ML Detection and Isolation of Functional Failures using Syndrome Diagnostics

Yan Li[1], Daniel Chan[2], Navid Zaman[3], Evan Apostolou[4], and Paddy Conroy[5]

[3] *PHM Technology, Melbourne, VIC, 3068, Australia*
navid.zaman@phmtechnology.com

Abstract

Failure identification in complex engineering systems often needs to be in form of multilevel analysis, the first of which involves detection and isolation of functional disruptions across the system down to a discrete item. Considering that there is a functional flow(s) of operation in the machine, the loss or deviation of that functional flow(s) will imply a Functional Failure of the system. Most often, the disruption may propagate from where it may be first found and thus, the root issue can be isolated via the causal relationship between components and their flows. This paper references the idea of Causation-based AI: intelligence that brings together the fast estimations of machine learning and the domain-based physics of failure via qualitative models in the form of Syndrome Diagnostics (SD). There are three main routines in SD. Firstly, the current operating mode of the system is determined. Secondly, functional failure detection techniques are used to detect the existence of an anomaly. Thirdly, a functional failure isolation routine is executed to isolate the failing component, which is composed of two steps: using classification methods to generate a predicted syndrome and matching this generated syndrome with the failure syndrome pattern extracted from the Digital Risk Twin of the system (where the causation aspect is taken advantage of). A series of experiments are conducted and 90% of the failures in validation data have been correctly identified, which verifies the effectiveness of SD in terms of functional failure detection and isolation.

1. INTRODUCTION

Diagnostics, the processes of detection and isolation of failures (Ly, Tom, Byington, Patrick, & Vachtsevanos, 2009), check the system health conditions. Failure detection is to detect the occurrence of failures in the system and failure isolation is the process of determining which type of failure occurs among all possible failures. Failure detection and isolation (FDI) is of great importance in complex and mission-critical engineering systems in that it helps reasonably schedule Predictive Maintenance (PdM) at an optimal timing to prevent catastrophic failure consequences. Detecting failures at an early stage can reduce unexpected system breakdowns and costs – the most urgent industrial issue to be addressed (Ye, 2018), which requires real-time condition monitoring.

The development of industry 4.0 and the Internet of Things (IoT) platforms provides a large amount of sensor data, which makes it possible to monitor system health state with multiple sensors (Zhang, Liu, Zhang, & Miao, 2020), thus facilitating the application of machine learning or artificial intelligence (AI) methodologies for failure diagnosis. In recent years, machine learning technologies such as classification (X. Zhao, 2012) and anomaly detection (Farbiz, Miaolong, & Yu, 2020) have elucidated the capability and feasibility of solving the FDI task. Even though this type of pure data-driven method can make the most of the sensor readings, they might not provide reliable information on system health conditions due to the lack of engineering domain knowledge, based on which ineffective maintenance would be arranged resulting in machine failure or shutdown eventually.

To provide reliable detection of functional failures in a system, we propose a real-time health condition monitoring machine learning system – Syndrome Diagnostics (SD), which not only takes advantage of the fast data analysis and pattern recognition of cutting-edge machine learning algorithms but also leverages the domain expertise and reliability of physics-based system modeling, that is, a Digital Risk Twin of the system of interest, to detect and isolate functional failures in real-time.

2. FUNCTIONAL FAILURES

2.1. Definition of Functional Failures

An engineering system and its constituent items may be defined by its function(s), and indeed this is a key step in the authoring of an Failure Mode Effects Analysis (FMEA), a ubiquitous analysis and document undertaken for virtually all critical engineering products.

If items within the system may be defined by their functions, then the failure of these items may be understood through a disruption of their capability to function. If an item can no longer perform its as designed for function, then the item may be said to have failed. It is this paradigm that allows us to conceptualize functional failures as a lens to analyze the system of interest (Rudov-Clark & Stecki, 2014).

Functional failures are of course, an abstraction of the root causes of failure that impact a system. An item does not experience a functional failure without a physical (in the broad sense of the word) event precipitating it. In the framework put forward in this paper these physical events (aka physical failures) are seen as the physical failure paths that act internally to the item experiencing the failure (e.g. the mechanism of failure is occurring on an item that is resulting in that item entering a fault state). These physical failures disrupt item function resulting in a functional failure (Zaman, Apostolou, Li, & Conroy, 2021). That functional failure may then propagate throughout the rest of system resulting in system wide function disruption.

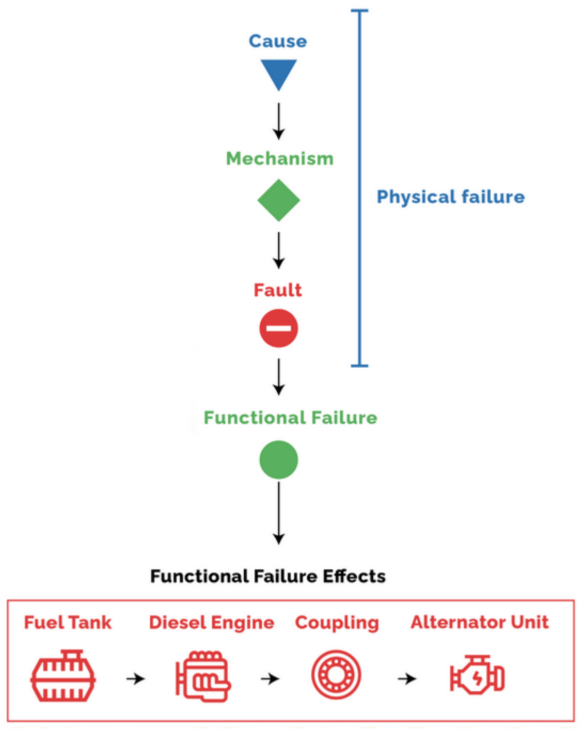


Figure 1. Failure Diagram

2.2. Modeling of Functional Failures

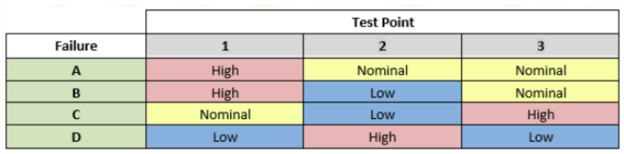
The basic structure of physical failures resulting in a local functional failure which then impacts the broader system is relatively recognizable and comparative to traditional FMEA generation processes and templates (DoD, 1980). As such this represents core knowledge of RAM engineers / teams. Transitioning this knowledge base away from static documentation to a model-based framework has many benefits for the RAM specialists themselves (Conroy, Kim, Tran, & Chan, 2021) but a fortuitous benefit is the externalization of knowledge into a structure that may be analyzed by those outside of the RAM sphere, including those in the diagnostics domain.

2.2.1. Representing System Dependencies

Utilizing the functional modelling topology outlined in (Rudov-Clark & Stecki, 2014) a system structure can be modelled where items that are defined functionally are connected to one another via functional flow properties that are modelled as inputs and outputs of items. These functional flow properties are the physical properties that a system or item may act upon. As an example, a pump (item) produces (function) a liquid flow rate (flow property). The causal interactions between the functional flow properties in the system is a dependency map from which a number of simulation methods can be overlaid for analysis.

2.2.2. Generating a Propagation Table

Two such methods are Fuzzy Cognitive Maps (FCM) and Power Bond Modelling. Background to these simulations are given in (Conroy, Stecki, Bautista, & Ringeri, 2018) and (Dransfield & Teo, 1979) respectively. Ultimately what may be produced is a propagation table that maps functional failures (i.e. an abstracted root cause of failure event) to potentially observable changes in system functional flow properties (aka potential test points). As an example, a blockage failure of the previously defined pump may reduce its flow rate output, downstream of this failure there will be a loss of flow rate observed, whereas upstream, an increase in pressure may be observed. These failure-functional flow change relationships occur throughout a system and using a model-based representation they may be simulated and captured in order to be translated into a propagation table.This propagation table can be further minimized to generate a set of sensors that would uniquely identify each failure, thus resulting in a minimized propagation table.



	Test Point		
Failure	1	2	3
A	High	Nominal	Nominal
B	High	Low	Nominal
C	Nominal	Low	High
D	Low	High	Low

Figure 2. Propagation Table

2.2.3. Defining a Syndrome and Symptoms

Taking the above format for understanding failures there are two key concepts that need to be defined. Firstly, a symptom of failure. A symptom is a singular instance or location in the propagation table. A symptom is the qualitative reading that describes the change that has occurred at a location in the system, it relates to a functional flow property. Symptoms may manifest in this framework as a nominal response (the flow property is within its nominal bounds of failure free operation), low (flow property is lower that its nominal), and high (flow property that is higher than its nominal). Secondly, a syndrome (or syndrome of failure), which is a series of symptoms that together indicate the presence of a specific failure. This is represented by one row in the propagation table, tracing originating failure to expected, system wide responses.

3. MACHINE LEARNING INTRODUCTION

The two important types of learning problems in the machine learning domain are supervised learning and unsupervised learning. Supervised learning requires that the training data comprises both the input vectors and the target vectors, and the classification task is one of its examples that targets assigning each input vector to a label, that is, one of the finite number of discrete categories. The target vectors are not always available, therefore, unsupervised learning works for this case where the training data only consists of the input vectors. As an example of unsupervised learning, clustering is to discover groups of samples with similarities in the data.

The machine learning pipeline employed in SD is illustrated in Figure 3, mainly including the following steps:

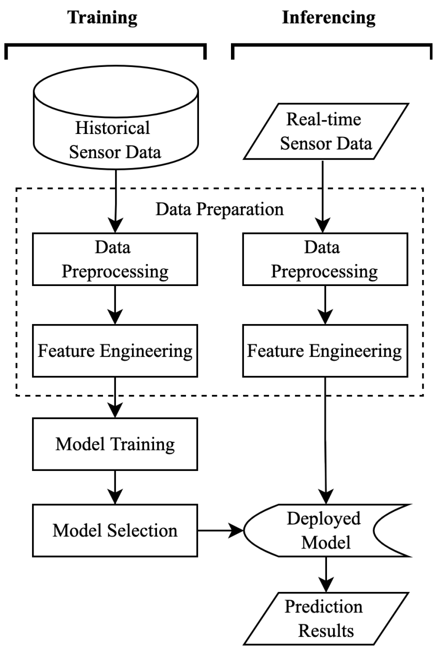


Figure 3. Machine Learning Workflow

- Data preprocessing is the first step of data preparation mainly involving tasks such as handling non-numeric values, imputation of missing values, re-sampling data, reducing noises, and scaling. Scaling is essential for some scale-sensitive machine learning algorithms like Principal Component Analysis (PCA), and the input variables with a higher value will dominate when training the models without pre-step scaling leading to unreliable results. MinMaxScaler and Normalizer are two options provided in SD for performing scaling before feature extraction (Buitinck et al., 2013). We also split the data into training, validation, and test sets in this step.
- Feature engineering is the second step of data preparation and it is mainly about selecting or extracting useful features from the input data to make it easier for algorithms to detect patterns in the data. Although feature engineering might not be needed for deep learning algorithms (Chollet, 2017), it is a significant aspect of traditional machine learning algorithms since it eliminates irrelevant and redundant data, and reduces the data size, thus enhancing model accuracy and speeding up computation. PCA, Statistics (Stat), Continuous Wavelet Transform (CWT), and some more algorithms are available in SD for feature engineering, among which the Stat method is basically extracting the statistics of the data as features, such as median, mean, maximum, minimum, root mean square values and so on. The user of SD can select a specific algorithm or make it run through all the combinations of feature engineering algorithms and training algorithms to get the best one.
- Model training and selection steps are performed after feature engineering, where hyper-parameter tuning is also involved. Bayesian Optimization method is a candidate for selecting a set of optimal hyper-parameters for the model. Different types of models with their best hyper-parameters are trained and tested on the test data set, then the best model is chosen based on the defined evaluation metrics.
- The selected best model is deployed for health condition monitoring. Similar to the training phase, the real-time sensor signals need to be first preprocessed and features need to be extracted by the feature engineering step. In most cases, the scalar and extractor used in real-time have been trained in the training phase and only data transformation is performed. Then the deployed model is used for making predictions on the prepared data.

4. SYNDROME DIAGNOSTICS

4.1. Definition of Syndrome Diagnostics

SD is a machine learning-powered system that detects and isolates functional failures in an engineering system in real-time by analyzing sensor readings (measured responses) and matching the results with failure propagation patterns (syndromes) identified from the Digital Risk Twin (a structured causation model) of the system.

SD consists of the following two main high-level components: a training engine that trained machine learning models including deep learning models on historical data and a real-time inference engine that made predictions on real-time sensor signals with the trained models. In real-time, SD will first trigger an alert when an abnormal behavior of the monitored system is detected, it then runs failure isolation algorithms to get an actual syndrome of the system, and finally, it matches the actual syndrome with the syndrome generated by the causation model and isolates the specific failure. The resulted failure information helps maintenance operators to understand the health condition of a system and identify the component responsible for any occurring failure.

4.2. Overall Workflow

The overall workflow of SD is illustrated in Figure 4.

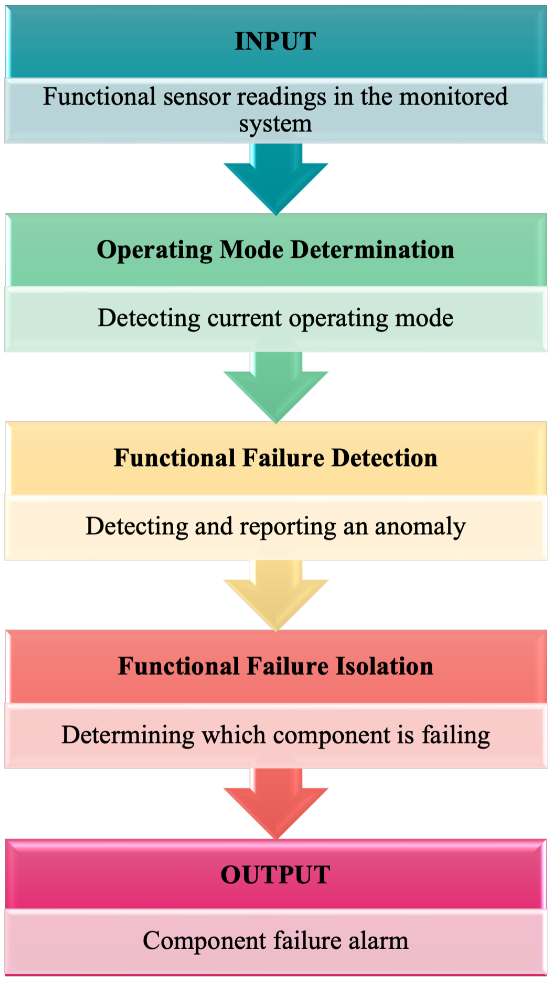

INPUT
Functional sensor readings in the monitored system
Operating Mode Determination
Detecting current operating mode
Functional Failure Detection
Detecting and reporting an anomaly
Functional Failure Isolation
Determining which component is failing
OUTPUT
Component failure alarm

Figure 4. Syndrome Diagnostics

Compared to the integrated systems-based framework for diagnosis and prognosis proposed in (Ly et al., 2009), SD takes into consideration the impact of different operating modes and adds an extra operating mode determination routine providing a sound basis for the following failure detection and isolation steps. In addition, SD separates the failure detection and isolation into two processes so that only abnormal time series window will be processed for failure isolation, which will significantly improve the inference efficiency in real-time monitoring. Besides, unlike pure data-driven techniques where FDI is treated as a classification task (Tidriri, Chatti, Verron, & Tiplica, 2016) and can only predict seen failures, SD implements a two-step failure isolation approach (classification and syndrome identification) leveraging the failure propagation table generated based on domain knowledge, not only making it possible to detect unseen failures, but also improving the effectiveness of failure diagnosis.

4.3. Input Data

Sensor readings are of increasing importance for the monitoring of complex, mission-critical systems to improve their availability with the development of IoT techniques. Various sensors continuously collect data from the physical world showing how the relationship between different variables of the system changes over time. The input data to SD is sensor reading – time-series data with failure labels. Each sensor is attached to a component to measure the functional flow of the system.

4.4. Operating Mode Determination

Most engineering systems have multiple operating modes corresponding to different functionalities. Systems behaves differently in different operating modes resulting in different trends or distributions of the attached sensor data. Therefore, it is significant to take into consideration the specific mode in which the monitored system is operating within for monitoring real-time system health status.

A candidate for operating mode determination is the classification method, which works for the case where the operating mode labels are available in the training phase. However, labeled system monitoring data is rarely readily available, especially for complex engineering systems with various operation modes, so unsupervised clustering methods come in handy. In general, we can categorize time series clustering into clustering by shape (similarity in space), clustering by time point (similarity in time), and clustering based on deep learning. SD takes advantage of both methods to handle different situations, where Support Vector Machine (SVM), Gradient Boost, and neural networks are used for classification, and KMeans, Deep Temporal Clustering (DTC) (Madiraju, Sadat, Fisher, & Karimabadi, 2018) are used for clustering.

4.5. Failure Detection

Correlation-based Anomaly Detection is the first step toward determining an issue with the system. The methods used here are expected to be lightweight, system-wide, and fast decision to trigger further analysis if needed. Compared to a specific failure, the prediction at this step would be a more general idea of "something wrong in the system". This is an important step before failure isolation for filtering out normal data in real-time to improve inference efficiency.

Failure detection is more of a novelty detection task across the system, the basic gist of which is that an anomaly score is computed based on one or multiple measurements, and the anomalies are determined based on a certain threshold, and if the anomaly score of observation is higher than this threshold. This threshold can be either pre-defined or computed from the data.

There are several anomaly detection methods available in SD, including individual methods such as K-Nearest Neighbors (KNN) (Angiulli & Pizzuti, 2002), Local Outlier Factor (LOF) (Breunig, Kriegel, Ng, & Sander, 2000), Local Correlation Integral (LOCI) (Papadimitriou, Kitagawa, Gibbons, & Faloutsos, 2003), and Angle-base Outlier Detection (ABOD) (Kriegel & Zimek, 2008), and method ensemble methods. In contrast to individual methods, outlier ensemble methods can obtain more robust anomalies by combining the results from the different algorithm executions (Aggarwal, 2017), with Isolation Forest, Feature Bagging, and Locally Selective Combination in Parallel Outlier Ensembles (LSCP) (Y. Zhao, Nasrullah, Hryniewicki, & Li, 2019) as representatives. Supervised algorithms are also available for the case where the anomaly labels are provided.

4.6. Failure Isolation

Failure isolation includes two steps, the first step is to classify sensor data into one of the classes (symptoms): high (1), nominal (0), and low(-1), and generate a syndrome from these labels for all the sensors. The second step is to match this generated syndrome with the syndrome pattern in the propagation table, so the failure mode can be pointed and the failing component is isolated.

4.6.1. Classification

Classification is a supervised learning task that categorizes sensor data into different classes. In the training phase, the high, nominal, and low labels are obtained by matching failures of historical data with the syndrome rows of the propagation table, then the data associated with the labels are fed into a classifier (classification algorithm) to get a trained model.

Another option is to use the normal classification routine, that is, directly feed the sensor and failure label to a classifier, which, however, is inapplicable to the industry scenario since in this case, the classifier can only classify the data into one of the failures it has seen in the training phase, while our two-step method can detect all the failures that could occur in the monitored system, that is, all the failures listed in the propagation table.

SD employs multiple classifiers including SVM, Gradient Boost, Multilayer Perceptrons Neural Network (MLP), Convolutional Neural Network (CNN), etc. All these algorithms are shared among all supervised tasks in SD.

4.6.2. Syndrome Identification

The next step after the syndrome is predicted is to match it with the minimized propagation table where there is this challenge that not all sensor symptoms match exactly with one row of the minimized propagation table. We come up with a method called Score Multiply to deal with this, and the main idea is, instead of getting the exact symptoms (high, nominal, low), to first get the probabilities of each symptom by making predictions with the trained classifier, then multiply the corresponding probabilities for each failure based on the propagation syndrome to generate failure scores, finally the failure with the highest score is selected as the identified failure mode. The component of this failure mode is the failing component isolated.

Here is a toy example. Assuming that Table 1 is the probabilities of each symptom for each sensor generated from the classification step and the first four columns of Table 2 is the minimized propagation table, the score of Failure 1 is calculated by multiplying the probability of Sensor 1 being High (0.3), the probability of Sensor 2 being Nominal (0.1), and the probability of Sensor 3 being Nominal (0.3), that is 0.009. In the same way, we can get the scores for the rest three failures as shown in the last column of Table 2. Eventually, Failure 4 with the largest score is identified as the failure occurring.

Table 1. Probabilities of Symptoms

Symptom	**Sensor 1**	**Sensor 2**	**Sensor 3**
Low	0.5	0.2	0.6
Nominal	0.2	0.1	0.3
High	0.3	0.7	0.1

Table 2. Minimized Propagation Table and Scores

Failure	**Sensor 1**	**Sensor 2**	**Sensor 3**	**Score**
Failure 1	High	Nominal	Nominal	0.009
Failure 2	High	Low	Nominal	0.018
Failure 3	Nominal	Low	High	0.004
Failure 4	Low	High	Low	0.21

5. CASE STUDY: LUBRICATION SYSTEM

5.1. System Description

The lubrication system is required to provide lubrication and cooling for other parts of a system such as bearings, splines and gears. The lubrication system is made up of an oil tank which supplies lubricant to the system – the lubricant is pumped through the system via a gear pressure pump which is driven by the engine but is also driven by the pressure differential at the bearing and sumps (Rolls-Royce, 1996).
The lubricant travels from the pump through a filter to collect any contaminants which may be deposited into the oil from the wear of bearings and gears. To prevent high pressures which may damage the filters, high pressure relief valves are located in several locations in the lubrication system (Rolls-Royce, 1996). Following the filters, the manifold then distributes the lubricant to the required areas through check valves and pipes, such as bearings, gears and splines.

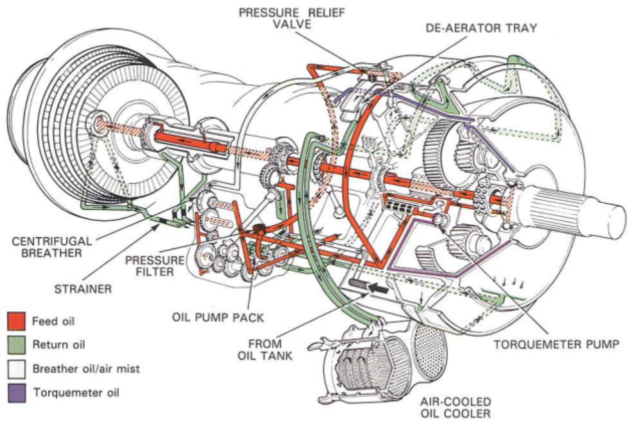
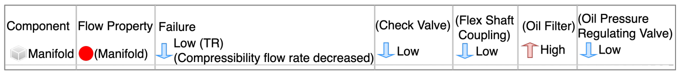


Figure 5. Lubrication System (Rolls-Royce, 1996)

5.2. Propagation Table

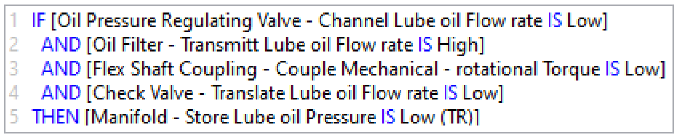
A digital diagnostic twin (DDT) (using Power Bond Modelling) of the lubrication system was developed to generate a propagation table which is used to drive the functional failure detection of SD.
The propagation table of the lubrication system contains 34 rows (corresponding to 34 possible failures) and 26 columns (corresponding to 26 components in the DDT). A subset of the propagation table is shown below.



Component	Flow Property	Failure	(Check Valve)	(Flex Shaft Coupling)	(Oil Filter)	(Oil Pressure Regulating Valve)
Manifold	● (Manifold)	↓ Low (TR) (Compressibility flow rate decreased)	↓ Low	↓ Low	↑ High	↓ Low

Figure 6. Manifold Failure - Propagation Table Row

The propagation table was minimized to generate 13 sensors for this case study, which would uniquely identify each failure. This led to a set of rules which can be used to determine where the failure has occurred within the system (i.e., diagnostic rules). SD uses a table of these rules, essentially a minimized propagation table, to determine where the failure has occurred within the lubrication system - notice the diagnostic rules in Figure 7 match the responses shown in the propagation table row image.



```
IF [Oil Pressure Regulating Valve - Channel Lube oil Flow rate IS Low]
 AND [Oil Filter - Transmitt Lube oil Flow rate IS High]
 AND [Flex Shaft Coupling - Couple Mechanical - rotational Torque IS Low]
 AND [Check Valve - Translate Lube oil Flow rate IS Low]
THEN [Manifold - Store Lube oil Pressure IS Low (TR)]
```

Figure 7. Manifold Failure - Diagnostic Rule

5.3. Dataset

There are 13 sensors in both the training and validation data sets. Figure 8 and Figure 9 shows the training and validation sensor signals respectively, where each color indicates different failure labels.

The first part of Figure 8 highlighted by green color is the healthy data and labeled as "No Failure", followed by 4 failures: "Oil Nozzles _ Flow rate (Lube oil) _ Low (Flow resistance increased)", "Line Volume B2-6 _ Pressure (Lube oil) _ Low (Compressibility flow rate decreased)", "Oil Filter _ Flow rate (Lube oil) _ Low (Flow resistance increased)", and "Flex Shaft Coupling _ Torque (Mechanical rotational) _ Low (Angular velocity differential decreased)". It is easily observed that there are many more healthy data points than failure data in the training data set, which reflects a real industry scenario, that is, historical failure data is much harder to obtain than the healthy data.

There are 10 colors in Figure 9 corresponding to 10 labels: "Line Volume _ Pressure (Lube oil) _ Low (Compressibility flow rate decreased)", "Manifold _ Pressure (Lube oil) _ Low (Compressibility flow rate decreased)", "No Failure", "Oil Gear Pressure Pump _ Flow rate (Lube oil) _ Low (Pressure differential decreased)", "Oil Nozzles _ Flow rate (Lube oil) _ Low (Flow resistance increased)", "Line Volume B2-6 _ Pressure (Lube oil) _ Low (Compressibility flow rate decreased)", "Line Volume B7-8 _ Pressure (Lube oil) _ Low (Compressibility flow rate decreased)", "Oil Filter _ Flow rate (Lube oil) _ Low (Flow resistance increased)", "Oil Nozzles _ Flow rate (Lube oil) _ High (Flow resistance decreased)", and "Flex Shaft Coupling _ Torque (Mechanical - rotational) _ Low (Angular velocity differential decreased)".

The ratio of the number of failures in training and validation sets is 4 to 9, which makes it impossible to isolate failures by using a normal classification routine where the classifier can only recognize seen failures, thus revealing the need for an effective strategy – the two-step failure isolation method we propose.

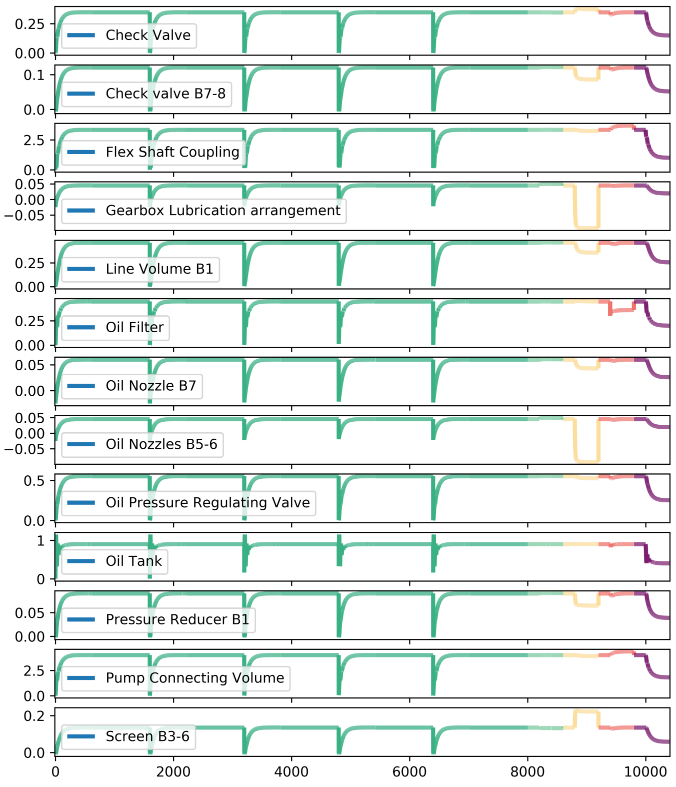


Figure 8. Training Data

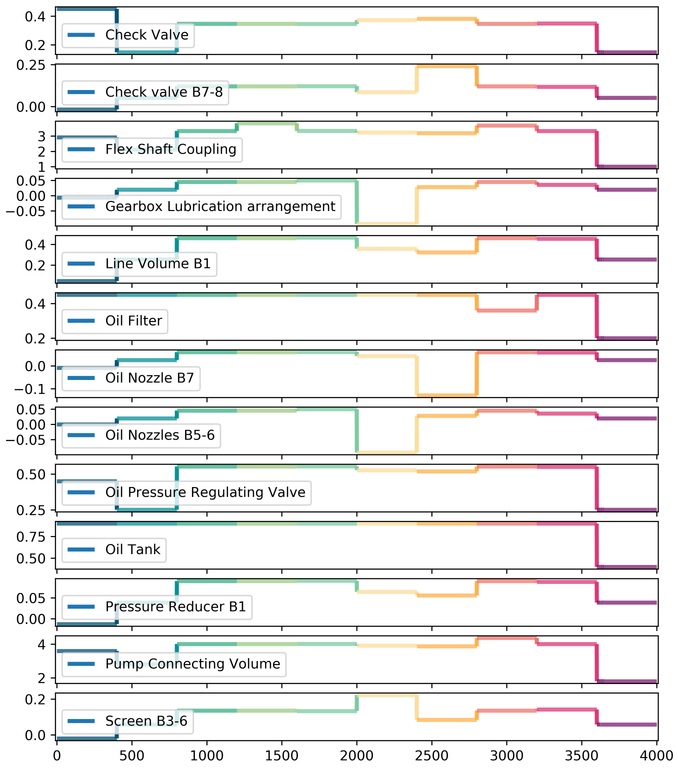


Figure 9. Validation Data

5.4. Anomaly Detection Visualization

We took windows of data from the training and validation sets with 20 points for each window and the windows with all normal data are labeled as 0s and the windows with all failure data are labeled as 1s, so we have 320 windows of healthy data and 120 windows of failure data for training. This task turns out to be an unbalanced binary classification task.

SD provides both unsupervised and supervised anomaly detection algorithms, we used the supervised Gradient Boost method to conduct experiments. The data was scaled and features were extracted, a machine learning model was then trained on top of the features. After that, the model was used to make predictions on the validation dataset. Accuracy and f1_score are used as the evaluation metrics and the results are 90.00%, and 91.37% respectively.

As indicated in the Figure 10 drawn by using PCA to project high-dimensional validation data into 2D, most of the predictions are correct in comparison to the ground truth labels, and only the circled points are misclassified.

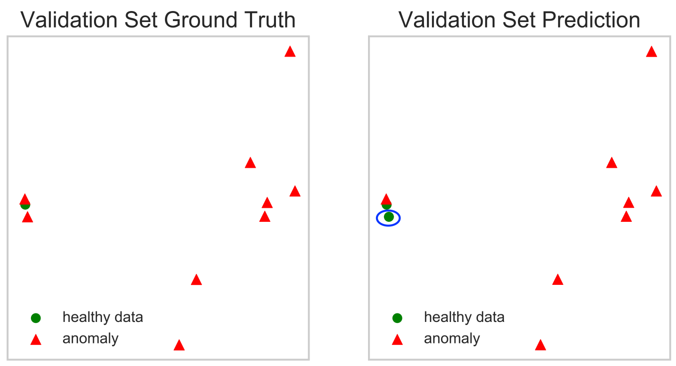


Figure 10. Anomaly Detection Results

5.5. Syndrome Identification Experiments

We conducted several experiments using SVM and Gradient Boost methods. The general steps were first using one of these two algorithms to train a model, then making predictions on the validation data, after that, we utilized the Score Multiply algorithm to isolate the failures by matching the predictions with the minimized propagation table. The accuracy metric was used to evaluate how many failures in the validation data were correctly predicted, and f1_score was another metric to assess the failure isolation performance.

The experiment results are shown in Table 3. The first experiment was using SVM and Stat to classify the validation data into high, nominal, and low labels, resulting in 60% accuracy and 53.33% f1_score. To improve the experiment performance, we kept the Stat method for feature extraction but used GB as the training method, leading to slightly higher accuracy and f1_score with values of 70% and 61.67% respectively.

Table 3. Failure Isolation Results

Algorithm	Feature Extraction	accuracy	f1_score
SVM	Stat	0.6	0.5333
GB	Stat	0.7	0.6167
GB	CWT	0.9	0.8667

While Stat allows us to explore the time domain of the input data, CWT makes it possible to explore the frequency domain. Therefore, the third experiment was conducted with GB and CWT, which dramatically increased the accuracy to 90%, and the f1_score was improved to 86.67%.

5.6. Evaluation Results Visualization

A confusion matrix of the third experiment results was illustrated in Figure 11, where the y axis represents the ground truth of 10 failure labels (including "No Failure") in the validation data and the x-axis represents the predicted failures which are the index of the same failures as the y-axis. We can observe that only failure "Manifold _ Pressure (Lube oil) _ Low (Compressibility flow rate decreased)" was isolated wrongly as "Flex Shaft Coupling _ Torque (Mechanical - rotational) _ Low (Angular velocity differential decreased)", and all the rest 90% has been correctly isolated.

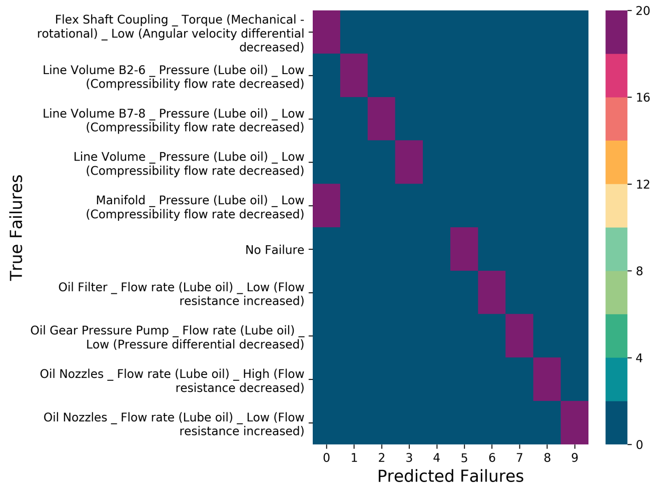


Figure 11. Confusion Matrix for Failure Isolation Results

6. Discussion and Future Work

The results above suggest the capability of SD isolating failures through the proposed two-step causation-based failure isolation strategy by training only on a small part of the failure data, which cannot be achieved by a normal data-driven classification routine.

The evaluation results can be further improved by exploiting balancing algorithms, hyper-parameter tuning, and more complex algorithms, which are all available options in SD.

However, the complexity of these training processes and the possible lack of data scientists in the engineering industry asks for automated training of machine learning models and eventually automated failure diagnosis, which is not shown in this paper but is already incorporated in SD. Besides, SD is still evolving as a machine learning system, which requires more attention on the software engineering side of it, such as the API to connect with the customer's database, the continuous training pipeline, and the real-time processing infrastructure.

7. Conclusion

Diagnosis of failures occurring is of great importance to engineering systems. This paper has presented a causation-based strategy to detect and isolate failures in an engineering system by taking advantage of both the domain knowledge about the system and machine learning algorithms. As a machine learning-powered application, SD has proven to be a valuable tool for failure diagnosis, which involves not only predicting the existence of an anomaly or failure but also identifying what the failure is and the exact component in the system that has caused this failure.

References

Aggarwal, C. C. (2017). *Outlier analysis*. Springer, Cham.

Angiulli, F., & Pizzuti, C. (2002). Fast outlier detection in high dimensional spaces. Springer.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). Lof: identifying density-based local outliers. ACM.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., ... Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *Ecml pkdd workshop: Languages for data mining and machine learning* (pp. 108–122).

Chollet, F. (2017). *Deep learning with python* (1st ed.). USA: Manning Publications Co.

Conroy, P., Kim, H., Tran, K., & Chan, D. (2021). Model-based rams: Optimizing model development in a distributed working environment. In *2021 annual reliability and maintainability symposium.*

Conroy, P., Stecki, J., Bautista, J., & Ringeri, A. (2018). Applications of artificial intelligence and decision making methods in phm. In *European conference of the prognostics and health management society.*

DoD, U. (1980). *Procedures for performing a failure mode effects and criticality analysis* (Tech. Rep. No. MIL-STD-1629A). Department of Defense.

Dransfield, P., & Teo, M. K. (1979). Using bond graphs in simulating an electra-hydraulic system. In *Journal of* *the franklin institute.*

Farbiz, F., Miaolong, Y., & Yu, Z. (2020). A cognitive analytics based approach for machine health monitoring, anomaly detection, and predictive maintenance. In *2020 15th ieee conference on industrial electronics and applications (iciea)* (p. 1104-1109). doi: 10.1109/ICIEA48937.2020.9248409

Kriegel, H.-P., & Zimek, e. a., Aurthur. (2008). Angle-based outlier detection in high-dimensional data. ACM.

Ly, C., Tom, K., Byington, C. S., Patrick, R., & Vachtsevanos, G. J. (2009). Fault diagnosis and failure prognosis for engineering systems: A global perspective. In *2009 ieee international conference on automation science and engineering* (p. 108-115). doi: 10.1109/COASE.2009.5234094

Madiraju, N. S., Sadat, S. M., Fisher, D., & Karimabadi, H. (2018). Deep temporal clustering : Fully unsupervised learning of time-domain features. *ArXiv*, *abs/1802.01059*.

Papadimitriou, S., Kitagawa, H., Gibbons, P. B., & Faloutsos, C. (2003). Loci: fast outlier detection using the local correlation integral. IEEE.

Rolls-Royce. (1996). *The jet engine*. Rolls-Royce plc.

Rudov-Clark, S. D., & Stecki, J. (2014). The language of fmea: on the effective use and reuse of fmea data. In *Aiac-13 thirteenth australian international aerospace congress.*

Tidriri, K., Chatti, N., Verron, S., & Tiplica, T. (2016). Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges. *Annual Reviews in Control*, *42*, 63-81. doi: https://doi.org/10.1016/j.arcontrol.2016.09.008

Ye, C. (2018). *A system approach to implementation of predictive maintenance with machine learning* (phdthesis). Massachusetts Institute of Technology.

Zaman, N., Apostolou, E., Li, Y., & Conroy, P. (2021). Real-time diagnosis of physical failures using causation-based ai. In *Proceedings of the 6th european conference of the prognostics and health management society.*

Zhang, H., Liu, E., Zhang, B., & Miao, Q. (2020, 08). Rul prediction and uncertainty management for multisensor system using an integrated data-level fusion and upf approach. *IEEE Transactions on Industrial Informatics*, *PP*, 1-1. doi: 10.1109/TII.2020.3017194

Zhao, X. (2012). Data-driven fault detection, isolation and identification of rotating machinery: With applications to pumps and gearboxes.

Zhao, Y., Nasrullah, Z., Hryniewicki, M. K., & Li, Z. (2019, May). LSCP: locally selective combination in parallel outlier ensembles. In *Proceedings of the 2019 SIAM international conference on data mining, SDM 2019* (pp. 585–593). Calgary, Canada. Retrieved from `https://doi.org/10.1137/1.9781611975673.66` doi: 10.1137/1.9781611975673.66

BIOGRAPHIES

Y. Li Yan Li is a data scientist at PHM Technology, mainly focusing on the research and development of the Syndrome Diagnostics product. She has got her Master of Information Technology degree at the University of Melbourne, Australia since 2020. Before her master's study, she had been working for 6 years in the logistics and training industries and has a good understanding of customer needs for business software tools.

D. Chan Daniel Chan is the Chief Operating Officer at PHM Technology. Since graduating as a systems engineer, Daniel has been involved with various projects across multiple industries including automotive, industrial engineering and aerospace defense. His interests are in the design of optimized model-based processes and methodologies to enable efficiency and consistency for engineering analyses.

N. Zaman Navid Zaman is a master of electrical engineering graduate from the University of Melbourne, Australia since 2020, where he focused on signals, systems and control theory. He has interned at Outotec Ausmelt for a few months before joining PHM Technology as lead data scientist. He has co-authored a RAMS paper previously, centered around the causation-based AI tool, Syndrome Diagnostics.

E. Apostolou Evan Apostolou is currently working at PHM Technology as an Engineer. His primary focus in the company is engineering support for Syndrome Diagnostics and MADe. Since graduating from the University of Melbourne with a Master of Engineering (Mechanical), he has worked at PHM Technology, gaining experience and knowledge in the safety, RAMS and PHM industries. He previously worked at RUAG Australia, Bosch Australia and Airbus Helicopters Germany where his roles varied from materials engineering, maintenance and system design.

P. Conroy Paddy Conroy is a Senior Engineer and Research Lead at PHM Technology. He received a Bachelor degree of Aerospace Engineering through RMIT University, Australia. Paddy has 6+ years of experience in a range of engineering disciplines including aerospace, naval, land, and automotive across mechanical and electrical domains. His specific interests are the research and development of RAMS and diagnostic model-based engineering methodologies and the processes required to integrate them into design and sustainment activities.