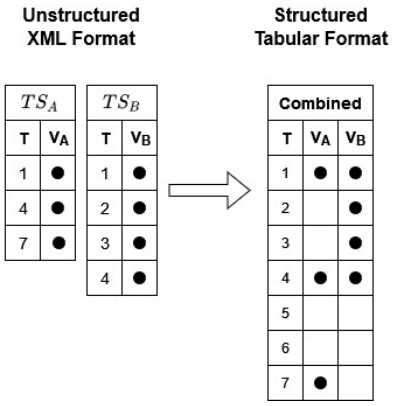


Figure 2: Directly transforming unstructured data to any tabular format time series data recognized by machine learning algorithms would create sparsity.

1.1. Task Description

Overall, there are three main tasks we aim to tackle in this research. The first is to propose a solution for anomaly detection on unstructured time series datasets, the second is to interpret the detected anomalies by identifying the sensors responsible for each detected anomaly, and the last is to apply our solution on Thermo Fisher sensor data as a use case to find and interpret anomalies.

1.1.1. Anomaly Detection on Unstructured and Unlabeled Sensor Time Series Datasets

For the task of anomaly detection in real-world time series sensor datasets, we face the following challenges.

- **Lack of Directly Applicable Models**: Data generated from sensors are often unstructured; thus, it cannot be processed directly by anomaly detection models. Direct transformation of such unevenly-timestamped data into evenly-spaced format is not an option because it would result in a sparse dataset.
- **Complex and Sparse**: Sensor data often have high dimensionality, as hundreds of sensors can be used to monitor an entity and values are logged constantly. This increases computational complexity and amplifies data sparsity if missing values are present.
- **Multivariate**: The use of multiple sensors implies that the corresponding dataset is multivariate, where the data type can be different from one sensor to another. This means that the model we use should have the ability to learn well whether the data is numerical, categorical, or a mixture of both.
- **Unsupervised**: Real-world datasets often do not have labeled training data, and it is labor intensive to create one. Thus, the methods we choose should use unsupervised learning.
- **Time-dependent**: Time is an important factor for anomaly detection, so time information should be preserved as much as possible in formulating our methods.

As a result, no existing anomaly detection method can be used directly on this type of dataset while overcoming the challenges listed above.

1.1.2. Interpretation of Detected Anomalies

Once an anomaly is detected, it is useful to identify sensors that are responsible for the occurrences of anomalies because it helps technicians to quickly diagnose the source of the problem and fix it. Therefore, finding responsible sensors for each predicted anomaly is the second task.

1.1.3. Apply Proposed Solution to Thermo Fisher Use Case

The datasets from Thermo Fisher are streams of timestamped values generated from health monitoring (HM) sensors from the numerous models of electron microscope systems. For brevity this type of data is referred to as the HM data for the rest of the thesis.

The solution proposed for the above tasks is applied on a HM dataset to detect and interpret anomalies from the company's electron microscopes.

1.2. Research Questions

With the tasks defined, we can formulate the research questions for this thesis: How to detect anomalies on an unstructured multivariate time series without labeled training data, characterized by irregular logging intervals both within and between sensors? In addition, once anomalies are detected, how to interpret them such that the sensors responsible for each anomaly are identified?

1.3. Contributions of this Research

In this section, we list the contributions of this research.

- Propose a novel anomaly detection pipeline that takes an unstructured time series and outputs detected anomalies along with their responsible sensors. The pipeline is composed of two frameworks: The preprocessing framework transforms an unstructured time series dataset into a structured dataset while retaining time and value information from the original data; consequently, a wide range of anomaly detection algorithms become accessible and can be applied efficiently on the data. The postprocessing framework interprets detected anomalies, by using the error scores of the anomaly detection model, to identify the sensors that contribute to a positive anomaly prediction.
- Evaluate the performance of the proposed pipeline using a state of the art anomaly detection model on a labeled unstructured multivariate time series dataset.
- Apply the pipeline on a real-world dataset generated from Thermo Fisher electron microscopes to detect and interpret anomalies.

2. Related Work

2.1. Unsupervised Anomaly Detection Techniques

The definition of an anomaly in literature often varies with the focus of study at hand; hence there are several definitions for an anomaly. To summarize, an anomaly is an observation that deviates remarkably from other members of the dataset, and this deviation can be measured either in terms of distance, density, or probability of occurrence [14] [15]. As a result, various anomaly detection algorithms are developed to detect anomalies that fit these definitions. In addition, the usage of deep neural networks on the problem of anomaly detection introduces new possibilities, such as classification-based, prediction-based, and reconstruction-based methods [14], into this field. This section provides a brief summary of some of the main anomaly detection techniques.

In distance-based anomaly detection algorithms, anomalies are detected-based on their distances to nearest neighbors. For example, the K-Nearest Neighbors (KNN) algorithm is a prominent unsupervised distance-based anomaly detection algorithm. For an observation, the model computes its distances to K-nearest neighbors, and if the distance is large, the observation is predicted as an anomaly. However, the drawback of this method is that it is very computationally expensive for high dimensional data, like sensor data, with a running time of $O(n^2M)$ where n is the number of observations and M is the number of attributes [14]. Also, as Talagala et al. pointed out, if two anomalous clusters are close to each other but far away from the rest of the dataset, then they could protect each other from being detected since they are nearest neighbors for one another [13].

Clustering-based algorithms can also be used to detect anomalies unsupervisedly. This technique generally clusters the observations into multiple groups, then calculates some kind of metric for each cluster to determine which cluster and data points are anomalies. Various such clustering algorithms exist, like K-Means and DBSCAN. Although these are popular techniques, they are most adept in finding clusters, not outliers [2]. For example, K-Means centroids can be dragged away by outliers, and if several anomalous observations form a cluster, they can be masked as normal data points.

Local Outlier Factor (LOF) is a density-based unsupervised anomaly detection algorithm based on KNN [2]. For each observation, it computes a density score reflecting the local density deviation with respect to its neighbors. And if an

observation has a much lower density than its neighbors, it is predicted as an anomaly. Since the density score for an observation is an average of ratios between its own density to its K-nearest neighbors' densities, it is hard to determine a threshold for that is suited for all anomalies. Another drawback is that the running time for LOF $O(n^2)$ as it is required to find nearest neighbors for each observation [2].

Classification-based anomaly detection algorithms are also widely used. One-Class Support Vector Machine (OCSVM) is a popular unsupervised model from this category [11]. It aims to find a hyperplane separating the data from the origin of its feature space, such that the marginal distance between data and the hyperplane is minimized. Any observation on the opposite side of data is considered an anomaly. A parameter ν is used to adjust the number of anomalies allowed.

For probability-based techniques, Bayesian Belief Networks are examples of unsupervised anomaly detection algorithm. As introduced in [15] and [5], such models use data points surrounding the observation of interest as evidence, and then compute the probability of observation with maximize a posterior approach.

Lastly, there are also various deep neural network anomaly detection models, and they have shown promising results in recent years. One state of the art anomaly detection model for time series dataset is Long Short-Term Memory (LSTM) networks. Since its model weights are reused through time, it is particularly suited for time series data; thus, we will be using this type of model for evaluating our pipeline, and it is described in more detail in the following section.

2.1.1. LSTM-based Unsupervised Anomaly Detection Models

An LSTM is a variant of Recurrent Neural Network (RNN). Different from RNN cells, an LSTM cell uses gates to mitigate the exploding/vanishing gradient problem of RNN. Through gates, some information is retained while others are forgotten, and this selection process is optimized by training the model. This design allows LSTM networks to learn long term dependencies of sequences, and makes them particular adapted for time series datasets [8].

LSTM autoencoders [12] and LSTM predictive models [7] are both used for unsupervised anomaly detection. For LSTM autoencoder, during encoding, the input sequence goes through several LSTM encoding layers until it reaches a bottleneck where there are fewer neurons, and thus forcing the model to learn the most important features of the data, called latent representation. During decoding, the decoder attempts to reconstruct the original input from its learned latent representation. Anomalies are observations where their reconstructions deviate significantly from the original sequence. For LSTM predictive model, during training, sequences of $\mathbf{x}_{[t-T,t]}$ are inputs and $\mathbf{x}_{t+1}$ are targets. A trained model is able to generalize well the behavior of normal data, and thus during inference time, if the model's prediction is far off the input sequence, then the input is an anomaly.

2.1.2. Requirements on Input Data

All the anomaly detection models introduced so far have some common requirements on input data.

- Unstructured data is not allowed. As the definition for an observation is not clear when values do not align across attributes.
- Sparsity and missing values are not allowed. One reason is that distances between observations cannot be calculated when the dataset is sparse. Another reason is that, since anomalies are observations in sparse regions by definition, Under sparse data conditions, each data point can seem to lie in equally sparse locations as other data points [14].

2.2. Triplet Network

Triplet network, introduced by [4], is a type of neural network designed to learn a distance metric that captures the semantics of the data. The network learns through comparing three types of inputs; they are anchor, positive, and negative samples. The goal of the model is to learn a distance metric such that the distance between the anchor and positive samples are smaller than the distance between the anchor and negative samples, i.e. $d(x^{ref}, x^{pos}) \leq d(x^{ref}, x^{pos})$ where $x^{ref}, x^{pos}, x^{neg}$ represent anchor, positive, and negative samples respectively. To achieve this goal, triplet loss is used to train the model. The generic form of triplet loss is $Loss = max(||f(x^{ref}) - f(x^{pos})||^2 - ||f(x^{ref}) - f(x^{neg})||^2 + \alpha, 0)$ where hyperparameter α is used to make the network more robust [9].

2.3. Missing Value Imputation

Missing values can be caused by human error, i.e. the input value is not recorded by mistake; or they can be missing by design, i.e. there is no value to record at certain timestamps, and result in sparse data. Health monitoring sensor data often falls into the latter category, as measurement frequencies vary across different sensors. Many imputation techniques exist to fill missing values with numbers that represent well of the data.

A commonly used method for imputation involves filling the missing values with a statistic of the variable. Mean, median and mode are among the most popular statistics for this purpose. Forward and backward-filling are used when it is assumed that the missing values are the same as their predecessors or successors. More sophisticated methods for imputation include predicting missing values by linear regression or deep neural networks like bidirectional recurrent neural networks [1]. Although these techniques are all capable of filling missing values, a common problem is that, unless data are missing at random, imputation disguises data missingness pattern and thus cause information loss [6].

2.4. Interpretation

Interpretability is the degree to which a human can understand why a machine learning model makes certain predictions. It is important because it enables a human to examine the reasons behind decisions to ensure that those decisions are fair, reasonable, and trustworthy. There are two approaches to interpretation – the conventional meaning of interpretation and the use case-specific industry demand of interpretation.

Conventionally, model decisions can be interpreted by either using an interpretable model, like logistic regression and decision trees, or by applying model agnostic methods to decode black-box models in order to understand feature interactions, feature importance, and feature attributions [10].

On the other hand, use case-specific industrial demand for interpretation can sometimes be different from the conventional definition of interpretation. For example, for Thermo Fisher, their interest is to match the detected anomalies with call records that document customer complaints whenever there are potential microscopic malfunctions. Therefore, it is valuable for the company to obtain information from the model's anomaly predictions and identify sensors that contribute to the occurrences of anomalies. Since the research question of this thesis is stemmed from the real-world challenges put forward by Thermo Fisher, interpretation in this thesis is defined according to their specific industrial demand.

3. Preprocessing and Postprocessing Frameworks

We introduce an anomaly detection pipeline that consists of a preprocessing framework and a postprocessing framework. The objective of the pipeline is to first transform an unstructured time series dataset into a structured time series dataset for anomaly detection, then interpret the predicted anomalies by identifying relevant sensors that contribute to the positive predictions. There are four processing steps to the pipeline, and they are described in this section. Figure 3 shows a flowchart of the whole pipeline.

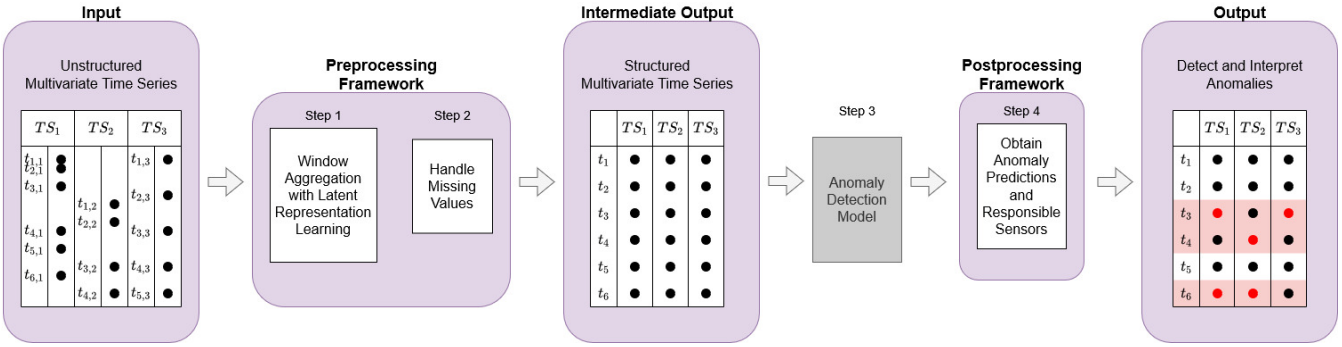


Figure 3: Flowchart of the proposed pipeline. It takes in an unstructured time series, uses unsupervised methods to process the dataset, and outputs predicted anomalies (red rectangular highlights) and interpret their responsible sensors (red circles).

3.1. Forethought of Designing the Pipeline

Before presenting the pipeline, there are several considerations worth mentioning, as they affect our choices in the pipeline design.

Firstly, data processing techniques, as well as anomaly detection models, range from attribute-wise detection to entity-wise detection (all attributes included). Certain values can only be perceived as anomalies if the values of all other attributes are considered. For example, a machine with a small fluctuation of fan speed might seem normal on its own, but given the condition that the machine has been off during this period, then such fluctuation is evidently an anomaly. Therefore, even though a multivariate time series can be seen as a dataset consisting of multiple individual univariate time series, our proposed pipeline will only use techniques that process the multivariate time series as a whole.

Secondly, directly transforming the unstructured data to a structured format and then imputing the empty spaces is not a viable option. Real-world sensor data, like the ones from Thermo Fisher, can have very different logging intervals across hundreds of sensors – as low as 1 second and as high as a few days. As discussed in section 1, direct transformation of such data would create high sparsity, 99.63% for Thermo Fisher dataset, and imputing the resulting empty spaces would lead to a dataset that consists of mostly imputed values rather than original data records. The pipeline we proposed avoids such a problem by circumventing the step of direct transformation.

3.2. Step 1 (Preprocessing): Create Evenly Spaced Time Series Dataset

One of the main difficulties is the irregular spacing of sensor values, because it makes most machine learning models inapplicable for the data. We tackle this problem using the window aggregation technique and representation learning with a triplet loss network.

To construct an evenly spaced time series, we use the window aggregation technique to align timestamps among sensors and create regular timestamp spacing within a sensor. Window aggregation is an approach for extracting insights from a span of data. For this technique, it is important to select a suitable window span that matches the intended purpose of the dataset. For example, the window span is different when anomalies need to be identified per every hour compared to per every second. For our datasets, the electron microscope experts from Thermo Fisher decided that it is sufficient to indicate anomalies per every hour, i.e. for values within each hour of the wall clock, we predict them as anomalous or not. Therefore, we aggregate the values of each hour into a single value. For other use cases, the window span needs to be adjusted based on field knowledge.

For aggregation, mean and median are commonly used. However, when the window span contains many values and when they exhibit complicated patterns, mean and median are not sufficient to capture the information contained in the time series within the window span. In this research, we are going to experiment using latent representation of values as the aggregation function, i.e. for a window of W values $\mathbf{x}_{t\in[t_1,t_W]} = (x_{t_1}, ..., x_{t_W})$ of one sensor, its window aggregated output is $f(\mathbf{x}_{t\in[t_1,t_W]}, \theta)$ where f is a triplet loss encoder proposed in [3], $f(\mathbf{x}_{t\in[t_1,t_W]}, \theta)$ is the encoding of input $\mathbf{x}_{t\in[t_1,t_W]}$, and θ is the parameters of the encoder.

There are several advantages of triplet loss learning which propel us to choose this encoder. Firstly, it can be trained under a fully unsupervised setting, which suits the reality of our datasets. Secondly, we can control the meaning of the learned metric space by selecting training samples. To train a meaningful embedding without labels, the authors of the paper uses time-based sampling to learn the various patterns of the input time series. Figure 4 illustrates the idea of time-based sampling. An anchor, $\mathbf{x}^{ref}$, of random length L where $0 < L \leq length(TS_m) \in \mathbb{N}$, is randomly chosen from a time series TS_m within the dataset. A positive sample, $\mathbf{x}^{pos}$ is then chosen within the anchor, such that $\mathbf{x}^{pos}$ is contained in $\mathbf{x}^{ref}$. Then, K negative samples, $\mathbf{x}^{neg}_{k\in[1,K]}$ are randomly chosen from different time series of the dataset. Then a triplet loss function, equation 1, is used for training the encoder. As can be seen from the equation, it rewards the model to maximize the similarity between the representations of $\mathbf{x}^{ref}$ and $\mathbf{x}^{pos}$, and at the same time maximize the dissimilarity between the representations of $\mathbf{x}^{ref}$ and $\mathbf{x}^{neg}_{k\in[1,K]}$, where the similarity between two representations is measured as their dot product scaled by the log sigmoid function.

$$
\begin{aligned}
loss = &- \log(\sigma(f(\mathbf{x}^{ref}, \theta)^T f(\mathbf{x}^{pos}, \theta))) \\
&- \sum_{k=1}^{K} \log(\sigma(-f(\mathbf{x}^{ref}, \theta)^T f(\mathbf{x}^{neg}_k, \theta))) \qquad (1)
\end{aligned}
$$

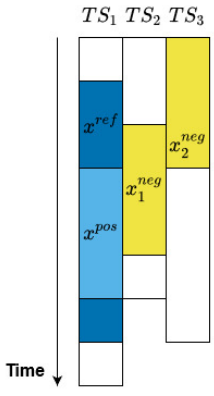


Figure 4: Example of time-based sampling used by the triplet loss encoder to learn time series patterns [3].

where σ is the sigmoid function, K is the number of negative samples per reference sample, and $f(\mathbf{x}, \theta)$ is a neural network with parameters θ.

As a result, the model learns the patterns of various sub-sequences contained in the dataset. This is advantageous for our purpose. Because our goal of using an encoder is to find a number that is able to properly represent the values within the span of a window, such that information of the original time series is retained by this number as much as possible. To obtain such a number, we train the encoder to learn a representation of length 1 for each window sequence $\mathbf{x}_{t\in[t_1,t_W]}$. Time-based sampling and triplet loss ensure that the value patterns of a window are distinguished against other time series patterns in the dataset, and thus obtaining a meaningful number for window values.

3.2.1. Δt Column to Capture Time Information

Time information in a time series is as important as sensor values, especially for unevenly spaced timestamped time series, as the spacing between logging times conveys extra information about the sensor. To prevent the loss of time information during the transformation of the dataset from unevenly spaced to evenly spaced, we introduce an extra column, $\Delta\mathbf{t}$, for each sensor. This column consists of time differences between the logging times of consecutive sensor values. An example of obtaining $\Delta\mathbf{t}$ column is shown in figure 5a. We use $\Delta\mathbf{t}$ instead of actual timestamps of the sensor because the time intervals between the log of values are more informative than which date or hour the values are logged.

Figure 5: (a) An example of obtaining $\Delta\mathbf{t}$ column from the timestamp column for a sensor m. When encoding the data, only value columns and $\Delta\mathbf{t}$ columns are used. Timestamp columna are discarded. (b) Time-based sampling with $\Delta\mathbf{t}$.

To incorporate $\Delta\mathbf{t}$ information into triplet loss encoding, both the value and Δt are selected together as a sample during time-based sampling, indicated in figure 5b.

As proposed in [3], the samples are then passed into a dilated causal convolutional neural network [16] with 10 layers each followed by weight normalization and leaky ReLU activation function. Then, global max pooling is applied on each output channel to extract channel representation. Lastly, a linear transformation is applied to reduce the representation to the desired length. In our case, the desired length is 1. As mentioned previously, we wish to aggregate a window of values into a single value. In this way, for each input time series sample, a representation value is obtained. Figure 6 shows the structure of model architecture.

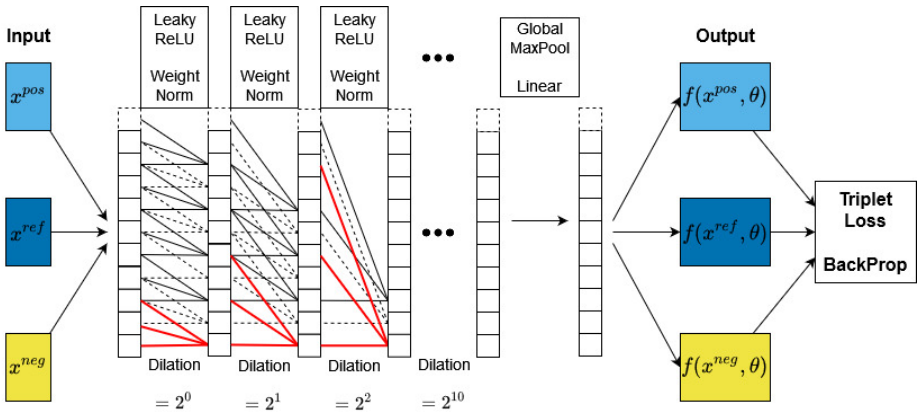


Figure 6: Architecture of dilated causal convolutional triplet loss network.

3.3. Step 2 (Preprocessing): Handle Missing Values

The output of step 1 is an evenly spaced time series with possible missing values present. The dataset is not completely filled because it is possible that for certain aggregation windows, there are no values belonging to the hour they cover; thus, the output of these windows are empty, i.e. NaN.

Handling missing values is a topic on its own, and there are a plethora of existing methods to fill empty spaces, such as the ones discussed in section 2.3. There is not a single best method to handle missing values for every scenario, and depending on the dataset and domain knowledge, one should select a method that fits the problem at hand. After consulting with Thermo Fisher engineers, we learned that a sensor logs a value (and its timestamp) whenever there is a significant change in value. Therefore, if a value is missing at a timestamp, it is reasonable to assume that the value is the same since last logged, and forward-fill is the most suitable method for our use case.

3.3.1. Indicator Column to Capture Missingness Information

Whichever method is chosen for data imputation, some information is lost after data is filled. This is because the missingness of time series contains information of when and how often a sensor detects a change, which could be an important factor for anomaly detection. Therefore, we use indicator columns [6] to retain missingness information. According to the authors of the paper, RNN models trained with indicator columns outperforms those without.

Suppose a value at time t of sensor m is missing, i.e. $x_{t,m} = NaN$, then our forward-fill strategy is as follows:

- If there is at least one previously recorded value, then we set $x_{t,m}$ to the last recorded value.
- If there is no value recorded before $x_{t,m}$, then we set $x_{t,m} = median(\mathbf{x}_{\cdot,m})$.

If a sensor only contains one value after filling, then it is discarded, as a constant line does not contain information for anomaly detection.

At the same time, indicator columns are used to capture missingness. For each value $x_{t,m}$, if it is forward-filled, then its indicator $I_{t,m} = 1$; otherwise, its indicator $I_{t,m} = 0$. Figure 7 shows an example data that is filled with this strategy.

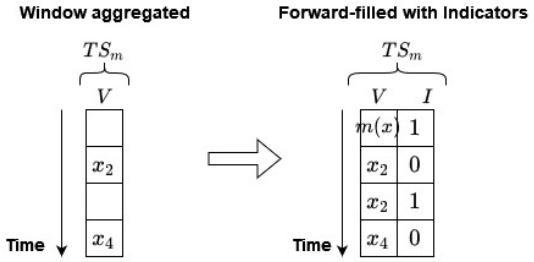


Figure 7: Example of forward-fill with indicator columns. V denotes sensor values, and I denotes sensor indicator column.

3.4. Step 3: Pass the Preprocessed Data to Anomaly Detection Models

The output of step 2 is a dataset that is preprocessed by our proposed pipeline and ready for anomaly detection using a model of choice. However, to maximize the learning potential and match the prerequisite of step 4 for anomaly interpretation, there are several requirements in selecting an anomaly detection model listed as follows.

- The anomaly detection model uses unsupervised learning.
- The anomaly detection model produces some type of error score for each input data point $x_{t,m}$. Reconstructive models produce reconstruction error, and regressive models produce prediction error. These are some example models that fulfill this requirement. The model error score is used for the anomaly interpretation method in step 4 to find relevant sensors for a predicted anomaly.
- The anomaly detection model handles multivariate datasets, as we intend to predict and interpret anomalies time-wise instead of sensor-wise.

In this research, we examine the efficacy of the proposed pipeline using an LSTM Variational Autoencoder model, a reconstruction-based model for time series datasets. The model outputs log reconstruction likelihood for each input data point, and these are the errors used for anomaly interpretation. The model is described in detail in section 4.2

3.5. Step 4 (Postprocessing): Provide Anomaly Interpretation Based on Reconstruction Error

For each input sample $\mathbf{x_t} = (x_{t,1}, ..., x_{t,M})$, where t is the timestamp and M is the number of sensors in the dataset, an anomaly detection model satisfying requirements listed in section 3.4 would output error scores $\mathbf{S}_t = (S_{t,1}, ..., S_{t,M})$. We determine that the sensors that are responsible for a detected anomaly are the ones with either the highest or the lowest error scores, depending on the score type. For example, if the error score is reconstruction likelihood, then the responsible sensors are the ones with the lowest scores. On the other hand, if the error score is reconstruction error, then the responsible sensors are those with the highest scores. For the anomaly detection model that we use in this research, the former case applies. Therefore, we define the interpretation for a predicted anomaly as any sensor that has an error score smaller than a threshold τ. We define two choices of τ depending on the use case:

$$\tau = \begin{cases} median(\mathbf{S}_t) - z \cdot \sigma(\mathbf{S}_t), z \in \mathbf{N}, & \text{if we want to minimize False Positive (Type I Error)} \\ median(\mathbf{S}_t), & \text{if we want to minimize False Negative (Type II Error)} \end{cases} \tag{2}$$

Two choices are available because in industrial scenarios, technicians might be overwhelmed by the error warnings of machines themselves, and precision is more important to save manpower, then the first choice of τ should be used. In other cases where it is more important to find all anomalous sensors, then the second choice of τ should be used.

Both choices of τ are easily interpretable. For the first choice, if $\mathbf{S}_t \sim \mathcal{N}(\mu, \sigma)$, then $median(\mathbf{S}_t) = \mu(\mathbf{S}_t)$, and we can control the number of sensors to be identified for anomaly interpretation by controlling z. Note that $median$ is used instead of μ because when the distribution of $\mathbf{S}_t$ is heavily right-skewed, it could happen that no sensor's anomaly score is under the threshold $\tau = \mu(\mathbf{S}_t) - 1 \cdot \sigma(\mathbf{S}_t)$. For the second choice, a q quantile threshold means that exactly $\lfloor q * M \rfloor$ sensors are chosen to be examined, and in the case of median, $q = 0.5$.

4. Result Evaluation Strategy

This section describes in detail the datasets and methods used for evaluating the performance of the proposed pipeline.

4.1. Engineering Labeled Unstructured Datasets For Evaluation

To evaluate the performance of proposed pipeline with quantitative metrics, labeled datasets are needed. Since HM data from Thermo Fisher is unlabeled, an external dataset that provides anomaly labels is used for evaluation. To choose a suitable external dataset, two conditions need to be satisfied:

1) The anomaly label should be a pair (t, a), where t is the time at which the anomaly occurred and a is an attribute, e.g. the sensor at which the anomaly occurred. This enables the evaluation of both anomaly detection accuracy (from time-wise labels) and interpretation accuracy (from attribute-wise labels).
2) The dataset should closely mimic the characteristics of Thermo Fisher's sensor data, i.e. a multidimensional time series, where each attribute is a univariate time series and its logging time intervals are not only different from the time intervals of other attributes but also varies within itself.

Although this type of unstructured time series is commonly produced in real-world scenarios, it is not readily available in machine learning dataset reservoir. Therefore, we engineered a labeled unstructured multidimensional time series dataset from a labeled structured multidimensional time series dataset. Server Machine Dataset (SMD) is one of the few anomaly detection datasets that satisfy condition 1. It is a five-week-long equidistant dataset collected from an internet company, consisting of 28 machines each with 38 sensors where values are logged 1 minute apart. For ease of comparison with HM data, its time unit is assumed to be "second" in this thesis. Its average anomaly ratio is 4.16%. For each step of the proposed pipeline, we run the experiment 5 times on every machine dataset to reduce aleatoric uncertainty. Therefore, 8 out of the 28 machine datasets are selected for result evaluation due to the limitation of time and computational resources. In total, the pipeline is run $8 \times 5 = 40$ times to obtain the results shown in the following sections.

To transform the original SMD into unstructured data satisfying condition 2, the equidistant data is first segmented into unevenly-spaced data by inserting NaN at random timestamps and sensors, where the number of NaN inserted at each location is determined by logging pattern of Thermo Fisher data as shown in figure 8, such that the resulting unevenly-spaced SMD data mimics Thermo Fisher data logging pattern. An example of such transformation of two sensors from machine-1-6 is shown in figure 9. After transformation, the average sparsity of the engineered SMD is 99.63%. We insert NaN rather than removing values from sensors because removing values would disrupt the continuity of sensor value patterns, which would misguide the machine learning models to learn erratic behaviors for sensor values. However, this is not the case in real-life data like HM. A sensor in HM logs a value whenever there is a change in value; therefore, despite inconsistent logging intervals, sensor values are still continuous.

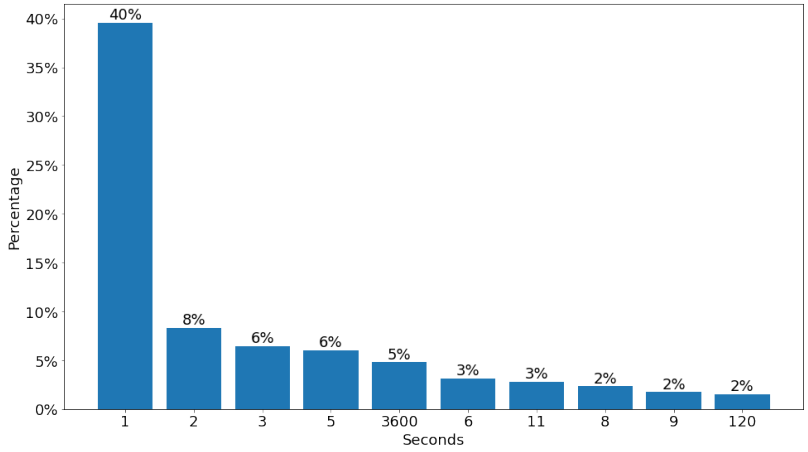


Figure 8: Occurrence frequency of top 10 most frequent logging intervals in a HM dataset.

The insertion of NaN changes anomaly labels, as seen in figure 9. To decide how NaN affects anomaly labels, we examine the original label files. The original labels show that a timestamp is marked as an anomaly if there is at least one sensor being anomalous at this timestamp. Therefore, we use the same strategy to mark anomalies after inserting NaN. This is also the reason why the number of anomalies increases after NaN insertion, as the anomalous sensors are more spread out throughout time.

In addition, the window aggregation step of the pipeline also affects anomaly labels. As a window of values are aggregated into one single value, a label is needed for the aggregated value. We determine that an aggregated value is only anomalous if at least half of the values it contains are anomalous. This idea is illustrated in figure 10.

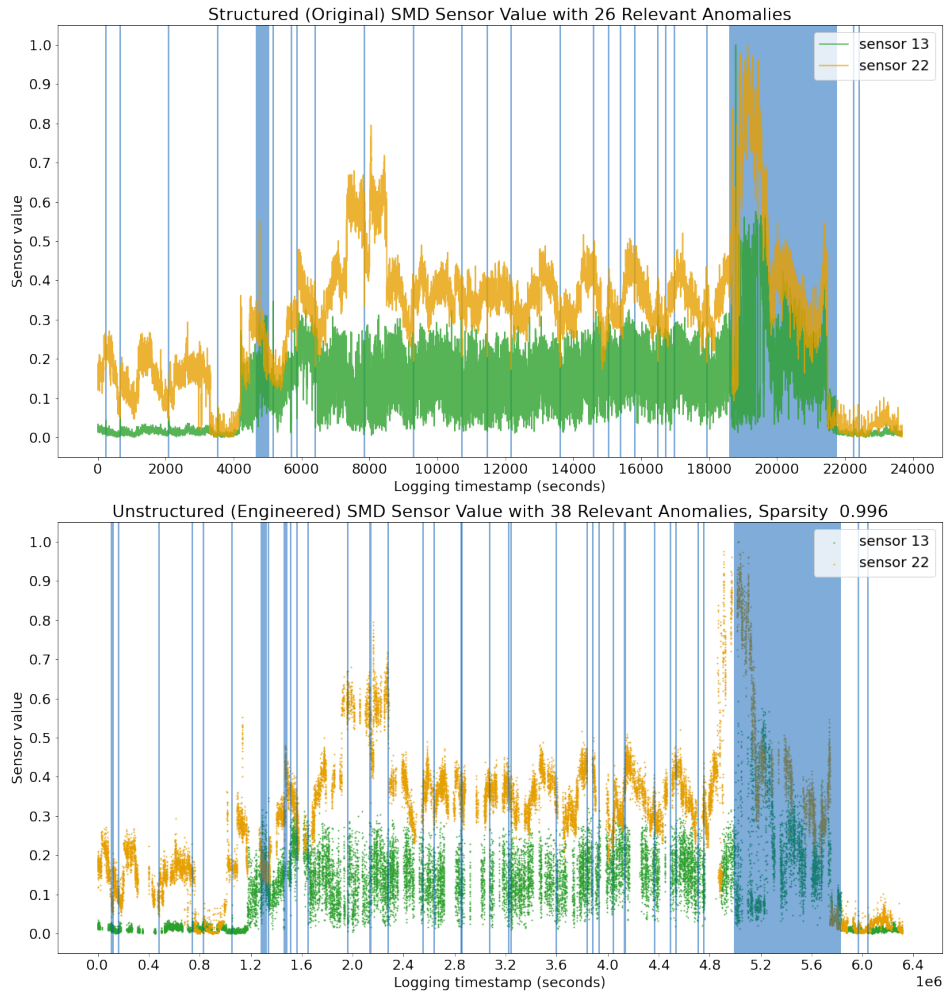


Figure 9: Comparison between the values of two sensors (13 and 22) in original structured SMD (top) and in engineered unstructured SMD (bottom). The labeled anomalous segments are highlighted in blue.

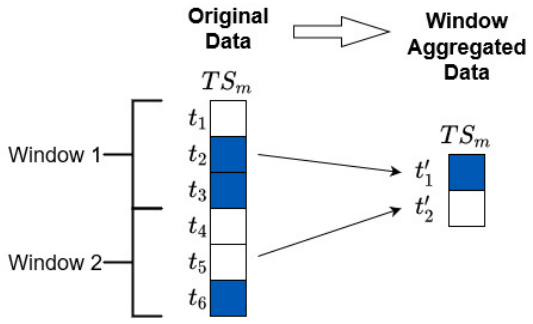


Figure 10: Suppose we aggregate the values of a sensor in the original data with a span of 3 values per window. The data points labeled as anomalies are colored in blue. Window 1 has two anomaly labels in the original data; thus, it is aggregated to a value that is labeled as an anomaly, as more than half of the values in this window are anomalies.

4.2. Anomaly Detection Model Used for Evaluation

A Stochastic Recurrent Neural Network anomaly detection model named OmniAnomaly, proposed in [12], is used for evaluating the performance of the proposed pipeline. This model is used because it belongs to the family of reconstructive

LSTM models, which are well suited for time series data as discussed in section 2.1, and the model also achieves high performance for anomaly detection tasks, according to the original authors. OmniAnomaly uses a Variational Autoencoder (VAE), composed of an encoder $q_\phi(\mathbf{z}_t|\mathbf{x}_t)$ and a decoder $p_\theta(\mathbf{x}_t|\mathbf{z}_t)$, which is used to learn the latent variable $\mathbf{z}_t$ and reconstruct the input sequence $\mathbf{x}_t$. Gated Recurrent Units (GRU) are used to capture temporal relations of the input time series. The model does not limit the distribution of the latent variable to be Gaussian; instead, Planar Normalizing Flow (Planar NF) is used to learn a non-Gaussian posterior density $q_\phi(\mathbf{z}_t|\mathbf{x}_t)$ by passing $\mathbf{z}_t$ to a chain of invertible functions $\mathbf{g}$. During inference, for each input sample $\mathbf{x}_t$ at time t, the encoder produces $\{\mu_{z_t}, \delta_{z_t}\}$ to obtain parametric latent variable $\mathbf{z}_t^0 = \mu_{z_t} + \xi_t \cdot \delta_{z_t}$ where $\xi_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The final latent variable $\mathbf{z}_t$ is obtained by transforming $\mathbf{z}_t^0$ into a non-parametric variable through Planar NF $\mathbf{z}_t = g^K(g^{K-1}(...g^1(\mathbf{z}_t^0)))$. The decoder then uses $\mathbf{z}_t$ to generate $\{\mu_{x_t}, \delta_{x_t}\}$. Finally, the reconstructed sequence $\hat{\mathbf{x}}_t$ is sampled from $\mathcal{N}(\mu_{x_t}, \delta_{x_t}^2\mathbf{I})$. The anomaly score for an input sample at time t is calculated as its reconstruction log-likelihood $\mathbf{S}(\mathbf{x}_t) = log(p_\theta(\mathbf{x}_t|\mathbf{z}_{t-T:t}))$, where T is the length of the input sequence and $p_\theta(\mathbf{x}_t|\mathbf{z}_{t-T:t}) \sim \mathcal{N}(\mu_{x_t}, \sigma_{x_t}^2\mathbf{I})$. If $\mathbf{S}(\mathbf{x}_t) < \tau$ for a certain threshold τ, then input $\mathbf{x}_t$ is marked as an anomaly, where τ is calculated from the Peak-Over-Threshold method over training data. Figure 11 illustrates the anomaly detection process of OmniAnomaly from receiving input sequence to identifying anomalies.

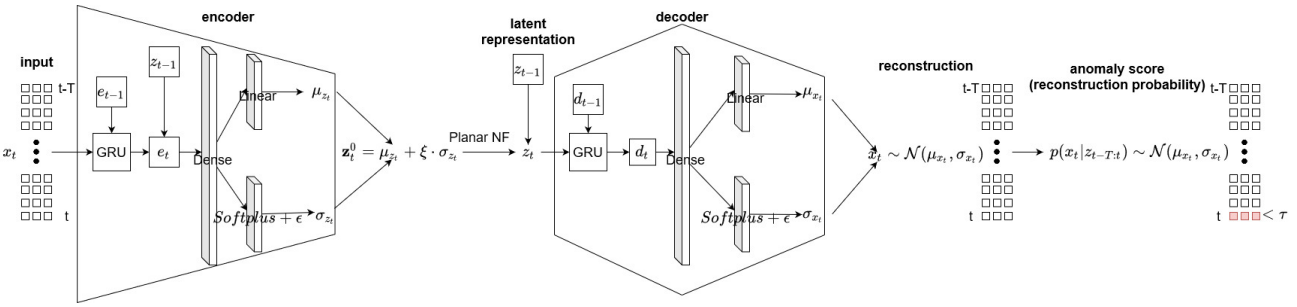


Figure 11: Structure of OmniAnomaly. At time t, the input $\mathbf{x}_{[t-T,t]}$ is a multidimensional time series from time $t-T$ to t. Hidden states e_t in encoder and d_t in decoder are concatenated with the latent representation of previous timestamp $\mathbf{z}_{t-1}$ to capture temporal information of $\mathbf{x}_{[t-T,t]}$. The identified anomalies are marked in red.

4.3. Evaluation Metrics

Since the proposed pipeline not only enables the detection of anomalies from unstructured data but also provides sensor interpretation for the detected anomalies, two metric sets are used for evaluating its performance. Metric Set 1: precision, recall and F1 for time-wise anomaly detection accuracy. Metric Set 2: precision, recall and F1 for sensor-wise interpretation.

$$precision = \frac{TP}{TP + FP} \tag{3}$$

$$recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5}$$

For Metric Set 1, there are several ways to define a True Positive (TP) for an anomaly detector. In this thesis, we define a TP to be a predicted anomalous sequence that overlaps with any ground truth anomaly along with the non-overlapping part of the ground truth anomaly, as shown in figure 12. This definition is chosen because it is the most commonly used definition in published papers, and it is also practical in industrial scenarios - signaling the occurrence of an anomaly anytime during the anomalous event is sufficient to bring technicians' attention to the time period for further examination.

For Metric Set 2, depending on the use case, the company might value precision over recall when the technicians are working under time-limited conditions; or vice versa when it is more valuable to present the technicians all possible malfunctioning sensors for inspection. Therefore, all three metrics, precision, recall, and F1 are calculated. An illustration of computing metric 2 for a prediction is shown in figure 13

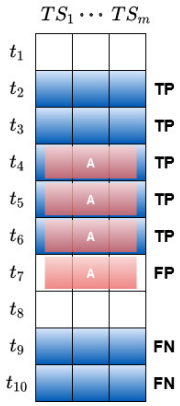


Figure 12: Example of TP, FP, FN for Metric Set 1. For a time series sequence with m sensors and 10 time steps, suppose there are two ground truth anomalous sequences (blue), and the model predicts one anomaly sequence (red). The predicted sequence between t_4 to t_6 overlaps with a label; therefore, the overlapping part t_4 to t_6 along with the non-overlapping part of the label t_2 to t_3 are considered to be correct predictions (TPs). t_7 is outside of the ground truth sequence; therefore, it is a FP.

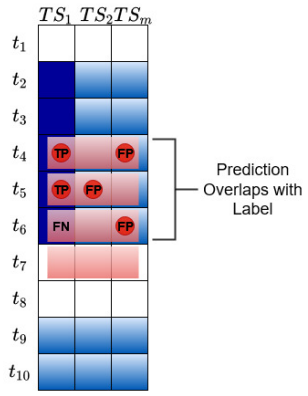


Figure 13: Example of TP, FP and FN for Metric Set 2. For each predicted anomaly, only the part that overlaps with a ground truth is considered for interpretation because the notion of interpreting anomaly is meaningless either when the model fails to detect the anomaly or when there is actually no anomaly. In this figure, interpretation labels are dark blue, predicted interpretations are represented with red circles, and the overlapping part where interpretation is computed is from t_4 to t_6.

5. Results: Window Aggregation with Triplet Loss Representation Learning

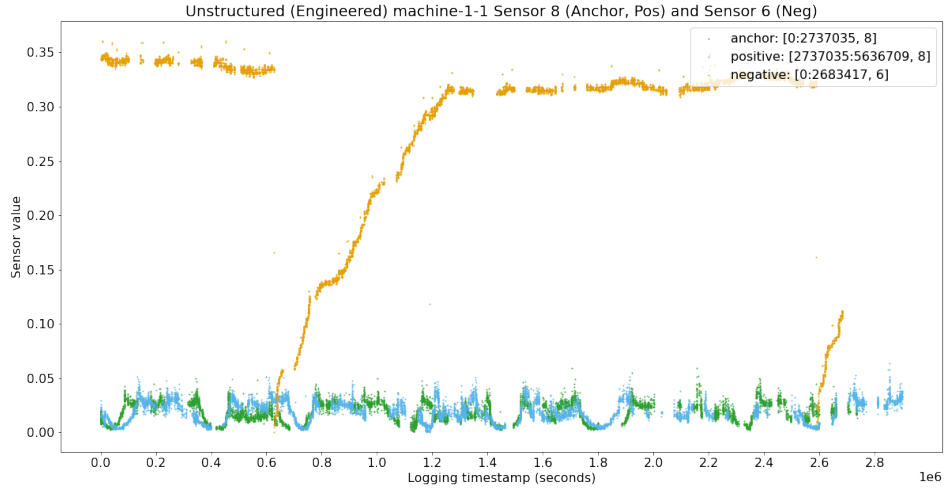
This section is a visual the evaluation of window aggregation step, including visualizations of the triplet loss embedding. Quantitative evaluation of window aggregation is performed together with the result of anomaly detection in section 6, as window aggregation itself does not produce quantitative results.

5.1. Sparsity Before VS. After Preprocessing Framework

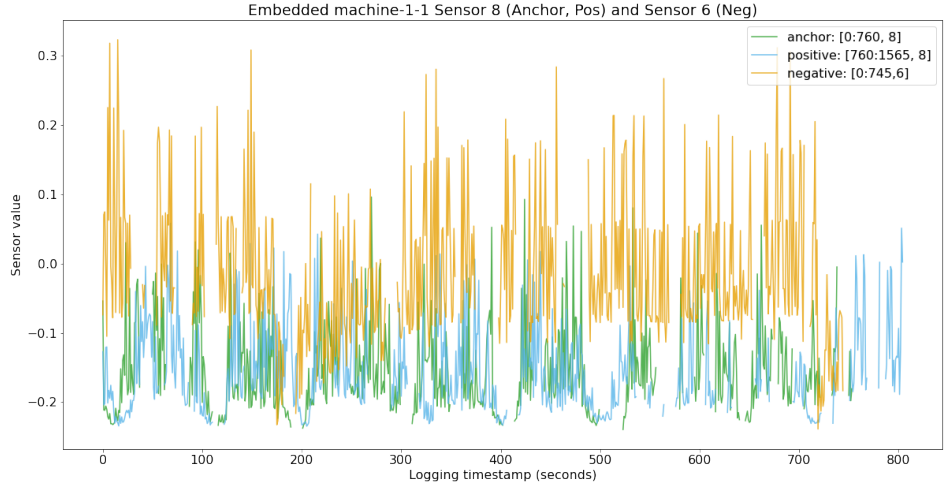
Before window aggregation, due to the fact that all sensors are logged at different timestamps and the logging frequency of each sensor varies from time to time, if the data is directly imported as any type of tabular format, a 99.63% sparse dataset would be created. Steps 1 and 2 of the proposed pipeline aim to tackle this problem. After performing step 1, the dataset becomes equidistant, and sparsity is reduced to 25.76%; after step 2, it is fully filled. As a consequence of window aggregation, dataset length is also reduced significantly – the original SMD has an average of $24,296$ rows, and after window aggregation the number is reduced to 1802.

5.2. Anchor, Positive, Negative Before VS. After Embedding

Since window aggregation is achieved using triplet loss embedding, graphs of anchor, positive, and negative samples can provide insights on how the embedding works. Figure 14 shows an example of how the encoder embeds different patterns of sensor values. The green and blue signals are the anchor and positive samples respectively, and they are values from the same sensor with the same length but from different time periods. The yellow signal is a negative sample, which consists of values from a different sensor. Figure 14a shows these values in the unstructured data space, and figure 14b shows the embedded representation of these values.



(a) Unstructured sensor values as input to encoder.



(b) The encoder outputs embedded sensor values.

Figure 14: Anchor, positive sample, and negative sample in the unstructured data space V.S. embedding space.

Table 1 calculates the Euclidean Distance and similarity between the anchor and positive sample in the embedding space, and the anchor and negative sample in the embedding space. The Euclidean distance between two time series $\mathbf{a}, \mathbf{b}$ is computed as $||\mathbf{a}, \mathbf{b}|| = \sqrt{(\mathbf{a} - \mathbf{b})^2}$, and their similarity is computed as $sim(\mathbf{a}, \mathbf{b}) = \sigma(\mathbf{x} \cdot \mathbf{x})$, where σ is the sigmoid function to rescale similarity values between 0 and 1. Since the anchor and the positive sample is closer than the anchor and the negative sample in this example, and it is also the case for most other pairs of sensors we have sampled, we can expect that the triplet loss model is able to distinguish patterns generated from different sensors.

	Euclidean Distance	Similarity [0,1]
Anchor, Positive	1.242	0.986
Anchor, Negative	2.431	0.660

TABLE 1: Euclidean distance and similarity between anchor and positive sample, and anchor and negative sample.

We can also feed the encoder different synthetic time series as experiments to further analyze how it responds to different time series patterns. Several experiments are performed to test how values and logging times influence their representations, and the result is shown in figure 15. Since each time series is encoded into a latent representation of 1-Dimension, the output of the embedded time series can be visualized on a number line. Each small figure in figure 15 is an experimental input to the encoder, and its output is pointed out on the number line by an arrow. The small figures above the number line

are different patterns of evenly spaced time series, while the small figures below the number line are the same time series but unevenly spaced. As can be seen from the figure, two time series of the same values will output embeddings that are close to each other; however, because they have different logging time intervals, their embeddings are not identical.

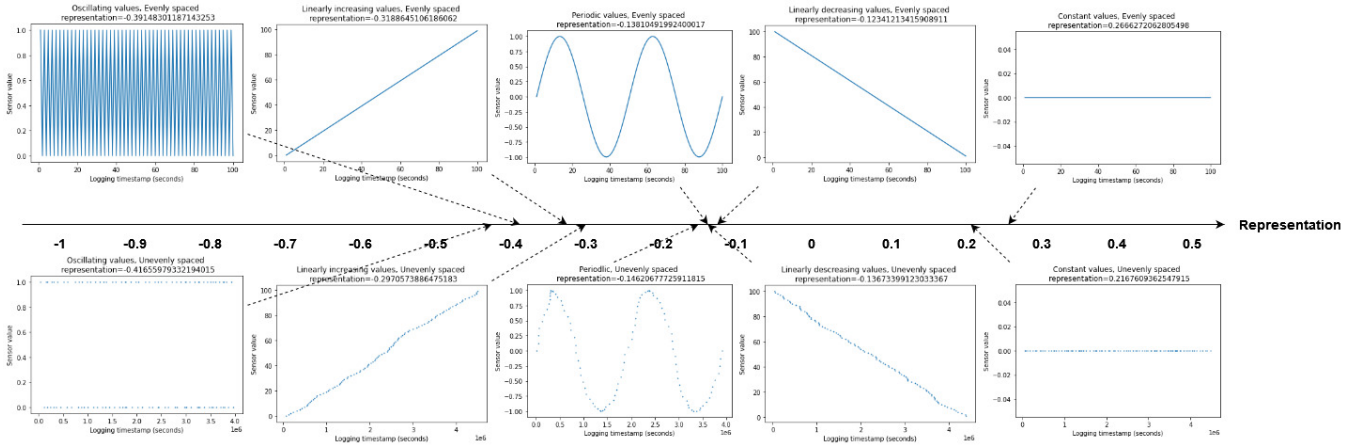


Figure 15: Experimental input time series patterns and their latent representations.

5.3. Sensor Signal Passing Through Preprocessing Steps

Visualizing the dataset before and after the preprocessing steps might also be interesting in terms of understanding how the pipeline transform the dataset. Figure 16 shows the evolution of one sensor's data (sensor 2 of machine-1-6) passing through the various steps of the preprocessing framework.

The top figure is the original structured signal; it is equidistant and contains no missing value. The second figure is the same signal after being engineered into unstructured data by us. At this point, it is 99.63% sparse, which can be inferred by the large scale of the x-axis. The third figure is the same signal after window aggregation with triplet loss embedding. This signal is upside down because of the loss function used in training the encoder. As stated in section 3.2, the triplet loss creates a 1-D metric space that assimilates $f(\mathbf{x}^{ref}, \theta)$ and $f(\mathbf{x}^{pos}, \theta)$ while separating $f(\mathbf{x}^{ref}, \theta)$ and $f(\mathbf{x}^{neg}, \theta)$. However, there is no incentive in the loss function to minimize the distance between $\mathbf{x}$ and $f(\mathbf{x}, \theta)$. Therefore, it is possible that the model finds a metric space where the condition of the loss function is satisfied but the representation is not close to the input. A simple example of this explanation is shown in figure 17. The dissimilarity between the values of $\mathbf{x}$ and $f(\mathbf{x}, \theta)$ is not a problem for anomaly detection, as detecting anomalies is about finding patterns of the whole data stream instead of measuring the exact values of individual data points. If we were to visualize the flipped encoded signal, i.e. $-f(\mathbf{x}, \theta)$, as shown in the fourth figure of figure 16, we can see that the window aggregated signal mimics the pattern of the original signal while enjoying the benefits of significantly reduced sparsity (25.4%), shorter data length, and evenly-spaced time series data. Lastly, the remaining missing values are imputed by step 2 of the pipeline with the forward filling technique, and the last figure shows its result. The obtained data is now ready to be fed into an anomaly detector of choice.

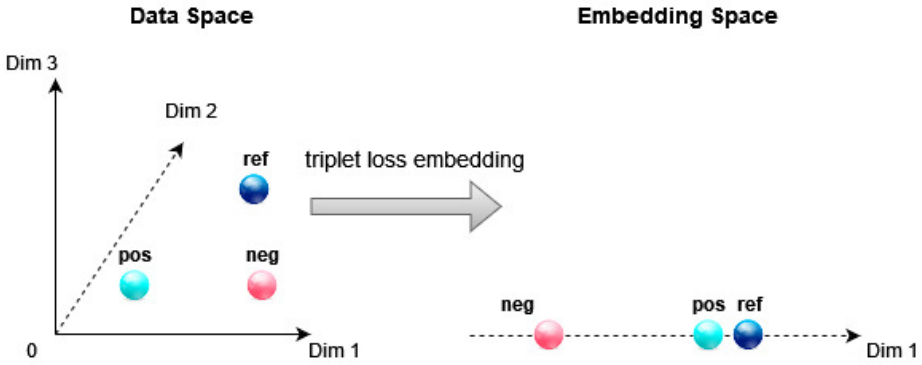


Figure 17: An example of values in original space represented as different values in embedding space while satisfying the objective of the triplet loss function.

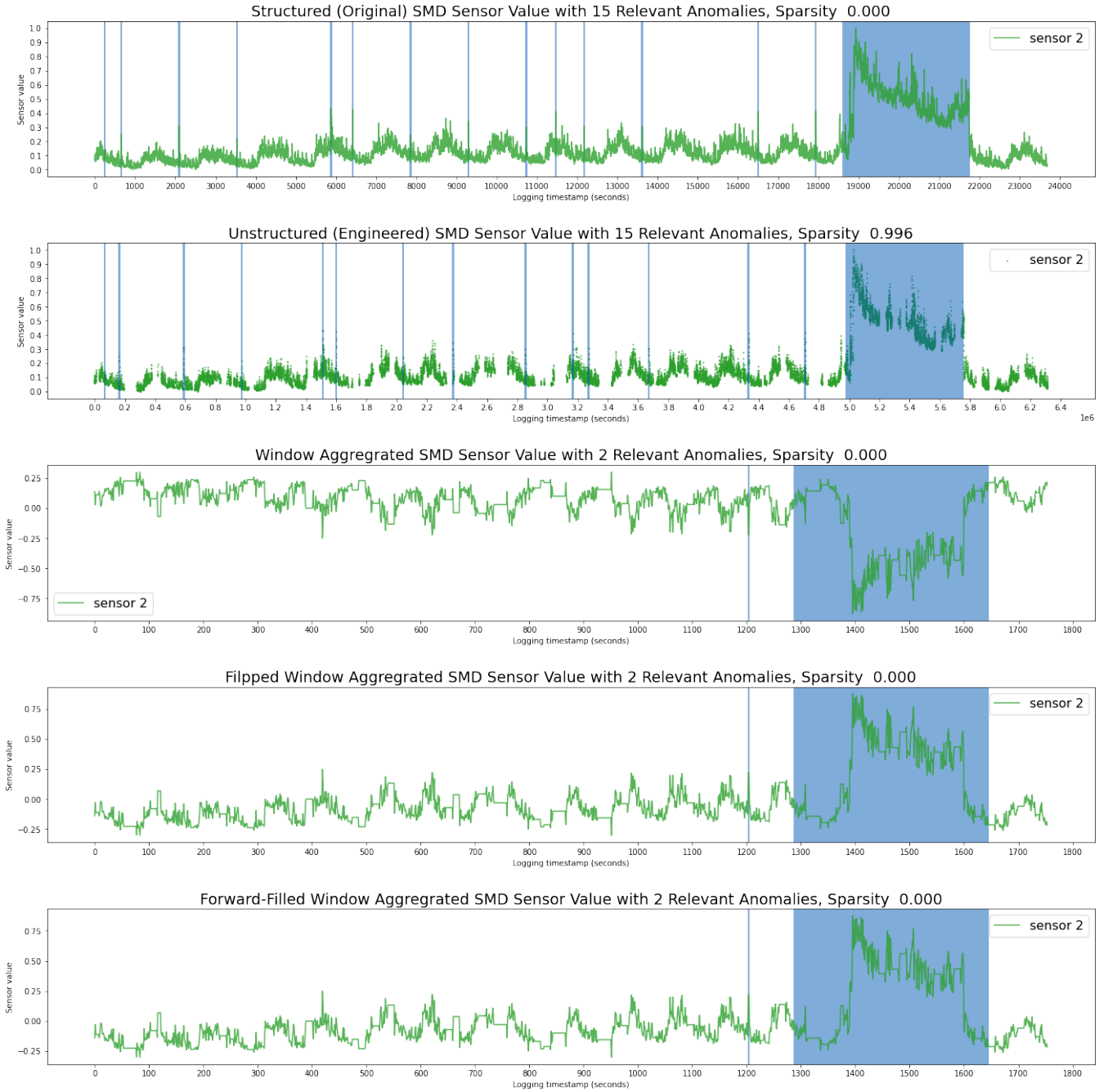


Figure 16: Evolution of Sensor 2 (machine-1-6) passing through steps 1 and 2 of the proposed framework. Anomaly labels are highlighted in blue. Labels are affected by the evolution of data, and this is discussed in section 4.1

6. Results: Anomaly Detection

This section analyzes the performance of the proposed pipeline using OmniAnomaly, an anomaly detector introduced in section 4.2. The preprocessed datasets, produced by applying the proposed pipeline on engineered unstructured SMD, are fed into OmniAnomaly, and the two metric sets discussed in section 4.3 are gathered for quantitative evaluation of the pipeline. In addition, for comparison, the same metrics are collected from the performance of OmniAnomaly applied to the original structured SMD datasets. As stated in section 4.1, datasets of 8 machines are used for evaluation, and the proposed pipeline is run 5 times for each machine to estimate the aleatoric uncertainty of anomaly predictions.

6.1. Metric Set 1 Performance - Anomaly Detection

The median anomaly detection performance of all machines and all runs is shown in table 2. Medians, instead of means, of the metric results are used because median is less affected by outlier results. The anomaly detection performance of the original SMD is also shown in the table, serving as the optimal result for this dataset and this particular choice of anomaly detector. The idea scenario is that the preprocessing framework causes little information loss, and the result of preprocessed data is close to the result of original data. In addition, the variance of each metric among 40 runs is also computed and shown in table 3, where the small variances indicate that the results are consistent between runs.

As can be seen from the table, compared to the original structured data, precision of the preprocessed unstructured data is slighter higher than the original dataset, while recall is much lower. This suggests that preprocessing framework might have a smoothing effect on the dataset, such that after window aggregation, despite the learning ability of triplet loss embedding, anomalies are less prominent, but the ones that still stand out after preprocessing are more likely to be true anomalies.

	Precision	Recall	F1
Preprocessed data	0.93	0.35	0.45
Original data	0.90	1.00	0.94

TABLE 2: Anomaly detection performance for original structured data and preprocessed unstructured data.

	Precision Variance	Recall Variance	F1 Variance
Preprocessed data	0.10	0.11	0.08
Original data	0.04	0.00	0.02

TABLE 3: Variance of each metric among 40 runs for original structured data and preprocessed unstructured data.

A more detailed comparison between original and preprocessed datasets, showing precision, recall, and F1 for each individual experiment is displayed in figure 22 of Appendix. It is evident from the figure that anomaly detection performance is dependent on the characteristics of the dataset – the performance variance of runs for each machine is generally small. Although, the preprocessing framework have the tendency to increase performance variances, especially for recall.

6.2. Metric Set 2 Performance - Anomaly Interpretation

Once the anomalies are obtained, they further go through the postprocessing framework, where the sensors that are responsible for each anomaly are obtained. Tables 4 and 5 summarize the interpretation performance of the original dataset and preprocessed dataset for the two choices of τ respectively. Opposite to the result of Metric Set 1, here, the pipeline has a more significant negative effect on precision while having a less negative influence on recall. As can be seen from table 4, when $z = 1$ for threshold calculation, precision is 0.54 for original dataset. This means that: assuming the anomaly scores are normally distributed, when a maximum of 15% of the sensors are selected for interpretation, 54% of the true anomalous sensors can be correctly identified. However, the interpretation precision for preprocessed data is much lower, which indicates that the preprocessing framework negatively impacts the interpretability of anomaly scores. For table 5, recall is above 0.5 for both original dataset and preprocessed dataset. This indicates that if the pipeline was deployed in real-life scenarios, it can identify more than half of the problematic sensors and inform engineers to examine them if an anomaly occurs. Details of Metric Set 2 is shown in figure 23 and figure 24 in Appendix, where the result for each experiment is displayed.

	Precision	Recall	F1
Preprocessed data	0.08	0.20	0.12
Original data	0.54	0.21	0.29

TABLE 4: Interpretation performance for $\tau = median(\mathbf{S}_t) - 1 \cdot \sigma(\mathbf{S}_t)$, for original structured data and preprocessed unstructured data.

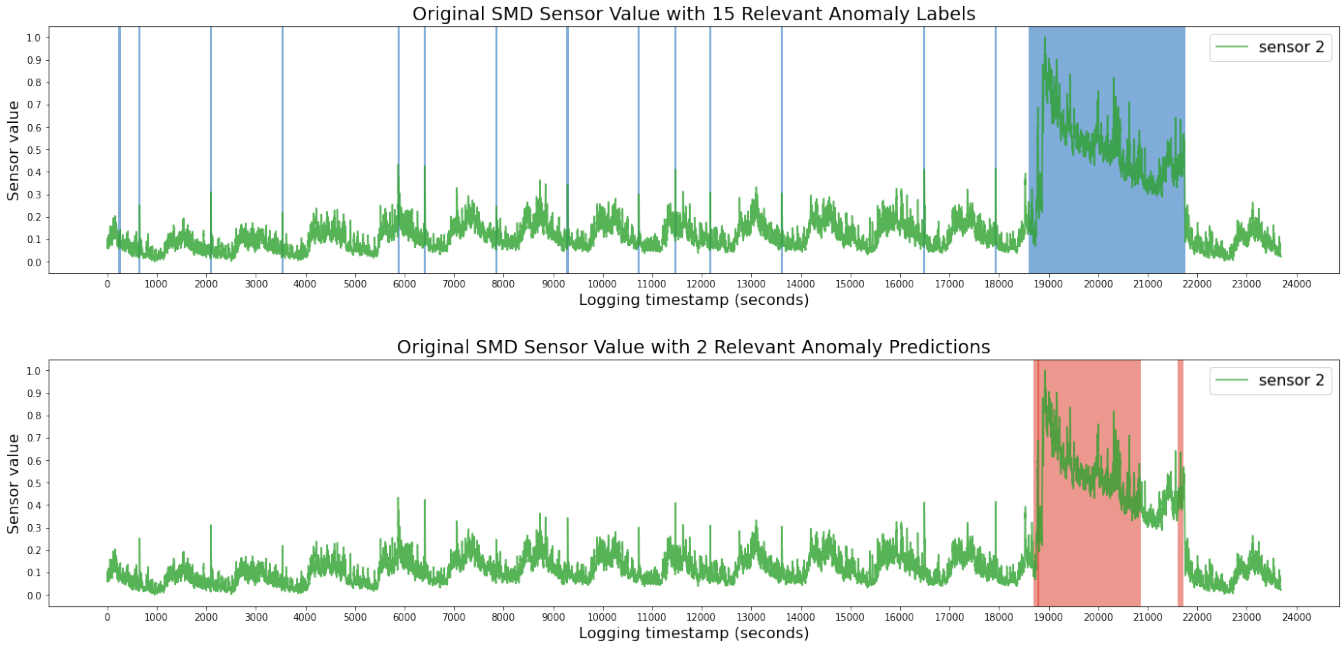
	Precision	Recall	F1
Preprocessed data	0.05	0.58	0.10
Original data	0.33	0.71	0.44

TABLE 5: Interpretation performance for $\tau = median(\mathbf{S}_t)$, for original structured data and preprocessed unstructured data.

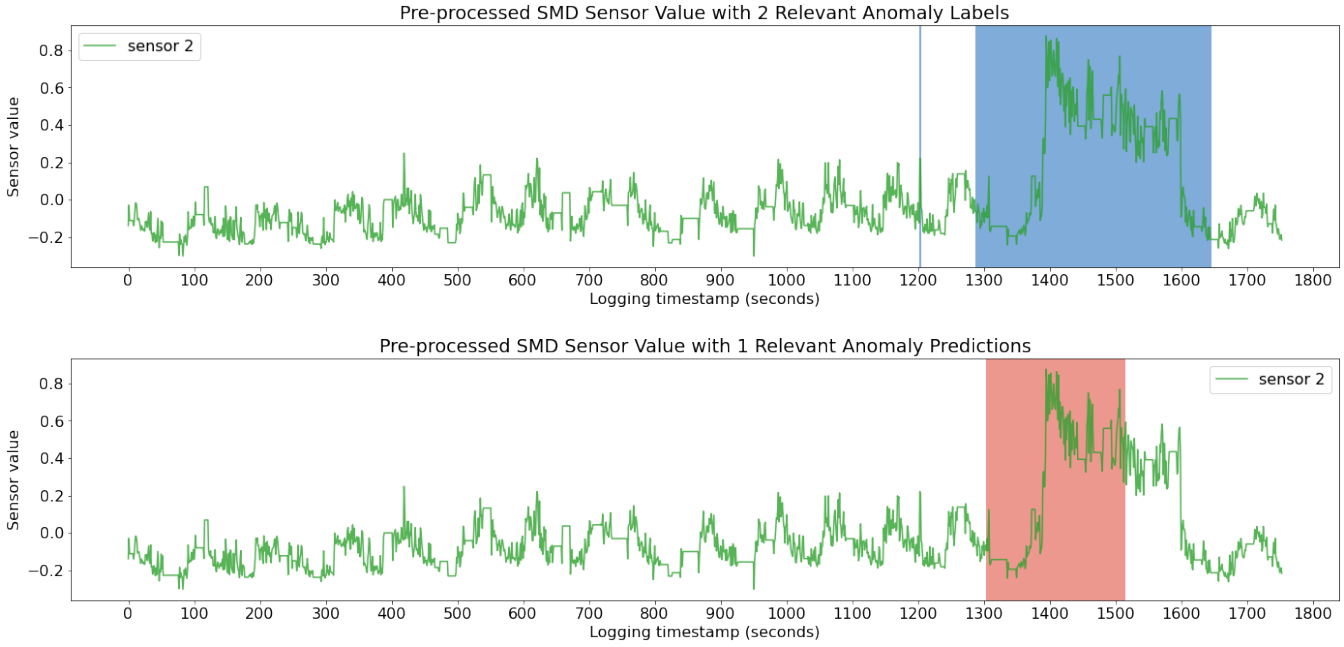
6.3. Visualization of Anomalies on Relevant Sensors

Given a dataset, it might be interesting to visualize the detected anomalies when the dataset is well structured versus when the dataset is unstructured and has to be processed by the proposed pipeline. For the same sensor example shown previously in figure 16, we can compare the anomalies being detected on this sensor before and after the proposed framework. The top

figure in figure 18a is original sensor 2 with its 15 anomaly labels highlighted in blue, and the lower figure in figure 18a shows the same sensor but highlights the two predicted anomalies in red. Figure 18b is the same comparison, but the sensor value is processed with the pipeline. After processing, it has only two anomaly labels because a window of a sensor is only labeled anomalous when more than half of the values in this window are anomalous. The effect of pipeline on anomaly labels is discussed in section 4.1. One anomaly is detected on the processed data highlighted in red.



(a) Original SMD sensor label (blue) VS predict (red).



(b) Preprocessed SMD sensor label (blue) VS predict (red)

Figure 18: Labels and predictions for original sensor and preprocessed sensor. Precision, recall, and F1 of anomaly prediction for the original sensor are 1.00, 0.97, and 0.99 respectively. Precision, recall, and F1 of this sensor after preprocessing are 1.00, 0.99, 1.00 respectively.

7. Results: Thermo Fisher Use Case

This section presents findings of applying the pipeline to a Thermo Fisher HM dataset. Since there are 215 sensors contained in this dataset, only a few sensors are selected as examples for result visualization. A complete list of detected anomalies and their responsible sensors is included in table 7 in Appendix.

7.1. Additional Processing for Thermo Fisher Use Case

There are some unique challenges associated with the datasets coming directly from Thermo Fisher's electron microscopes that are not of concern for datasets prepared for machine learning that are available online. This section addresses these challenges.

HM datasets come in XML format, each with large file size. The data of one system for one year amounts to 4 gigabytes of disk space and does not fit into memory for direct inspection and manipulation. Thus, we divide the dataset of one system into six parts and each stores the data of a unique set of sensors. For each part, the XML dataset is read into python numpy arrays containing time and value information, where time is converted from local time to UTC time to eliminate duplicated timestamps resulting from the daylight saving practice.

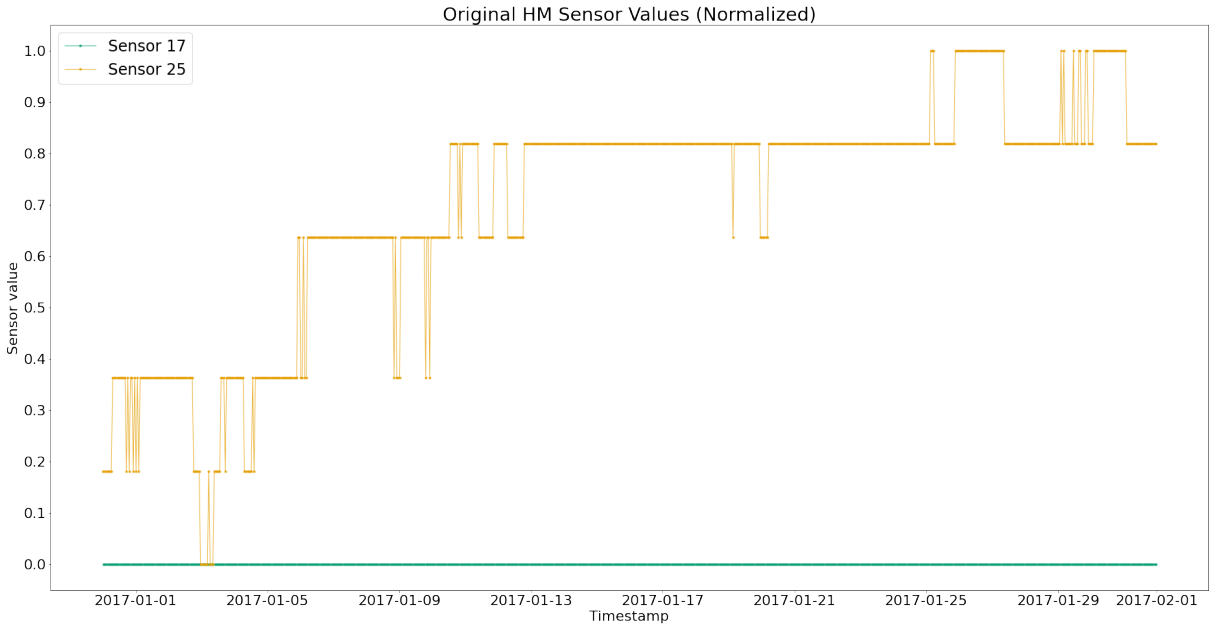
For the reasons mentioned above, during step 1 of the pipeline, one embedding is trained for each part. And such segregation of embedding training has some implications for the resulting HM data representation. Triplet loss is a metric learning technique where different patterns of sequences are matched to different values in the learned latent space. This principle works well when all data are trained with one triplet loss embedding. However, when several embeddings are trained for different parts of a dataset, a pattern could be mapped to different latent representations in different embeddings, and thus risking the latent representations to lose meaning. This problem could be avoided by using a computer with higher RAM to fit the whole dataset at once so that there is no need to divide the dataset into parts in the first place.

The dataset provided spans from 31/12/2015 to 30/03/2018. There are 208 sensors active in 2016, 231 sensors active in 2017, and 238 sensors active in 2018. Since the data for 2018 is recorded only until March, and there are more sensors active in 2017 than in 2016, we select data during the first month of 2017, i.e. 2017-01-01 to 2017-02-01, to be used as a prototype for pipeline deployment.

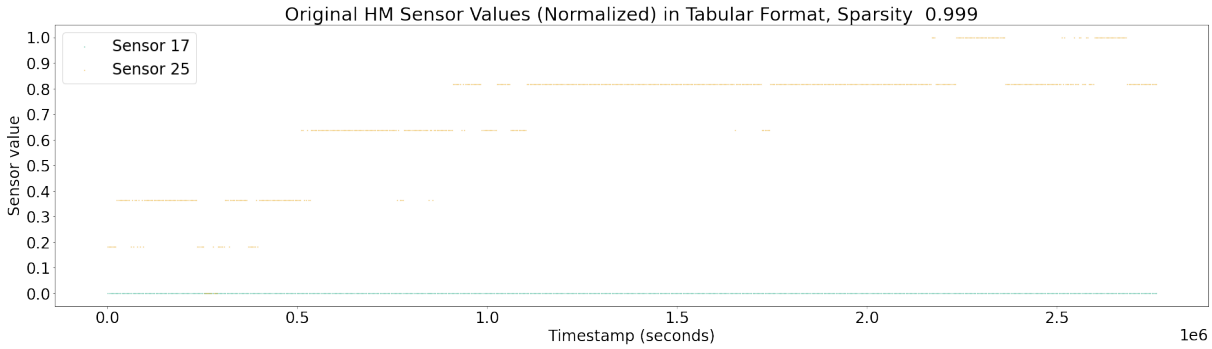
7.2. Visualization of Preprocessed HM Data

To visualize how the preprocessing framework modifies the unstructured dataset, we take two sensors from HM dataset as an example. Figure 19 shows the evolution of these sensors going through the preprocessing steps of the pipeline. Sensor 17 represents ('Optics electronics', 'Operational states', 'Optics Stigmator 3Fold Operational State') and sensor 25 represents ('GUN (XFEG1)', 'Emission', 'Emission Current'). In 19b, the values are dense because they are stored in XML format, which is unstructured and does not require timestamp alignment. Once the values are read into any tabular format, sparsity would be created. Figure 19b shows the dramatic increase in sparsity. The number of rows increased to 2.7 million after aligning timestamps of data points in these two sensors, which leads to 0.99 sparsity.

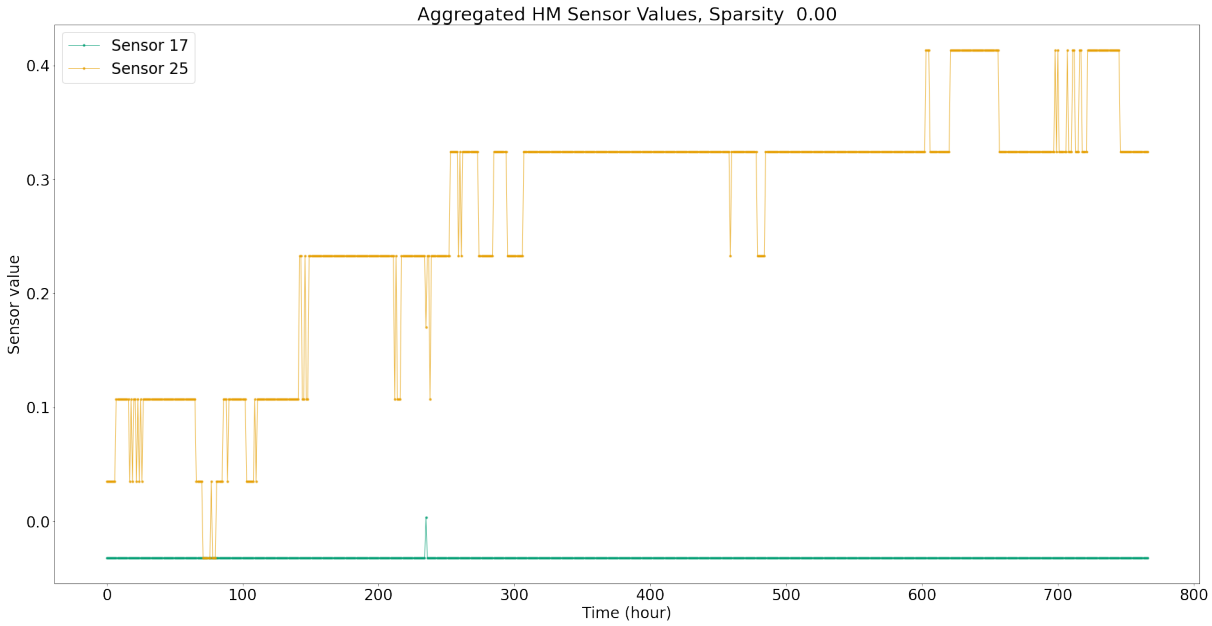
Figure 19c shows the output of preprocessed framework on these two sensors. By this point, sparsity is already eliminated, and it is an evenly-spaced and structured time series dataset, ready for machine learning models. As can be seen, figure 19c closely resembles figure 19a while eliminating the unstructured characteristic of original data.



(a) Original HM sensors in XML format.



(b) Original HM sensors in tabular format - the data type received by machine learning models.



(c) HM sensors preprocessed by the proposed pipeline.

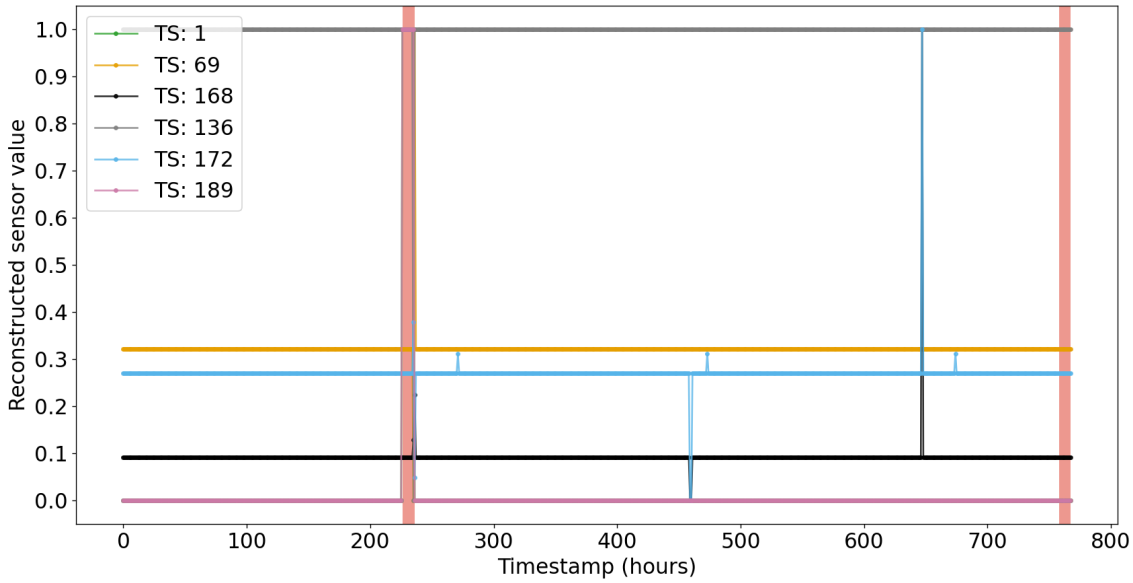
Figure 19: The evolution of sensors 17 and 25 from HM dataset going through the proposed preprocessing framework.

7.3. Visualization of Detected Anomalies

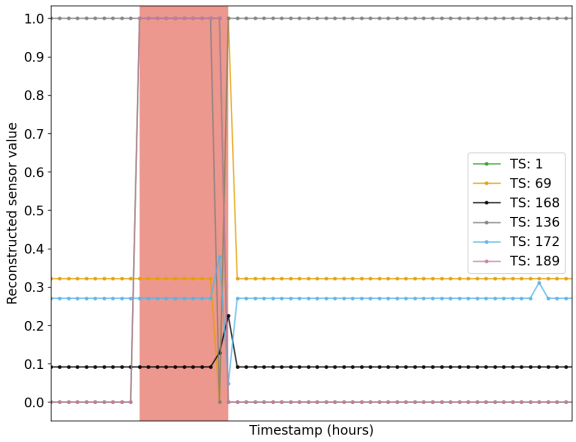
In step 3 of the proposed pipeline, the preprocessed data is fed to OmniAnomaly for anomaly detection, and then interpretation of the anomalies are obtained in step 4. This section shows these results for HM data.

Two anomalies are detected, and they are highlighted in figure 20. In reality, there are 115 responsible sensors for these two anomalies if we use the first definition of τ in equation 2, and 200 responsible sensors if we use the second definition of equation 2. From communications with Thermo Fisher, we learned that it is more desirable for the predictions to be correct than exhaustive because the electron microscopes are already producing many error warnings; thus, the first definition of τ is suitable in this case. For a better visibility, only the most anomalous sensor values are shown in the graph.

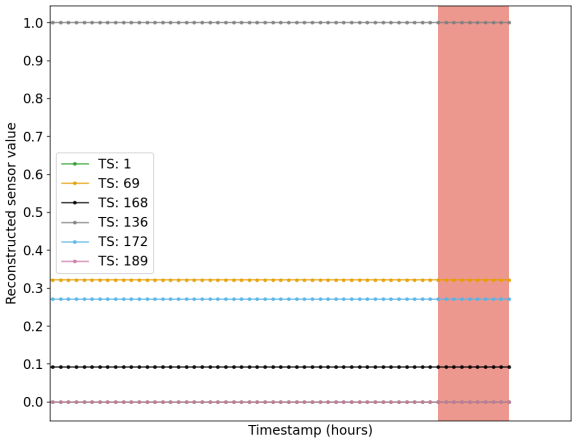
Figure 20a is an overall view of relevant sensors and anomalies. Figures 20b and 20c are zoomed-in view of the two anomalies respectively. As can be seen from the figures, the first anomaly can be well observed by the sudden change of sensor values, but the second anomaly cannot be explained by the sensor values alone. This is because indicators are used in step 2 of the proposed pipeline, and thus enabling the anomaly detector to also consider sensor logging patterns in addition to sensor values. Therefore, an irregular change of either sensor value or sensor timestamp pattern would lead to a positive anomaly prediction. This suggests that the second anomaly might have an irregular logging pattern during this time. This hypothesis is verified in figure 21. Figure 21a shows the indicator columns for interpretation sensors. If there is no data logged at a certain timestamp, then its corresponding indicator equals 1; else, its indicator equals 0. Figure 21b zooms in to the indicators during the occurrence of the second anomaly. The blue line on the bottom is a mixture of sensors 69, 168, 136 and 172, and the periodic purple line is a mixture of sensors 1 and 189. It is evident from figure 21b that the logging pattern changes during the second anomaly. For the sensors represented by the blue line, normally, there are values at each hour; however, during the second anomalous time period, there is no value logged for these sensors, and this might be an indication of sensor malfunctions. For the sensors represented by the purple line, they periodically log data points; however, during the second anomaly, the logging interval is longer than its normal cycle. Concluding from above, the pipeline not only enables anomaly detection on real-world unstructured data, but also allows engineers to examine the cause of anomalies from both time perspective and sensor value perspective.



(a) Detected anomalies (highlighted in red) on HM data. Interpretation based on $TS_{Value} + TS_{Indicators}$.

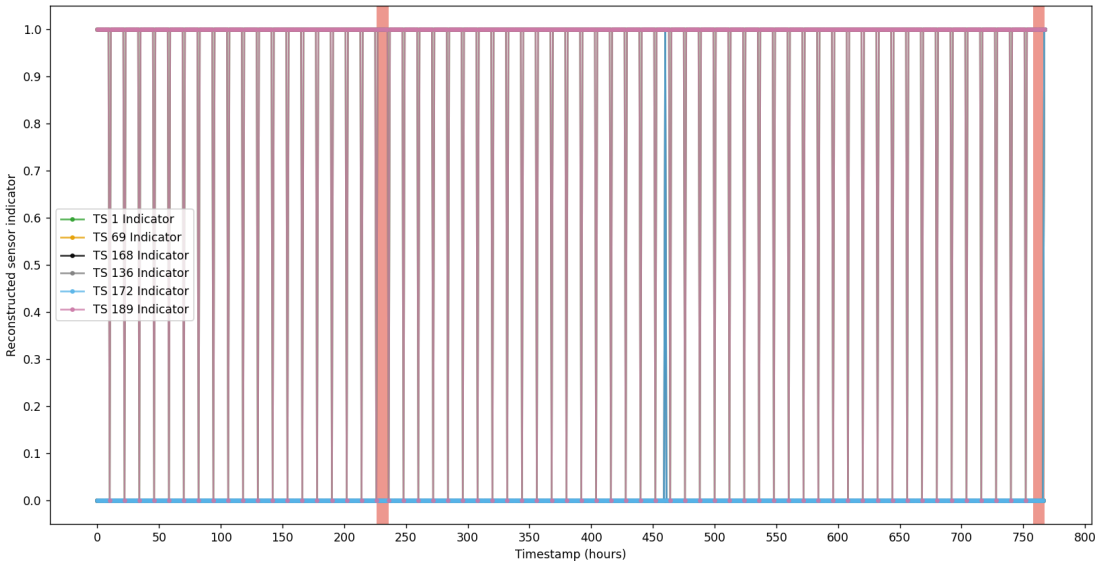


(b) Zoom-in view of first anomaly

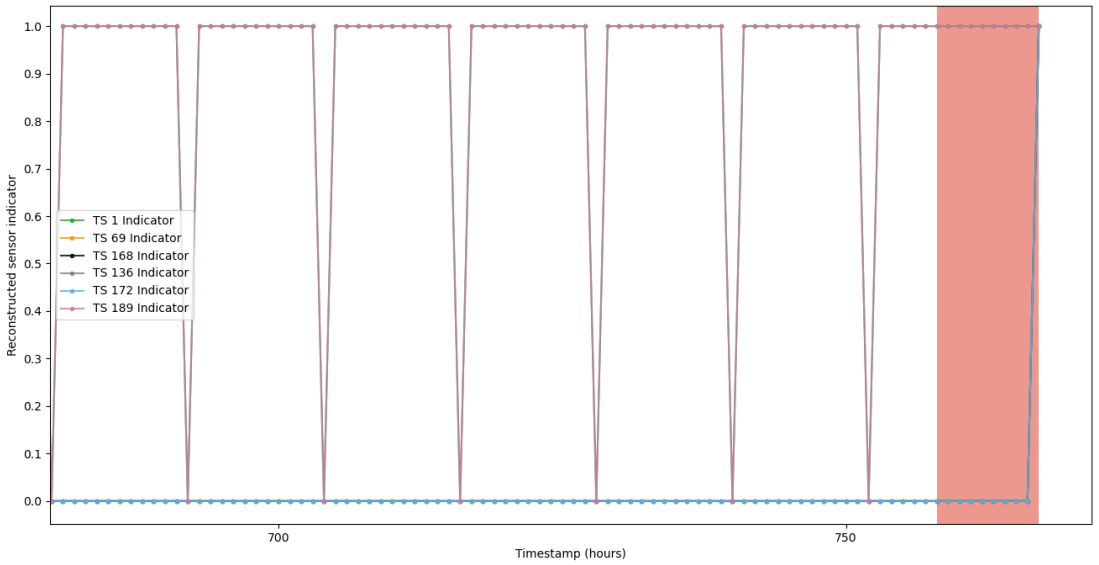


(c) Zoom-in view of second anomaly.

Figure 20



(a) Indicator columns for relevant sensors.



(b) Zoom-in view of indicator columns for the second anomaly.

Figure 21

7.4. Verify Detected Anomalies with Company's Call Data and Thermo Fisher Experts

The two anomalies detected translates to 158 different incidents in original HM XML data, which are listed in table 7. The reason is that the two anomalies are detected on the window aggregated dataset, and each time point in the window aggregated dataset represents a time span of 1 hour. In the case of HM dataset, since most sensors record at a frequency of less than 1 hour, each predicted anomalous data point on the window aggregated dataset could include up to 3600 values. And they are further grouped into consecutive time periods, thus resulting 158 different incidents.

To verify the detected anomalies, we ask Thermo Fisher for documents that could serve as an indication of when anomalous events might have happened. Customer call data that documents customer complaints is the best available record for such a purpose. This record is a CSV, containing the opening and closing date of the calls, phrases to describe the calls, and the technician's written notes on the actions taken and speculation of problems.

Since call data is unstructured text data, we can only manually compare its entries with our predictions. As can be seen from table 7, the predicted anomalies occur on dates 2017-01-08, 2017-01-09, 2017-01-31, and 2017-02-01. Scanning call records, there are two calls around these dates that might be linked to the predicted anomalies. For example, on dates 2017-01-19 and 2017-02-01, a customer called about a reoccurring "TEM distortion" issue. According to the technician's notes, "Objective Stigamator", "Objective Aperture", and "Gun" are problematic and might be causing to the issue. Comparing to our predicted anomalies, sensors involving "Stigamator" are marked anomalous on dates 2017-01-09, 2017-01-31, and 2017-02-01 (rows 56 59 and 107 108 in table 7); sensors involving "Aperture" are marked anomalous on dates 2017-01-09 and 2017-01-31 (rows 1 2 and 156 157 in table 7); sensors involving "Gun" are also marked anomalous on 2017-01-08, 2017-01-09, 2017-01-31, and 2017-02-01 (rows 84 86 and 115 136 in table 7). In fact, many of the sensors responsible for the first predicted anomaly show that there was a system restart or power failure on 2017-01-09, as critical sensors like "voltage" plumbed to 0. Judging from our anomaly detection result, we suspect that the anomaly first happened on 2017-01-08, and customer called on 2017-01-19 to complain about the issue. It was fixed by the technicians, but the anomalies happened again on 2017-01-31 and 2017-02-01, and the customer called again on 2017-02-01. These are the only calls occurring between 2017-01-01 and 2017-02-1.

These speculations are discussed with Thermo Fisher experts. They comment that the sensor behaviors occurring on the dates of the detected anomalies were indeed abnormal. However, it is not certain whether they are linked to the calls. Linking detected anomalies to call data is proposed as future work of this project.

8. Discussion

8.1. Provide Solutions to the Research Questions

The proposed pipeline solves two prevalent problems in the realm of anomaly detection. Firstly, since it is able to transform unstructured time series datasets to structured datasets while retaining information important for anomaly detection, it opens up a wide range of possibilities for numerous machine learning models to be deployed in real-world scenarios where sensors generate data at random timestamps; thus, answering the research questions asked in section 1.2. Secondly, the pipeline expands on the capability of unsupervised LSTM anomaly detection models such that both value pattern and logging time pattern of sensors are considered to be factors of an anomaly, whereas traditional methods of processing unstructured time series often cause the latter information to be lost.

8.2. Advantages

Despite satisfying its intended purpose of allowing anomaly detection on unstructured sensor data, there are a few additional advantages of transforming the data with the pipeline.

8.2.1. Encourage High Anomaly Detection Precision

As noted in table 2, the precision is higher for processed unstructured SMD than that of the original structured SMD. Although it is only a small increase, since the result is obtained after running the pipeline 40 times on 8 different datasets; thus, it is a somewhat reliable indication that the pipeline does not negatively affect anomaly detection precision.

One of the characteristics of the proposed pipeline that leads to this advantage is the use of window aggregation, which smooths the extreme values in a time series in addition to making raw data structured. The extreme values after window aggregation mean that the original values were even more extreme, thus more likely to be true anomalies. Even though the method used for window aggregation is not simple statistics like mean, mode and etc., the triplet loss embedding retains this characteristic nonetheless.

Another characteristic of the proposed pipeline is that it utilizes triplet loss embedding to capture the combined information of both value dimension and time dimension while staying similar enough to the original signal. For example, in figure 19a, the original signal of sensor 17 is a constant value throughout time. However, its embedded signal is not a perfect constant line, which is observed in figure 19c – there is a small turbulence at hour 235. This is because that at this point, its corresponding original signal is logged slightly earlier than its usual logging interval, i.e. sensor 17 is logged every 3600 seconds, except at this point where it is logged 2062 seconds after the previous value is logged. This subtle irregularity would be ignored by traditional processing methods, like mean aggregation, but captured and retained by triplet

loss embedding. Despite such sensitivity to the time dimension of the dataset, the overall pattern of the time series is retained in the embedding. As shown in figure 15, the representations of the same input values are close to each other regardless of their logging time interval differences. This sensitive yet nuanced representation trains a more robust anomaly detection model, which can detect abnormal behaviors more accurately.

8.2.2. Allow Versatile Anomaly Definitions to Suit Industry Need

One feature of the proposed pipeline is that the time information is preserved, thus enabling higher flexibility for anomaly definition. This characteristic of the pipeline enables the exploration of the following questions:

- Which sensors are recording abnormal values?
- Which sensors are recording values at an abnormal frequency?
- Which sensors are malfunctioning considering both of the above?

The anomaly detection results presented so far are based on the combined information from both the values and indicators, i.e. the third question, and under such definition, the anomaly score of a data at timestamp t from sensor m is defined as $S(\mathbf{x}_{t,m}) = S(V_m(\mathbf{x}_{t,m})) + S(I_m(\mathbf{x}_{t,m}))$ where V_m stands for the values of sensor m and I_m stands for the indicators of sensor m. However, this is not the only way to detect and interpret anomalies. For example, if in a situation where only the sensor values are important while logging time pattern is irrelevant, then $S(\mathbf{x}_{t,m}) = S(V_m(\mathbf{x}_{t,m}))$. Following this definition of anomaly score, we get a new set of predictions.

With this new set of predictions based on value only, we obtained its performance for Metric Set 1, and it is shown in table 6. Comparing to table 2, the precision of preprocessed data declined from 0.93 to 0.89, an evidence that the utilization of time information, i.e. indicator columns, in the proposed pipeline promotes higher anomaly detection precision. Empirical results show that the new anomaly predictions have little effect on Metric Set 2, so it is not discussed in this section again.

	Precision	Recall	F1
Preprocessed data	0.89	0.41	0.48
Original data	0.90	1.00	0.94

TABLE 6: Anomaly detection performance. For preprocessed dataset, only sensor value is taken into account while sensor indicator is ignored. For original structured data, since it is not processed by the pipeline, there is no change to its results.

In the scenario where it is of interest to find sensors that are logged at anomalous frequencies regardless of their values, a similar approach can be applied to obtain predictions.

8.2.3. Combine CNN and RNN

The proposed pipeline combines a Convolutional Neural Network (CNN) with a Recurrent Neural Network (RNN), where the triplet loss embedding learns features with a variant of CNN, and these learned features are passed into an LSTM anomaly detection model, a variant of RNN. The combination of CNN and RNN is a standard model for sequence classification [8], as CNN is strong in learning localized patterns while RNN is adept at learning long term patterns.

8.3. Drawbacks

8.3.1. Unsatisfying Anomaly Interpretation Performance

A noticeable problem of the proposed pipeline is that the performance of Metric Set 2 is inadequate. Interpretation precision is very low for processed data, and the recall is not high enough for the interpretation method to be deployed in industries. Because although the pipeline can inform engineers of the precise time when an anomaly has occurred as the precision for Metric Set 1 is high, it would still leave the engineers guessing which sensors are responsible. There are two possible reasons to explain the causes of the problem.

Firstly, the dimension of the triplet loss latent representation might be too low to capture both value patterns and logging interval patterns of sequences; thus, introducing ambiguity and noise to the processed data. Figure 15 illustrates such an example. The representations of the periodic sequence and linearly decreasing sequence are concentrated between $(-0.2, -0.1)$, while in fact, these are distinct patterns that ideally should have more separated latent representations. Therefore, this might have contributed to the poor performance of Metric Set 2. As the 1D latent representation become over-saturated with information, noise is introduced to the processed dataset. As a result, the embedded data losses its direct interpretable meaning. This weakness of the pipeline might be able alleviated by increasing latent representation dimension; however, such a solution would complicate the structure of the processed dataset.

Secondly, to generate anomaly interpretation sensors for a detected anomaly $\mathbf{x}_t$, selecting a suitable threshold is crucial for the success of correct interpretation. So the problem becomes: for a list of anomaly scores $S(\mathbf{x}_t) = (S(x_{t,1}), ..., S(x_{t,M}))$ among all M sensors, how to find a threshold τ such that it maximizes the distance between the scores lower than τ and the scores higher than τ. Threshold selection is a mini anomaly detection problem. In this work, we experiment with two different τ and obtained vastly different results – one achieved relatively high precision (for original structured dataset) but lower recall, and the other produced low precision but high recall. This indicates that anomaly scores contain valuable information for anomaly interpretation, but extracting this information might require a more sophisticated approach, as our use of median and standard deviation is a rather naive approach.

8.3.2. Difficulty in Visualizing the Complete Latent Space of Triplet Loss Embedding

When a latent space is trained with supervised methods, the resulting embedding will show nice data arrangement in the latent space – data points from the same label category will be close to each other, and thus it is possible to visualize the whole latent space by querying all labels. However, since our triplet loss embedding is not trained with labels, but rather with random sampling of time series patterns, it is very difficult for us to visualize a complete picture of the latent space, because we cannot generate all possible time series patterns existing in our dataset. But, we can still generate a few distinguished examples as in figure 15 to visualize the data arrangement of the triplet loss latent space.

9. Future Work

The research question was proposed because Thermo Fisher wish to use the predicted anomalies and their interpretations to match with the company's call data, such that when a customer calls about an issue, corresponding anomalous sensors can be quickly identified. Although we have done some preliminary manual inspections to match predicted anomalous sensors with calls, a machine learning model could do this work more efficiently. Due to time limitations, such a model is not developed in this research; however, it could be an interesting topic for future work.

10. Appendix

Figure 22: Anomaly detection performance: precision, recall, F1 box plots for 8 machines' preprocessed (green) and original (pink) datasets, each with 5 runs.

Figure 23: Interpretation performance ($\tau = median(\mathbf{S}_t) - 1 \cdot \sigma(\mathbf{S}_t)$): precision, recall, F1 box plots for each machine's preprocessed (green) and original (pink) dataset.

Figure 24: Interpretation performance ($\tau = median(\mathbf{S}_t)$): precision, recall, F1 box plots for each machine's preprocessed (green) and original (pink) dataset.

	parameter_name	anomaly_start(local)	anomaly_end(local)
0	('Software', 'Server', 'Server Watchdog')	2017-01-09 12:00:30	2017-01-09 21:58:52
1	('Apertures', 'C1', 'Diameter of Inserted Aperture')	2017-01-09 11:51:34	2017-01-09 11:51:34
2	('Motion AvaC2 Aperture', 'C2Y - axis', 'Axis position after move stopped')	2017-01-09 10:01:19	2017-01-09 20:23:22
3	('Edx', 'Detectors', 'Leakage Current Detector 1')	2017-01-09 11:37:23	2017-01-09 21:11:36
4	('Edx', 'Detectors', 'Leakage Current Detector 2')	2017-01-09 11:37:23	2017-01-09 21:11:36
5	('Edx', 'Detectors', 'Leakage Current Detector 3')	2017-01-09 11:37:23	2017-01-09 21:11:36
6	('Edx', 'Detectors', 'Leakage Current Detector 4')	2017-01-09 11:37:23	2017-01-09 21:11:36
7	('Edx', 'Detectors', 'Temperature Detector 2')	2017-01-09 09:27:56	2017-01-09 21:10:53
8	('Edx', 'Detectors', 'Temperature Detector 3')	2017-01-09 09:19:43	2017-01-09 21:10:55
9	('Edx', 'Detectors', 'Temperature Detector 4')	2017-01-09 11:57:19	2017-01-09 21:10:57
10	('Edx', 'Heater', 'Edx Defrost Cycles')	2017-01-09 12:50:23	2017-01-09 21:11:36
11	('Edx', 'Heater', 'Edx Heater Voltage 2')	2017-01-09 14:10:36	2017-01-09 21:10:53
12	('Edx', 'Heater', 'Edx Heater Voltage 3')	2017-01-09 11:53:46	2017-01-09 21:10:55
13	('Edx', 'Heater', 'Edx Heater Voltage 4')	2017-01-09 11:34:06	2017-01-09 21:10:57
14	('Edx', 'OverdoseLimiter', 'Overdose Limiter Disable Count')	2017-01-09 11:36:30	2017-01-09 11:36:30
15	('Edx', 'OverdoseLimiter', 'Overdose Limiter Disable Time')	2017-01-09 11:36:30	2017-01-09 11:36:30
16	('Edx', 'Shutter', 'Edx Shutter Errors')	2017-01-09 12:50:23	2017-01-09 21:11:36
17	('Optics electronics', 'Firmware states', 'Beam Deflector AC Firmware State')	2017-02-01 00:11:08	2017-02-01 00:11:08
18	('Optics electronics', 'Firmware states', 'Beam Deflector AC Firmware State')	2017-01-31 15:11:08	2017-01-31 23:11:08
19	('Optics electronics', 'Firmware states', 'Image Deflector AC Firmware State')	2017-02-01 00:11:08	2017-02-01 00:11:08
20	('Optics electronics', 'Firmware states', 'Image Deflector AC Firmware State')	2017-01-31 15:11:08	2017-01-31 23:11:08
21	('Optics electronics', 'Firmware states', 'Beam Blanker and Shutter Firmware State')	2017-02-01 00:11:08	2017-02-01 00:11:08
22	('Optics electronics', 'Firmware states', 'Beam Blanker and Shutter Firmware State')	2017-01-31 15:11:08	2017-01-31 23:11:08
23	('Optics electronics', 'Firmware states', 'Beam Deflector X Firmware State')	2017-01-31 15:26:20	2017-02-01 00:16:37
24	('Optics electronics', 'Firmware states', 'Beam Deflector Y Firmware State')	2017-01-31 15:26:20	2017-02-01 00:16:37
25	('Optics electronics', 'Firmware states', 'Condenser Deflector X Firmware State')	2017-02-01 00:11:08	2017-02-01 00:11:08
26	('Optics electronics', 'Firmware states', 'Condenser Deflector X Firmware State')	2017-01-31 15:11:08	2017-01-31 23:11:08
27	('Optics electronics', 'Firmware states', 'Condenser Deflector Y Firmware State')	2017-02-01 00:11:09	2017-02-01 00:11:09
28	('Optics electronics', 'Firmware states', 'Condenser Deflector Y Firmware State')	2017-01-31 15:11:08	2017-01-31 23:11:09
29	('Optics electronics', 'Firmware states', 'Gun Deflector X Firmware State')	2017-02-01 00:11:09	2017-02-01 00:11:09
30	('Optics electronics', 'Firmware states', 'Gun Deflector X Firmware State')	2017-01-31 15:11:09	2017-01-31 23:11:09
31	('Optics electronics', 'Firmware states', 'Gun Deflector Y Firmware State')	2017-01-09 12:36:46	2017-01-09 21:11:08
32	('Optics electronics', 'Firmware states', 'Image Deflector X Firmware State')	2017-01-09 12:36:46	2017-01-09 21:11:08
33	('Optics electronics', 'Firmware states', 'Image Deflector Y Firmware State')	2017-01-09 12:36:46	2017-01-09 21:11:08
34	('Optics electronics', 'Firmware states', 'Fixed Ref HV Drive Firmware State')	2017-02-01 00:11:09	2017-02-01 00:11:09

	parameter_name	anomaly_start(local)	anomaly_end(local)
35	('Optics electronics', 'Firmware states', 'Fixed Ref HV Drive Firmware State')	2017-01-31 15:11:09	2017-01-31 23:11:09
36	('Optics electronics', 'Firmware states', 'C1 Lens Firmware State')	2017-02-01 00:11:09	2017-02-01 00:11:09
37	('Optics electronics', 'Firmware states', 'C1 Lens Firmware State')	2017-01-31 15:11:09	2017-01-31 23:11:09
38	('Optics electronics', 'Firmware states', 'C2 Lens Firmware State')	2017-01-31 22:24:38	2017-02-01 00:24:38
39	('Optics electronics', 'Firmware states', 'C2 Lens Firmware State')	2017-01-31 15:24:38	2017-01-31 21:24:38
40	('Optics electronics', 'Firmware states', 'C3 Lens Firmware State')	2017-01-31 22:17:34	2017-02-01 00:17:34
41	('Optics electronics', 'Firmware states', 'C3 Lens Firmware State')	2017-01-31 15:17:34	2017-01-31 21:17:34
42	('Optics electronics', 'Firmware states', 'Diffraction Lens Firmware State')	2017-01-31 23:00:56	2017-02-01 00:00:56
43	('Optics electronics', 'Firmware states', 'Diffraction Lens Firmware State')	2017-01-31 15:00:56	2017-01-31 22:00:56
44	('Optics electronics', 'Firmware states', 'Intermediate Lens Firmware State')	2017-01-09 11:36:47	2017-01-09 20:36:47
45	('Optics electronics', 'Firmware states', 'Mini Condenser Lens Firmware State')	2017-02-01 00:11:09	2017-02-01 00:11:09
46	('Optics electronics', 'Firmware states', 'Mini Condenser Lens Firmware State')	2017-01-31 15:11:09	2017-01-31 23:11:09
47	('Optics electronics', 'Firmware states', 'Objective Inner Lens Firmware State')	2017-01-09 11:36:47	2017-01-09 20:36:47
48	('Optics electronics', 'Firmware states', 'Objective Middle Lens Firmware State')	2017-02-01 00:11:09	2017-02-01 00:11:09
49	('Optics electronics', 'Firmware states', 'Objective Middle Lens Firmware State')	2017-01-31 15:11:09	2017-01-31 23:11:09
50	('Optics electronics', 'Firmware states', 'Objective Focus Outer Lens Firmware State')	2017-02-01 00:15:20	2017-02-01 00:16:09
51	('Optics electronics', 'Firmware states', 'Objective Focus Outer Lens Firmware State')	2017-01-31 15:01:39	2017-02-01 00:15:09
52	('Optics electronics', 'Firmware states', 'P1 Lens Firmware State')	2017-02-01 00:11:10	2017-02-01 00:11:10
53	('Optics electronics', 'Firmware states', 'P1 Lens Firmware State')	2017-01-31 15:11:10	2017-01-31 23:11:10
54	('Optics electronics', 'Firmware states', 'P2 Lens Firmware State')	2017-02-01 00:11:10	2017-02-01 00:11:10
55	('Optics electronics', 'Firmware states', 'P2 Lens Firmware State')	2017-01-31 15:11:10	2017-01-31 23:11:10
56	('Optics electronics', 'Firmware states', 'Optics Stigmator 2Fold Firmware State')	2017-02-01 00:11:10	2017-02-01 00:11:10
57	('Optics electronics', 'Firmware states', 'Optics Stigmator 2Fold Firmware State')	2017-01-31 15:11:10	2017-01-31 23:11:10
58	('Optics electronics', 'Firmware states', 'Optics Stigmator 3Fold Firmware State')	2017-02-01 00:11:10	2017-02-01 00:11:10
59	('Optics electronics', 'Firmware states', 'Optics Stigmator 3Fold Firmware State')	2017-01-31 15:11:10	2017-01-31 23:11:10
60	('Optics electronics', 'Lens temperatures', 'C1 channel 1 lens coil temperature')	2017-01-31 18:24:23	2017-02-01 00:24:23
61	('Optics electronics', 'Lens temperatures', 'C1 channel 1 lens coil temperature')	2017-01-31 15:24:23	2017-01-31 17:24:23
62	('Optics electronics', 'Lens temperatures', 'C1 channel 2 lens coil temperature')	2017-01-09 12:05:39	2017-01-09 21:25:07

	parameter_name	anomaly_start(local)	anomaly_end(local)
63	('Optics electronics', 'Lens temperatures', 'C3 channel 2 lens coil temperature')	2017-01-31 15:20:09	2017-02-01 00:54:18
64	('Optics electronics', 'Lens temperatures', 'Intermediate channel 1 lens coil temperature')	2017-01-31 15:42:38	2017-02-01 00:32:32
65	('Optics electronics', 'Lens temperatures', 'MiniCondenser channel 2 lens coil temperature')	2017-01-31 15:24:26	2017-02-01 00:54:27
66	('Optics electronics', 'Lens temperatures', 'ObjSubOuter channel 2 lens coil temperature')	2017-01-31 22:20:05	2017-02-01 00:20:05
67	('Optics electronics', 'Lens temperatures', 'ObjSubOuter channel 2 lens coil temperature')	2017-01-31 19:20:05	2017-01-31 21:20:05
68	('Optics electronics', 'Lens temperatures', 'P1 channel 1 lens coil temperature')	2017-01-31 15:36:20	2017-02-01 00:36:20
69	('Optics electronics', 'Lens temperatures', 'P1 channel 2 lens coil temperature')	2017-01-31 15:44:52	2017-02-01 00:44:52
70	('Optics electronics', 'Lens temperatures', 'P2 channel 2 lens coil temperature')	2017-01-31 21:32:40	2017-02-01 00:32:40
71	('Optics electronics', 'Lens temperatures', 'P2 channel 2 lens coil temperature')	2017-01-31 15:32:40	2017-01-31 20:32:40
72	('Optics electronics', 'Operational states', 'Beam Deflector AC Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
73	('Optics electronics', 'Operational states', 'Beam Deflector AC Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
74	('Optics electronics', 'Operational states', 'Image Deflector AC Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
75	('Optics electronics', 'Operational states', 'Beam Blanker and Shutter Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
76	('Optics electronics', 'Operational states', 'Beam Blanker and Shutter Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
77	('Optics electronics', 'Operational states', 'Beam Deflector X Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
78	('Optics electronics', 'Operational states', 'Beam Deflector X Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
79	('Optics electronics', 'Operational states', 'Beam Deflector Y Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
80	('Optics electronics', 'Operational states', 'Condenser Deflector X Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
81	('Optics electronics', 'Operational states', 'Condenser Deflector X Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
82	('Optics electronics', 'Operational states', 'Condenser Deflector Y Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
83	('Optics electronics', 'Operational states', 'Condenser Deflector Y Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
84	('Optics electronics', 'Operational states', 'Gun Deflector X Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
85	('Optics electronics', 'Operational states', 'Gun Deflector X Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
86	('Optics electronics', 'Operational states', 'Gun Deflector Y Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
87	('Optics electronics', 'Operational states', 'Image Deflector X Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
88	('Optics electronics', 'Operational states', 'Image Deflector Y Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
89	('Optics electronics', 'Operational states', 'Image Deflector Y Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
90	('Optics electronics', 'Operational states', 'Fixed Ref HV Drive Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31

	parameter_name	anomaly_start(local)	anomaly_end(local)
91	('Optics electronics', 'Operational states', 'C1 Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
92	('Optics electronics', 'Operational states', 'C1 Lens Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
93	('Optics electronics', 'Operational states', 'C2 Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
94	('Optics electronics', 'Operational states', 'C3 Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
95	('Optics electronics', 'Operational states', 'Diffraction Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
96	('Optics electronics', 'Operational states', 'Diffraction Lens Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
97	('Optics electronics', 'Operational states', 'Intermediate Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
98	('Optics electronics', 'Operational states', 'Intermediate Lens Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
99	('Optics electronics', 'Operational states', 'Mini Condenser Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
100	('Optics electronics', 'Operational states', 'Objective Inner Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
101	('Optics electronics', 'Operational states', 'Objective Inner Lens Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
102	('Optics electronics', 'Operational states', 'Objective Middle Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
103	('Optics electronics', 'Operational states', 'Objective Focus Outer Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
104	('Optics electronics', 'Operational states', 'P1 Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
105	('Optics electronics', 'Operational states', 'P2 Lens Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
106	('Optics electronics', 'Operational states', 'P2 Lens Operational State')	2017-01-31 15:10:53	2017-02-01 00:10:53
107	('Optics electronics', 'Operational states', 'Optics Stigmator 2Fold Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
108	('Optics electronics', 'Operational states', 'Optics Stigmator 3Fold Operational State')	2017-01-09 11:36:31	2017-01-09 20:36:31
109	('Optics electronics', 'Operational states', 'Optics Overall Operational State')	2017-01-09 12:37:16	2017-01-09 21:11:29
110	('Camera Server', 'BM_UltraScan', 'Camera availability')	2017-01-31 18:34:35	2017-02-01 00:34:35
111	('Camera Server', 'BM_UltraScan', 'Camera availability')	2017-01-31 15:04:35	2017-01-31 18:04:35
112	('Camera Server', 'FluCam', 'Camera availability')	2017-01-31 22:13:36	2017-02-01 00:43:36
113	('Camera Server', 'FluCam', 'Camera availability')	2017-01-31 15:13:36	2017-01-31 21:43:36
114	('Flu screen', 'Flu screen', 'Screen current')	2017-01-31 15:00:52	2017-02-01 00:55:52
115	('GUN (XFEG1)', 'Extractor', 'Extractor Testline Voltage')	2017-01-09 11:26:54	2017-01-09 20:26:54
116	('GUN (XFEG1)', 'Extractor', 'Extractor Testline Voltage')	2017-01-31 15:11:29	2017-02-01 00:11:29
117	('GUN (XFEG1)', 'Extractor', 'Extractor User Setting')	2017-01-31 15:34:00	2017-02-01 00:34:00
118	('GUN (XFEG1)', 'Feg State', 'Cold Start Attempts Count')	2017-01-08 23:36:30	2017-01-08 23:36:30
119	('GUN (XFEG1)', 'Feg State', 'Cold Start Count')	2017-01-08 23:36:30	2017-01-08 23:36:30
120	('GUN (XFEG1)', 'Feg State', 'Feg State')	2017-01-09 11:37:38	2017-01-09 20:37:38
121	('GUN (XFEG1)', 'Feg State', 'Standby Time Counter')	2017-01-09 11:36:30	2017-01-09 20:36:30
122	('GUN (XFEG1)', 'Feg State', 'Warm Start Count')	2017-01-09 11:37:38	2017-01-09 20:37:38
123	('GUN (XFEG1)', 'Feg State', 'Warm Start Count')	2017-01-31 15:10:53	2017-02-01 00:10:53
124	('GUN (XFEG1)', 'Filament', 'Filament Limit')	2017-01-31 15:34:00	2017-02-01 00:34:00
125	('GUN (XFEG1)', 'Filament', 'Filament Testline')	2017-01-09 11:10:49	2017-01-09 20:10:49
126	('GUN (XFEG1)', 'Filament', 'Filament Testline')	2017-01-31 15:11:12	2017-02-01 00:11:12
127	('GUN (XFEG1)', 'HT Tank', 'HT-emission')	2017-01-31 20:20:05	2017-02-01 00:20:05

	parameter_name	anomaly_start(local)	anomaly_end(local)
128	('GUN (XFEG1)', 'HT Tank', 'HT-emission')	2017-01-31 15:20:05	2017-01-31 19:20:05
129	('GUN (XFEG1)', 'HT Tank', 'HT Tank Measured Voltage')	2017-01-31 15:18:09	2017-02-01 00:18:09
130	('GUN (XFEG1)', 'HT Tank', 'HT Tank Setting Voltage')	2017-01-31 15:18:07	2017-02-01 00:18:07
131	('GUN (XFEG1)', 'HT Tank', 'HTTankState')	2017-01-31 23:14:01	2017-02-01 00:14:01
132	('GUN (XFEG1)', 'HT Tank', 'HTTankState')	2017-01-31 15:14:01	2017-01-31 22:14:01
133	('GUN (XFEG1)', 'IGPf', 'IGPf Testline Voltage')	2017-01-09 11:27:43	2017-01-09 20:27:43
134	('GUN (XFEG1)', 'IGPf', 'IGPf Testline Voltage')	2017-01-31 15:11:29	2017-02-01 00:11:29
135	('GUN (XFEG1)', 'Suppressor', 'Suppressor Testline Voltage')	2017-01-31 15:44:50	2017-02-01 00:58:14
136	('GUN (XFEG1)', 'Electronics Vessel', 'Electronics Vessel Temperature')	2017-01-09 12:53:00	2017-01-09 20:53:00
137	('CompuStage', 'CompuStage Status', 'CompuStage Status')	2017-01-09 11:37:23	2017-01-09 21:22:23
138	('Main Vacuum', 'CCGp Pressure', 'CCGp Pressure')	2017-01-31 15:09:59	2017-02-01 00:53:04
139	('Main Vacuum', 'IGPa', 'IGPa Pressure')	2017-01-09 11:37:19	2017-01-09 21:11:31
140	('Main Vacuum', 'IGPa', 'IGPa Voltage')	2017-01-09 21:10:52	2017-01-09 21:11:31
141	('Main Vacuum', 'IGPcl', 'IGPcl Pressure')	2017-01-09 11:37:19	2017-01-09 21:11:31
142	('Main Vacuum', 'IGPcl', 'IGPcl Voltage')	2017-01-31 15:28:15	2017-02-01 00:28:15
143	('Main Vacuum', 'Autonomous Cryo Cycle', 'Refill state')	2017-01-09 11:36:31	2017-01-09 20:36:31
144	('Main Vacuum', 'Autonomous Cryo Cycle', 'Refill state')	2017-01-31 15:10:53	2017-02-01 00:10:53
145	('Main Vacuum', 'PIRbf', 'Buffer pressure')	2017-01-31 15:02:59	2017-02-01 00:57:11
146	('Main Vacuum', 'PIRpv', 'Prevacuum pressure')	2017-01-31 15:39:10	2017-02-01 00:22:15
147	('Main Vacuum', 'PPcl Pressure', 'PPcl Pressure')	2017-01-31 15:47:25	2017-02-01 00:52:22
148	('Stem', 'InsertRetractTimeouts', 'Stem HAADF Insert Retract Timeouts')	2017-01-09 11:36:30	2017-01-09 11:36:30
149	('Stem', 'SynchronizationTimeouts', 'Pia Mainslock Synchronization Timeouts')	2017-01-09 11:36:30	2017-01-09 11:36:30
150	('Piezo', 'Piezo Y', 'Piezo Y Position')	2017-01-09 05:35:03	2017-01-09 21:19:28
151	('Piezo', 'Piezo Z', 'Piezo Z Position')	2017-01-09 11:37:33	2017-01-09 21:19:28
152	('SystemMonitoring', 'HDD', 'Reallocation Event Count')	2017-01-08 23:34:36	2017-01-08 23:34:36
153	('SystemMonitoring', 'HDD', 'Current Pending Sector Count')	2017-01-08 23:34:36	2017-01-08 23:34:36
154	('SystemMonitoring', 'HDD', 'Uncorrectable Sector Count')	2017-01-08 23:34:36	2017-01-08 23:34:36
155	('SystemMonitoring', 'HDD', 'Reallocated Sectors Count')	2017-01-08 23:34:36	2017-01-08 23:34:36
156	('Motion AvaC2 Aperture', 'C2X - axis', 'Error code')	2017-01-09 20:21:32	2017-01-09 21:29:32
157	('Motion AvaC2 Aperture', 'C2X - axis', 'Error code')	2017-01-31 22:37:00	2017-01-31 22:37:00

TABLE 7: Detected anomalies with their interpretations for one HM system during the first month of 2017.

References

[1] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series. 2016.

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 2009.

[3] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *33rd Conference on Neural Information Processing Systems, Neural Information Processing Systems Foundation, Dec 2019, Vancouver, Canada*, 2020.

[4] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In Aasa Feragen, Marcello Pelillo, and Marco Loog, editors, *Similarity-Based Pattern Recognition*, pages 84–92, Cham, 2015. Springer International Publishing.

[5] Dharanipragada Janakiram, Arun Kumar, and Adi Mallikarjuna Reddy V. Outlier detection in wireless sensor networks using bayesian belief networks. In *2006 1st International Conference on Communication Systems Software Middleware*, 2006.

[6] Zachary C. Lipton, David C. Kale, and Randall Wetzel. Modeling missing data in clinical time series with rnns. *Proceedings of Machine Learning for Healthcare 2016*, 2016.

[7] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *ESANN 2015 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2015.

[8] Vlado Menkovski. Models for sequential data (rnn), deep learning (2imm10 lecture notes), 2020.

[9] Vlado Menkovski. Spatially distributed data (cnn), deep learning (2imm10 lecture notes), 2020.

[10] Christoph Molnar. *Interpretable Machine Learning*. 2019.

[11] Hyun Joon Shin, Dong-Hwan Eom, and Sung-Shick Kim. One-class support vector machines – an application in machine fault detection and classification. *Computers Industrial Engineering*, 2005.

[12] 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 *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 2828–2837, 2019.

[13] Priyanga Dilini Talagala, Rob J. Hyndman, and Kate Smith Miles. Anomaly detection in high-dimensional data. *Journal of Computational and Graphical Statistics*, 2020.

[14] Srikanth Thudumu, Philip Branch, Jiong Jin, and Jugdutt (Jack) Singh. A comprehensive survey of anomaly detection techniques for high dimensional big data. 2020.

[15] Chafiq Titouna, Mackhlouf Aliouat, and Mourad Gueroui. Outlier detection approach using bayes classifiers in wireless sensor networks. 2015.

[16] Aaron Van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. 2016.