Feature Selection for Fault Detection and Prediction based on Event Log Analysis

Zhong Li
Leiden Institute of Advanced Computer Science
Leiden University
Leiden, The Netherlands
z.li@liacs.leidenuniv.nl

Matthijs van Leeuwen
Leiden Institute of Advanced Computer Science
Leiden University
Leiden, The Netherlands
m.van.leeuwen@liacs.leidenuniv.nl

ABSTRACT

Event logs are widely used for anomaly detection and prediction in complex systems. Existing log-based anomaly detection methods usually consist of four main steps: log collection, log parsing, feature extraction, and anomaly detection, wherein the feature extraction step extracts useful features for anomaly detection by counting log events. For a complex system, such as a lithography machine consisting of a large number of subsystems, its log may contain thousands of different events, resulting in abounding extracted features. However, when anomaly detection is performed at the subsystem level, analyzing all features becomes expensive and unnecessary. To mitigate this problem, we develop a feature selection method for log-based anomaly detection and prediction. Specifically, our method consists of three main modules: the *Log Event Vectorization* module that converts semi-structured log texts into time series; the *Selection of Relevant Features* module that leverages Kendall rank correlation and Granger causality test to select log events for fault detection and prediction; and the *Removal of Redundant Features* module that utilises Kendall rank correlation to reduce redundant log events. Results on 25 real-world datasets show that our method can detect and predict faults more accurately by selecting a small proportion of log events, thereby improving the effectiveness and efficiency.

1. INTRODUCTION

A lithography machine is a piece of complex structural equipment used to manufacture chips. Typically, it consists of the following main subsystems: the light source subsystem, the objective lens subsystem, the table subsystem, the mask table subsystem, the mask transfer subsystem, the wafer transfer subsystem, and the exposure subsystem [19]. Particularly, the wafer transfer subsystem serves to transfer silicon wafers between the track and wafer stage, having a great impact on the precision of chip fabrication. A wafer transfer subsystem usually contains two robots, namely a load robot and an unload robot. When the lithography machine goes into production, these two robots may encounter faults. We assume there are $L + M$ types of faults, viz. GF_1,GF_2,...,GF_L and SF_1,SF_2,...,SF_M. Specifically, GF_1,GF_2,...,GF_L represent faults that occur gradually and thus can be detected in an early stage (i.e., they are typically predictable). In contrast, SF_1,SF_2,...,SF_M denote faults that generally occur suddenly and are often hard to predict.

To minimize machine downtime and thus maximize productivity, the possible faults of load and unload robots should be detected and predicted (if possible) in an automated way. To this end, the wafer transfer subsystem usually uses sensors to measure the position of the two robots in real time, collecting time series data that can be used for data-driven fault detection and prediction. However, due to the limited information contained in sensor data, it is challenging to detect all possible faults based on time series data alone. Meanwhile, as shown in Figure 1 and Table 1, a lithography machine also has an information system that records all triggered events—in the form of logs—when the machine is working. Since the different subsystems in a lithography machine are interconnected, a fault incurred in one subsystem (e.g., the wafer transfer system) may trigger events not only in that subsystem, but also in other subsystems. Besides, the components in the same subsystem are usually closely interconnected. Therefore, the fault of one component is very likely to cause faults of other components. Therefore, the event logs contain important information for fault detection and prediction. Traditional log-based anomaly detection methods can be used to detect such faults [17].

Due to the complexity of the lithography machine, there can be thousands of unique log events, resulting in millions of log events in a relatively short working time of the machine. Moreover, when attempting to detect faults of certain components in a specific subsystem (e.g., the load and unload robots in the wafer transfer subsystem), many of these log events are irrelevant or abundant. Hence, a direct application of existing log-based anomaly detection methods on all log events can be computationally prohibitive and may also produce misleading detection results due to the inclusion of irrelevant log events. To mitigate this problem, we regard each log event as a feature and develop a feature selection method that aims to select relevant features for log-based fault detection and prediction.

In brief, our method consists of three main modules, namely *Log Event Vectorization*, *Selection of Relevant Features* and *Removal of Redundant Features*. Specifically, the *Log Event Vectorization* module aims at converting unstructured log events into time series data; the *Selection of Relevant Features* module attempts to select relevant features for fault detection and prediction by using the variables measured by sensors as target; and the *Removal of Redundant Features* module focuses on eliminating redundant features to

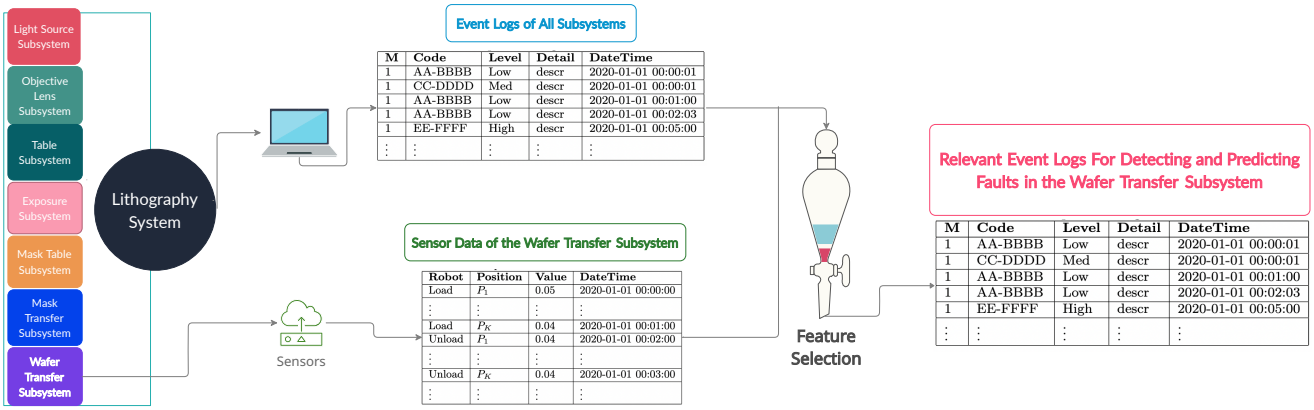

Figure 1: The composition of the lithography system, and an illustration of the feature selection problem addressed in this work.

further reduce the number of selected features. The main contributions of this paper can be summarized as follows:

1) We formalize the problem of feature selection for fault detection and prediction based on event log analysis. To our knowledge, this problem is not uncommon in the industry, but has not yet been studied.

2) To address this problem, we further propose a novel approach using two types of feature selection.

3) We demonstrate the effectiveness of our approach by conducting extensive experiments on 25 real-world datasets gathered from lithography systems.

The remainder of this paper is organized as follows. Section 2 introduces related work. Next, Section 3 introduces terminology, formalizes the problem, and presents our proposed method in detail. Section 4 empirically evaluates the performance of our method. Finally, Section 5 concludes the paper and discusses possible future directions.

2. RELATED WORK

Existing log-based anomaly detection methods usually consist of four main steps: *Log Collection*, *Log Parsing*, *Feature Extraction* and *Anomaly Detection* [7]. First, the *Log Collection* step is responsible for recording triggered events in the form of logs. A record is called a *log message*, which usually contains the date and time of occurrence and the detailed description of event. More concretely, detailed descriptions are typically presented in predefined templates, and may also include parameters. Second, the *Log Parsing* step aims at converting each log message into a specific *log event* template [21]. Usually, a *log event* corresponds to a unique template. Third, based on derived log events, the *Feature Extraction* step attempts to convert each *log sequence* into a *log count vector*. Specifically, a *log sequence* is composed of multiple *log events*. In general, a *log count vector* is a vector with each entry indicating the number of times that the corresponding *log event* was triggered. Note that the entries of the *log count vector* can be computed in another refined way [4]. Finally, the *Anomaly Detection* step performs anomaly detection based on the extracted *log count vectors*.

There exist many log parsing methods based on clustering [5], frequent pattern mining [2], or heuristic techniques [6]. Since our work centers around feature selection (e.g., log event selection) for fault detection and prediction, we will only consider the *Feature Extraction* and *Anomaly Detection* steps.

He et al. [9] have performed a systematic comparison of five state-of-the-art log-based anomaly detection methods, including DeepLog [3], LogAnomaly [11], PLELog [18], LogRobust [20], CNN [10]. Their findings show that log-based anomaly detection methods are often not as good as expected (i.e., as claimed by the original authors of each method) in real-world datasets. Importantly, they found that some log-based anomaly detectors, such as LogAnomaly, perform poorly on datasets with numerous log events. Therefore, feature selection can be exploited to improve the performance of these anomaly detection methods. However, we are not aware of any existing publications that attempt to address this problem.

3. METHOD

In the following section, we first introduce the terminology used in this work and then formalize the problem. Next, we elucidate our proposed method that consists of three modules: *Log Event Vectorization*, *Selection of Relevant Features*, and *Removal of Redundant Features*.

3.1 Terminology and Problem Formulation

We assume that we have access to two types of data. First, as shown in Table 1, we assume that the log data in a lithography machine, denoted by $\mathbf{X}$, has been collected and accurately parsed. Without loss of generality, we suppose there is a **Code** as the unique identifier for each *log event*, a **Level** roughly indicating the severity level of triggered *log event*, a **Detail** describing the detail of each *log message* that is an instantiation of a *log event* using a predefined template, and a **DateTime** containing the corresponding date and time. Hereinafter, we also call each *log event* a *log feature*. In addition, we may have log data for multiple lithography machines, and we use $\mathbf{M}$ to represent the corresponding name of the machine.

Ideally, applying an existing log-based anomaly detection

Table 1: An example event log. All values are fictional.

M	Code	Level	Detail	DateTime
1	AA-BBBB	Low	descr	2020-01-01 00:00:01
1	CC-DDDD	Med	descr	2020-01-01 00:00:01
1	AA-BBBB	Low	descr	2020-01-01 00:01:00
1	AA-BBBB	Low	descr	2020-01-01 00:02:03
1	EE-FFFF	High	descr	2020-01-01 00:05:00
⋮	⋮	⋮	⋮	⋮

Table 2: Example sensor data. All values are fictional.

Robot	Position	Value	DateTime
Load	P_1	0.05	2020-01-01 00:00:00
⋮	⋮	⋮	⋮
Load	P_K	0.04	2020-01-01 00:01:00
Unload	P_1	0.04	2020-01-01 00:02:00
⋮	⋮	⋮	⋮
Unload	P_K	0.04	2020-01-01 00:03:00
⋮	⋮	⋮	⋮

method on $\mathbf{X}$ can detect most faults related to load and unload robots. However, due to the large number of *log features*, it is computationally prohibitive to directly use most existing anomaly detection methods. Furthermore, the presence of irrelevant *log features* may significantly degrade detection performance and even lead to misleading detection results.

Second, we also assume the availability of sensor data, denoted by $\mathbf{Y}$, that measures the positions of robots. As shown in Table 2, there are measurements and corresponding timestamps from K different positions for the load robot and the unload robot, respectively. By using the *Log Event Vectorization* module in our proposed method (as will be explained in the sequel), $\mathbf{Y}$ can be rewritten as $(\mathbf{LP_1}, ..., \mathbf{LP_K}, \mathbf{UP_1}, ..., \mathbf{UP_K})$. Specifically, for $k \in \{1, ..., K\}$, $\mathbf{LP_k} = \{(Value_t, DateTime_t)\}_{t\in\mathbf{T}}$ denotes the corresponding time series of load robot from position k, and at the same time $\mathbf{UP_k} = \{(Value_t, DateTime_t)\}_{t\in\mathbf{T}}$ denotes the corresponding time series of unload robot from position k, where $\mathbf{T}$ denotes the timestamp when the time series was sampled.

By applying an appropriate time series anomaly detection method on $(\mathbf{LP_1}, ..., \mathbf{LP_K})$ and $(\mathbf{UP_1}, ..., \mathbf{UP_K})$, we can detect certain faults (especially gradual faults) of the load and unload robot, respectively. However, due to the limited fault information contained in $\mathbf{Y}$, these faults are difficult to predict using sensor data only.

Therefore, we aim to address the following problem: *Suppose there is a complex system Δ that is composed of several interconnected subsystems $\{\Gamma, \Lambda, ..., \Theta\}$. Given a database of logs $\mathbf{X}_\Delta$ generated by the entire system Δ and a database $\mathbf{Y}_\Theta$ consisting of time series measured by sensors from a certain subsystem Θ, we assume the set of unique log events in $\mathbf{X}_\Delta$ is $\mathbf{C}_\Delta = \{C_1, C_2, ..., C_N\}$. Based on $\mathbf{Y}_\Theta$, we attempt to select a subset of $\mathbf{C}_\Delta$, denoted by $\mathbf{C}_\Theta = \{C_1^{'}, C_2^{'}, ..., C_L^{'}\}$ with $L \ll N$ and resulting in a new database $\mathbf{X}_\Theta \subset \mathbf{X}_\Delta$, that mainly keeps relevant log events for detecting and predicting faults occurred in subsystem Θ.*

Specifically, as shown in Figure 1, system Δ is a lithography machine and subsystem Θ is the wafer transfer subsystem in this work. To solve the above problem, we propose a method consisting of three modules, namely *Log Event Vectorization*, *Selection of Relevant Features* and *Remove of Redundant Features*, each of which will be separately described next.

3.2 Log Event Vectorization (Module 1)

The first module, namely *Log Event Vectorization*, mainly aims at converting unstructured or semi-structured log events into time series data. Considering the log messages given in Table 1, it is straightforward to generate a time series for each **Code** per machine by keeping only the corresponding records. Without loss of generality, we assume that the smallest unit of time in the original data is seconds. Accordingly, for each log event in a certain machine, we can obtain a time series in the form of $\{(Value_t, DateTime_t)\}_{t\in\mathbf{T}}$, meaning that this log event is triggered $Value_t$ times at the timestamp $DateTime_t$ for $t \in \mathbf{T}$, where $\mathbf{T}$ represent all time points (an ordered list) when this log event was triggered.

After preliminary results and based on domain knowledge, we have decided to take each day as an interval to count the number of times a log event is triggered. Besides, we take the start point of this time interval to represent the time point when this log event is triggered.

3.3 Selection of Relevant Features (Module 2)

Given that the load robot and unload robots are very similar, for simplicity, we only consider the load robot when elucidating the proposed method. That is, we assume a multivariate time series database $\mathbf{Y} = (\mathbf{Y_1}, ..., \mathbf{Y_K})$. Note that all these time series are of equal length. Therefore, we assume $\mathbf{Y_k} = \{y_{kt}\}_{t\in\mathbf{T}}$ for $k \in \{1, ..., K\}$, with $\mathbf{T}$ denoting the timestamp when $\mathbf{Y_k}$ was sampled. Meanwhile, after applying the *Log Event Vectorization* module on the log event database $\mathbf{X}$, we can obtain another multivariate time series database $\mathbf{Z} = (\mathbf{Z_1}, ..., \mathbf{Z_n}, ..., \mathbf{Z_N})$, where N represents the number of unique log features (i.e., log events). Although different log events are usually triggered at different time points, we have taken each day as an interval to count the number of times that each log event is triggered. As a result, for $n \in \{1, 2, ..., N\}$, $\mathbf{Z_n}$ has a fixed length and thus we assume $\mathbf{Z_n} = \{z_{ns}\}_{s\in\mathbf{S}}$, with $\mathbf{S}$ denoting the timestamp when $\mathbf{Z_n}$ was sampled.

To detect faults, for $k \in \{1, ..., K\}$, we can apply a univariate time series anomaly detector $\phi(\cdot)$ on $\mathbf{Y_k}$, resulting in $\phi(\mathbf{Y_k})$. Alternatively, we can apply a multivariate time series anomaly detector $\Phi(\cdot)$ on $\mathbf{Y}$ by jointly considering all $\mathbf{Y_k}$ for $k \in \{1, ..., K\}$, leading to $\Phi(\mathbf{Y_1}, ..., \mathbf{Y_K})$. As shown in Figure 2 (the two subplots at the bottom), the identification of faults is feasible by applying time series anomaly detectors on $\mathbf{Y}$. However, for gradual faults, just identifying them is not enough. It is also necessary to predict them accurately. Since there is only limited fault information in $\mathbf{Y}$, it is difficult to predict these faults based on $\mathbf{Y}$ alone. Therefore, we attempt to select relevant log features from $\mathbf{Z}$ to better detect and predict faults. For $k \in \{1, ..., K\}$, by considering $\mathbf{Y_k}$ as the target variable and $\mathbf{Z_1}, ..., \mathbf{Z_n}, ..., \mathbf{Z_N}$ as the prediction variables, it becomes a supervised feature selection problem. Compared to traditional supervised feature selection problems, however, there are three novel challenges:

1) The features considered are time series rather than

numeric tabular data;

2) The target and prediction variables are not of equal length, and their timestamps are also different;

3) Traditional similarity metrics (e.g. Euclidean distance, dynamic time warping) do not give meaningful results when measuring the relevance/similarity of the predictor variable to the target variable.

We now detail why traditional similarity metrics fail to provide meaningful results when trying to find relevant log features. As shown in Figure 2, we can see that from 'D3' some faults started to appear in the robot and became detectable after a certain time based on **P1** and **P2**. These faults disappeared after the replacement of specific components on 'Replacement Date'. Meanwhile, we can observe that log event **Z** was triggered several times before the replacement date. More importantly, it was triggered several times even before the faults became detectable based on **P1** and **P2**. At other times, this log event was not triggered. In other words, the log event **Z** could potentially be used to detect and predict these faults. However, if we consider their shapes (wrapped or not) or values (normalised or not), we can see that log feature **Z** is not similar to **P1** or **P2**. To address this problem, we propose a novel similarity metric that consists of three steps, as follows.

As summarized in Algorithm 1, for $k \in \{1, ..., K\}$, we first apply an appropriate univariate time series anomaly detector $\phi(\cdot)$ on $\mathbf{Y_k}$, resulting in a time series of anomaly scores $\mathbf{U_k} = \{u_{kt}\}_{t \in \mathbf{T}}$ (Lines 5-8). Second, for $n \in \{1, 2, ..., N\}$, we apply an appropriate univariate time series anomaly detector $\varphi(\cdot)$ on $\mathbf{Z_n}$, resulting in a time series of anomaly scores $\mathbf{V_n} = \{v_{ns}\}_{s \in \mathbf{S}}$ (Lines 9-12). Note that $\phi(\cdot)$ and $\varphi(\cdot)$ can be different considering that $\mathbf{Z_n}$ is sampled at a regular frequency (i.e., an observation per day) but $\mathbf{Y_k}$ is sampled at an irregular frequency (e.g., multiple observations in day A but no observation in day B). Third, we can select relevant log events by comparing $\mathbf{U_k}$ with $\mathbf{V_n}$ (Lines 13-23). Note that the lengths and scales of $\mathbf{U_k}$ and $\mathbf{V_n}$ may be different, but their peaks should overlap (for detection) or preferably the peak of $\mathbf{V_n}$ precedes the corresponding peak of $\mathbf{U_k}$ (for prediction) if we compare them using the same timeline. A peak here means a relatively high degree of outlyingness.

3.3.1 Time series anomaly detector $\phi(\cdot)$

Since $\mathbf{Y_k}$ is sampled at an irregular frequency, it may have multiple observations on a given day, but no observations in the following days. However, traditional time series anomaly detection methods usually assume that the input time series is regularly sampled. To circumvent this limitation, we adapt a simple yet effective anomaly detection strategy, which is called *persistence checking*. Specifically, for each time series value Y_{kt} in $\mathbf{Y_k}$, we compare this value with its previous value to obtain an anomaly score, defined as $U_{kt} = \phi(Y_{kt}) = |Y_{kt} - Y_{kh}|$ with $t = h + 1$. Particularly, this anomaly detector can deal with time series sampled at irregular frequencies.

3.3.2 Time series anomaly detector $\varphi(\cdot)$

Although $\mathbf{Z_n}$ is in the form of a time series, we are not concerned about the temporal order of observations when detecting anomalies. This is because if the lithography machine is working under a normal production environment, the occurrence of each log event should remain stable. Therefore, we can apply a traditional anomaly detector designed for tabular data on it. Specifically, we define $V_{ns} = \varphi(Z_{ns}) = \frac{|Z_{ns} - med(\mathbf{Z_n})|}{std(\mathbf{Z_n})}$ as the anomaly score for the sample point Z_{ns} in the time series $\mathbf{Z_n}$, where *med* and *std* denote the median and standard deviation of all sample points in $\mathbf{Z_n}$, respectively.

3.3.3 Feature selection for fault detection

We perform feature selection for fault detection by comparing obtained anomaly scores $\mathbf{U_k}$ and $\mathbf{V_n}$. Assuming that the robot works continuously for T days, for $\mathbf{V_n}$ we have a value per day. However, for $\mathbf{U_k}$, we may have multiple values on some days, but no values on most days. To make $\mathbf{U_k}$ and $\mathbf{V_n}$ comparable, we modify $\mathbf{U_k}$ as follows: for each day, if there are multiple anomaly scores, we take the maximum of these scores as the final anomaly score; if there is no anomaly score, we set the final anomaly score as zero. We denote the modified $\mathbf{U_k}$ as $\hat{\mathbf{U}}_\mathbf{k}$, which has the same length as $\mathbf{V_n}$.

On this basis, we compute the Kendall's τ coefficient [8] between $\hat{\mathbf{U}}_\mathbf{k}$ and $\mathbf{V_n}$ for feature selection. We prefer Kendall's rank correlation measure over other correlation measures for several reasons:

- it measures monotonicity relationships and has a straightforward interpretation;
- it does not require the tested features to follow a normal distribution, as anomaly scores usually do not follow a normal distribution;
- it is robust to noise and can be calculated easily.

Suppose $\mathbf{V_n} = (\alpha_1, ..., \alpha_T)$ and $\hat{\mathbf{U}}_\mathbf{k} = (\beta_1, ..., \beta_T)$, then Kendall's τ coefficient measures the similarity of orderings of elements in $\hat{\mathbf{U}}_\mathbf{k}$ and $\mathbf{V_n}$. Specifically, we let (α_1, β_1) , ..., (α_T, β_T) be paired observations. Furthermore, (α_i, β_i) and (α_j, β_j) are regarded as concordant if and only if one of the following conditions are fulfilled:

- $\alpha_i < \alpha_j$ and $\beta_i < \beta_j$;
- $\alpha_i > \alpha_j$ and $\beta_i > \beta_j$.

Otherwise, (α_j, β_j) are considered discordant. Kendall's τ coefficient is then defined as:

$$\tau = \frac{\#concordant - \#discordant}{\#concordant + \#discordant}, \tag{1}$$

where $\#concordant$ and $\#discordant$ represent the number of concordant pairs and the number of discordant pairs, respectively. Hence, τ can take a value in $[-1, 1]$, with a high absolute value indicating a high relevancy of log event $\mathbf{Z_n}$ to $\mathbf{Y_k}$ for detecting faults. By setting a threshold τ_0, we can select a subset of log events that are considered to be the most relevant for fault detection.

3.3.4 Feature selection for fault prediction based on Kendall rank correlation test

For $k \in \{1, ..., K\}$ and $n \in \{1, ..., N\}$, after obtaining $\mathbf{V_n} = (\alpha_1, ..., \alpha_T)$ and $\hat{\mathbf{U}}_\mathbf{k} = (\beta_1, ..., \beta_T)$ as described in Section *3.3.3*, we achieve feature selection for fault prediction by comparing $\mathbf{V_n}$ and the lagged values of $\hat{\mathbf{U}}_\mathbf{k}$. The underlying rational is that if $\mathbf{Z_n}$ is useful for predicting faults in $\mathbf{Y_k}$,

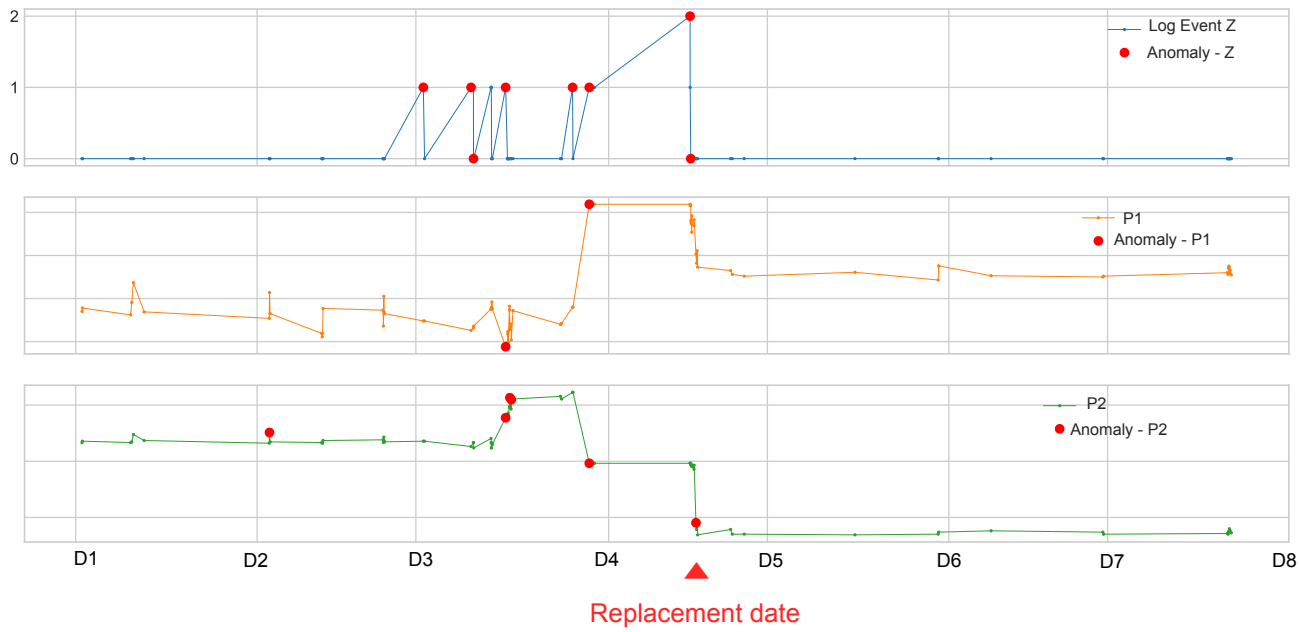

Figure 2: An example showing the relevance of a specific log event (upper plot) to detecting and predicting faults of a load robot based on sensor data (lower two plots). The shared x-axis represents the timestamp and the y-axes represent the measured values of each feature. Note that the y-axes of P_1 and P_2 are intentionally hidden and their timestamps are anonymized.

then the anomaly score vector $\mathbf{V_n}$ should be correlated with the lagged values of anomaly score vector $\hat{\mathbf{U}}_\mathbf{k}$. In other words, the value of Kendall's τ coefficient between $\mathbf{V_n}$ and $Lag(\hat{\mathbf{U}}_\mathbf{k}, l)$ should be high for some $l \in \{1, 2, ..., T\}$, where $Lag(\cdot, l)$ denotes the lag operator and l represents the the number of lagged times. For instance,

$$Lag((\beta_1, \beta_2, \beta_3, ..., \beta_{T-2}, \beta_{T-1}, \beta_T), 2) = (\beta_3, \beta_4, \beta_5, ..., \beta_T).$$

Hence, the resulting vector is shorter than the original vector.

For a specific l, we calculate the Kendall's τ coefficient between $\mathbf{V_n}$ and $Lag(\hat{\mathbf{U}}_\mathbf{k}, l)$ as follows. We let (α_1, β_{1+l}) , ..., (α_{T-l}, β_T) be paired observations, and then count the number of concordant pairs and discordant pairs, followed by using Equation (1) to obtain the corresponding value of τ. A high value of τ indicates a high relevancy of log event $\mathbf{Z_n}$ to predict faults in $\mathbf{Y_k}$. Particularly, although l can take a value in $\{1, 2, ..., T\}$, an overly large value will cause few observations, and may produce misleading results. Therefore, we only consider $l \in \{1, 2, 3, 4, 5\}$ in this work. Similarly, this coefficient can take a value in $[-1, 1]$, with a high absolute value indicating a high relevancy of log event $\mathbf{Z_n}$ to predict $\mathbf{Y_k}$. By setting a threshold τ_1, we can select a subset of log events that are considered to be the most relevant for fault prediction.

Our preliminary results suggest that only applying Kendall rank correlation to select features for fault prediction may not be sufficient. Given a value for l, we only consider a single and equal number of lag operations at each time point. Therefore, it may result in few features selected for prediction. To mitigate this problem, we utilize Granger causality test to select more features for fault prediction, as follows.

3.3.5 *Feature selection for fault prediction based on Granger causality test*

The Granger causality test is widely used to determine whether a time series is useful for predicting another time series [14]. Formally, given an information set $\Omega_t = (\alpha, \beta)$ with $\alpha = (\alpha_1, ..., \alpha_t)$ and $\beta = (\beta_1, ..., \beta_t)$, α is said to Granger cause β if and only if $\delta_f^2(\beta_t - P(\beta_t|\beta_{j:j<t}, \alpha_{i:i<t})) < \delta_r^2(\beta_t - P(\beta_t|\beta_{j:j<t}))$. Specifically, δ_f^2 represents the variance of prediction errors generated by the optimal linear predictor $P(\beta_t|\beta_{j:j<t}, \alpha_{i:i<t})$ based on full information $(\alpha_{i:i<t}, \beta_{j:j<t})$. Meanwhile, δ_r^2 denotes the variance of prediction errors generated by the optimal linear predictor $P(\beta_t|\beta_{j:j<t})$ based on reduced information $(\beta_{j:j<t})$. In other words, the 'usefulness' for prediction in Granger causality test means the ability to increase the prediction accuracy. Therefore, the Granger causality test is suitable for selecting log features that can be used to more accurately predict faults.

More concretely, we specify the following linear prediction equation:

$$\beta_t = c + \sum_{i=1}^{I} a_i \alpha_{t-i} + \sum_{j=1}^{J} b_j \beta_{t-j} + \mu_t, \tag{2}$$

where I and J represent the number of lagged operations considered for α and β, respectively, while μ_t denotes the corresponding residual. Intuitively, if $\sum_{i=1}^{I} |a_i| \neq 0$, we can say α is useful for predicting β. Accordingly, we call Equation (2) the full or unrestricted regression model. Meanwhile, we call the following equation the reduced or re-

stricted regression model:

$$\beta_t = c + \sum_{j=1}^{J} b_j \beta_{t-j} + \mu_t, \quad (3)$$

Specifically, given two time series $\alpha = (\alpha_1, ..., \alpha_t)$ and $\beta = (\beta_1, ..., \beta_t)$, the Granger causality test calculates the following F-statistic:

$$F = \frac{\frac{SSR_r - SSR_f}{J}}{\frac{SSR_f}{D-(I+J+1)}}, \quad (4)$$

where SSR_r is the sum of squared residuals on the reduced regression model and SSR_f is the sum of squared residuals on the full regression model. Besides, D denotes the degrees of freedom (i.e., the number of observations), while I and J have the same meaning as in Equation (2). Consequently, a higher value of F indicates that α is more useful for predicting β. As a result, α is declared Granger causal for β if the F-statistic is larger than the $(1 - Sig)\%$ quantile of an $F(J, D - (I + J + 1))$ distribution, where Sig denotes the significance level.

We should be aware that the Equation (4) requires several explicit and implicit assumptions to effectively identify Granger causality [14]:

- Linearity: the generating processes of the two time series are linear, and their causal effects are also linear;
- Continuous-valued series: the two time series are supoosed to have continuous-valued observations;
- Discrete-time: the two time series are sampled on a discrete and regular manner;
- Stationarity: the statistical properties of these two time series do not change over time;
- Known lag: the linear dependency on past values are assumed to have a known order;
- Perfectly observed: the two time series do not have measure errors;
- Complete system: there are no unmeasured confounders.

Overall, this subsection is summarized on Lines 5-23 in Algorithm 1.

3.4 Removal of Redundant Features (Module 3)

After selecting a subset of relevant log events for detecting and predicting faults in Module 2, we can further reduce the number of selected log events by computing the correlation between them. Specifically, we compute the pairwise Kendall's τ coefficient between selected log events and remove the redundant ones by setting a threshold τ_2 for the τ coefficient. More concretely, given two time series α and β, if the absolute value of their pairwise τ coefficient exceeds the threshold, we randomly remove one feature. This subsection is summarized on Lines 24-27 in Algorithm 1.

4. EXPERIMENTS AND RESULTS

To demonstrate the effectiveness and efficiency of our method, we perform a series of experiments. The experimental setup, results, and corresponding analysis are described as follows.

4.1 Data description

Given the novelty of the problems faced by this work, we are not aware of any publicly available datasets that can be used. The traditional benchmark datasets for log anomaly detection, including HDFS [16], BGL [12], Spirit [12] and Thunderbird [12], are mainly produced by complex systems such as cloud service providers or supercomputers. As a result, sensor data for monitoring hardware is often unavailable.

Therefore, we use 25 real-world datasets provided by our industrial partners to test our method. Specifically, the 25 datasets were generated by 20 different test machines, of which 16 machines each generated one dataset and the remaining 4 machines each generated at least 2 datasets. A summary of the datasets is given in Table 3. These datasets contain multiple types of faults, including gradual faults, sudden faults, and some unknown types of faults. Moreover, these faults may occur in the loading and/or unloading robot. In particular, for some machines it is unknown whether the fault occurred in the unloading or loading robot. Therefore, we examined the datasets of both robots to find the robot most likely to have failed.

4.2 Baseline, parameter settings and performance metrics

Based on domain knowledge and preliminary experiments, in module 2 we set the threshold $\tau_0 = 0.6$ for the similarity coefficient to select log events for fault detection. Furthermore, we set $\tau_1 = 0.6$ in the Kendall rank correlation and $Sig = 0.05$ in the Granger causality test to select features for fault prediction. In module 3, we further remove some highly correlated log features by setting $\tau_2 = 0.95$.

After constructing the event count matrix based on the selected features, we apply a commonly used unsupervised anomaly detector on it: KNN [13]. To demonstrate the necessity of feature selection, we also build an event count matrix from all original features and then apply KNN on it as a baseline. Specifically, a fault is considered detected if at least one of the points with the highest anomaly scores (we consider the top 3 points in our experiments) is on the date of the known fault. Moreover, a fault is considered predicted if one of the points with the highest anomaly scores is located slightly earlier (we consider 1-7 days) than the date when the fault is known to occur. On this basis, as a performance metric, we count the number of datasets in which the faults are accurately detected and/or predicted.

4.3 Results and analysis

As shown in Table 3, our proposed feature selection method can help improve log-based anomaly detection performance. Specifically, based on the selected log features, KNN was able to accurately detect or predict faults in 24 out of 25 machines. In contrast, KNN can only accurately detect or predict faults in 17 out of 25 datasets based on all log features. One possible reason is that logs from all subsystems are entangled and the inclusion of many irrelevant log events renders the detection/prediction of gradual faults difficult.

5. CONCLUSION AND FUTURE WORK

In this work we have proposed a simple yet effective feature selection method for log based anomaly detection. This method has been empirically proven to be effective on 25

Algorithm 1 Feature Selection for Fault Detection and Prediction (FS4FDP)

Input: Event log database $\mathbf{Z} = (\mathbf{Z}_1, ..., \mathbf{Z}_n, ..., \mathbf{Z}_N)$; Sensor database $\mathbf{Y} = (\mathbf{Y}_1, ..., \mathbf{Y}_k, ..., \mathbf{Y}_K)$; Anomaly detector $\varphi(\cdot)$ for $\mathbf{Z}_n$; Anomaly detector $\phi(\cdot)$ for $\mathbf{Y}_k$; Threshold parameters τ_0, τ_1, τ_2 and Sig with $\tau_0 = \tau_1 = 0.6$, $\tau_2 = 0.95$ and $Sig = 0.05$ by default.

Output: Subset of log features $\mathbf{F}$

procedure FS4FDP($\mathbf{Z}, \mathbf{Y}, \varphi(\cdot), \phi(\cdot), \tau_0, \tau_1, \tau_2, Sig$)
  $\mathbf{F} \Leftarrow \{\}$
  $\mathbf{U} \Leftarrow \{\}$
  $\mathbf{V} \Leftarrow \{\}$
  for $k \in \{1, ..., K\}$ **do** ▷ Obtaining anomaly score for each sensor time series
    $\mathbf{U}_k = \varphi(\mathbf{Y}_k)$
    $\mathbf{U}.append(\mathbf{U}_k)$
  end for
  for $n \in \{1, ..., N\}$ **do** ▷ Obtaining anomaly score for each log event time series
    $\mathbf{V}_n = \varphi(\mathbf{Z}_n)$
    $\mathbf{V}.append(\mathbf{V}_n)$
  end for
  for $\mathbf{U}_k \in \mathbf{U}$ and $\mathbf{V}_n \in \mathbf{V}$ **do**
    $\hat{\mathbf{U}}_k \Leftarrow Modify(\mathbf{U}_k)$ ▷ Modifying the length of anomaly score vector of each sensor time series
    if $|\tau(\hat{\mathbf{U}}_k, \mathbf{V}_n)| > \tau_0$ **then** $\mathbf{F}.append(\mathbf{Z}_n)$ ▷ Feature selection for fault detection using Kendall rank correlation
    end if
    for $l \in \{1, 2, 3, 4, 5\}$ **do** ▷ Feature selection for fault prediction using Kendall rank correlation on lagged values
      if $|\tau(Lag(\hat{\mathbf{U}}_k, l), \mathbf{V}_n)| > \tau_1$ **then** $\mathbf{F}.append(\mathbf{Z}_n)$
      end if
    end for
    if $Pvalue(GrangerCausalityTest(\mathbf{V}_n, \hat{\mathbf{U}}_k)) < Sig$ **then** $\mathbf{F}.append(\mathbf{Z}_n)$ ▷ Feature selection for fault prediction using Granger causality test
    end if
  end for
  for $\mathbf{F}_i, \mathbf{F}_j \in \mathbf{F}$ with $i \neq j$ **do** ▷ Removing redundant features using Kendall rank correlation
    if $|\tau(\mathbf{F}_i, \mathbf{F}_j)| > \tau_2$ **then** $\mathbf{F}.delete(\mathbf{F}_i)$
    end if
  end for
  return F
end procedure

real-world datasets. In the future, we plan to include more datasets for testing, evaluating whether our approach generalizes to other settings, and investigating hyperparameter tuning. More importantly, we will try more anomaly detection methods when defining $\phi(\cdot)$ and $\varphi(\cdot)$. Particularly, the Equation (4) used in Granger causality requires several explicit and implicit assumptions to effectively identify Granger causal effects. Some of these assumptions may not be fulfilled by our use-case though. Therefore, we will further explore and improve Granger causality test [1] or other similar techniques to find log events that can be used to predict sensor time series anomalies. Furthermore, we have not fully explored the causal relationships between different log events. In the future, by constructing a causality graph using techniques such as the PC-algorithm [15] on log events, we can investigate the causal relationships between log events. As a result, it might be possible to pinpoint the root causes of anomalies.

6. ACKNOWLEDGEMENT

This publication is part of Project 4 of the Digital Twin research programme, a TTW Perspectief programme with project number P18-03 that is (primarily) financed by the Dutch Research Council (NWO).

7. REFERENCES

[1] A. Arnold, Y. Liu, and N. Abe. Temporal causal modeling with graphical granger methods. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 66–75, 2007.

[2] H. Dai, H. Li, C. S. Chen, W. Shang, and T.-H. Chen. Logram: Efficient log parsing using n-gram dictionaries. *IEEE Transactions on Software Engineering*, 2020.

[3] M. Du, F. Li, G. Zheng, and V. Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 1285–1298, 2017.

[4] H. Guo, S. Yuan, and X. Wu. Logbert: Log anomaly detection via bert. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[5] H. Hamooni, B. Debnath, J. Xu, H. Zhang, G. Jiang, and A. Mueen. Logmine: Fast pattern recognition for log analytics. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1573–1582, 2016.

Table 3: Summary of datasets and experiment results. *Replacement* indicates the date when some components of the robot are replaced (faults always happen on or earlier than this date). *#Messages* represents the number of log messages. *#Raw* denotes the number of log features in the original dataset and *#Selected* denotes the number of selected log features. Besides, AD indicates whether the fault is detected based on the corresponding log features, where 'Yes' means the failure has been successfully detected or predicted and 'No' otherwise. If a fault is detected early (i.e., predicted), we further use 'P' to highlight it. Moreover, if a fault is only detected or predicted with the top 3 points but not the top 1 point, we denote it with an asterisk. Note that the value of *Fault* is anonymized, where GF, SF and $Unknown$ represent gradual faults, sudden faults or unknown types of faults, respectively.

Machine	**Robot**	**Fault**	**Replacement**	**#Messages**	**#Raw(AD)**	**#Selected(AD)**
1	Unload	GF_1	2021-01-05	103294	301(Yes)*	34(Yes)
2	Unload	GF_1	2021-01-06	90974	246(No)	47(Yes/P)
3	Unknown (Load)	GF_1	2021-01-10	90783	276(Yes)*	37(Yes)*
4	Unload	GF_1	2021-02-08	93729	437(No)	48(Yes)*
5	Unload	GF_1	2021-02-08	76797	245(No)	16(Yes)*
6	Load	SF_1	2019-06-24	86988	303(Yes)	38(Yes)
7	Unload	SF_1	2020-03-10	95513	376(Yes/P)	49(Yes/P)
8	Load	SF_1	2020-07-21	23262	404(Yes)	47(Yes)
9	Unload	SF_1	2020-07-24	34158	366(Yes)	69(Yes)
11	Unknown (Load)	SF_2	2019-01-24	13173	434(No)	27(Yes)
12 (1)	Load	SF_1	2019-03-22	51877	358(Yes/P)	95(Yes/P)
12 (2)	Unload	GF_1	2020-10-06	100223	337(Yes)	56(Yes)
13	Load	SF_3	2019-12-04	94426	294(Yes)	51(Yes)
14 (1)	Unload	$Unknown$	2020-01-28	116177	246(Yes)	15(Yes)
14 (2)	Unload	GF_1	2021-03-11	117146	327(Yes)*	72(Yes/P)
15 (1)	Load	$Unknown$	2020-03-24	101482	238(Yes)	9(Yes)
15 (2)	Unload	$Unknown$	2020-11-18	96719	293(Yes)*	39(Yes)
17	Unload	GF_1	2020-06-03	107414	217(No)	39(Yes)
18	Load	$Unknown$	2020-06-09	93875	402(Yes/P)	21(Yes/P)
19 (1)	Load	GF_1	2020-06-14	94125	259(No)	14(Yes)*
19 (2)	Load	GF_1	2020-10-19	106073	310(No)	19(No)
19 (3)	Load	$Unknown$	2021-03-29	105027	227(Yes)*	12(Yes)*
20	Unload	GF_1	2020-06-22	106070	215(Yes)*	21(Yes)*
21	Load	$Unknown$	2020-08-23	28414	345(Yes/P)	76(Yes/P)
22	Load	GF_1	2021-03-29	89350	187(No)	15(Yes)*

[6] P. He, J. Zhu, Z. Zheng, and M. R. Lyu. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE international conference on web services (ICWS)*, pages 33–40. IEEE, 2017.

[7] S. He, J. Zhu, P. He, and M. R. Lyu. Experience report: System log analysis for anomaly detection. In *2016 IEEE 27th international symposium on software reliability engineering (ISSRE)*, pages 207–218. IEEE, 2016.

[8] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[9] V.-H. Le and H. Zhang. Log-based anomaly detection with deep learning: how far are we? In *Proceedings of the 44th International Conference on Software Engineering*, pages 1356–1367, 2022.

[10] S. Lu, X. Wei, Y. Li, and L. Wang. Detecting anomaly in big data system logs using convolutional neural network. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 151–158. IEEE, 2018.

[11] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, et al. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *IJCAI*, volume 19, pages 4739–4745, 2019.

[12] A. Oliner and J. Stearley. What supercomputers say: A study of five system logs. In *37th annual IEEE/IFIP international conference on dependable systems and networks (DSN'07)*, pages 575–584. IEEE, 2007.

[13] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 427–438, 2000.

[14] A. Shojaie and E. B. Fox. Granger causality: A review and recent advances. *Annual Review of Statistics and Its Application*, 9:289–319, 2022.

[15] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[16] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pages 117–132, 2009.

[17] R. B. Yadav, P. S. Kumar, and S. V. Dhavale. A survey on log anomaly detection using deep learning. In *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pages 1215–1220. IEEE, 2020.

[18] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang, X. Dong, and W. Zhang. Semi-supervised log-based anomaly detection via probabilistic label estimation. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1448–1460. IEEE, 2021.

[19] X. Zhang. An introduction to lithography machine. In *2021 6th International Conference on Modern Management and Education Technology (MMET 2021)*, pages 49–53. Atlantis Press, 2021.

[20] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, et al. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 807–817, 2019.

[21] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu. Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 121–130. IEEE, 2019.