INL/EXT-21-64303

Light Water Reactor Sustainability Program

Process Anomaly Detection for Sparsely Labeled Events in Nuclear Power Plants

September 2021

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-21-64303

Process Anomaly Detection for Sparsely Labeled Events in Nuclear Power Plants

Jacob Farber[1]
Ahmad Al Rashdan (Principial Investigator)[1]
Hany Abdel-Khalik[2]
Yeni Li[2]
Mohammad Abdo[1]
Diego Mandelli[1]
Dongli Huang [2]

[1]Idaho National Laboratory
[2] Purdue University

September 2021

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy

EXECUTIVE SUMMARY

An essential aspect of online monitoring, subtle anomaly detection increases the detection lead time for equipment failure and enables a nuclear power plant (NPP) to mitigate unexpected partial or full outages, resulting in significant cost saving to the plant. Once an anomaly is detected by plant staff, its cause and severity are investigated. Because the vast majority of anomalies require some level of investigation, including some that require time-consuming examination, before they are passed over to the engineering organization for further analysis, plants are often equipped with tools to assist the staff in performing anomaly detection. Those tools operate as a black box and are often based on statistical methods that establish sensor correlations using preconfigured mathematical models and flag correlation deviations as anomalies.

Due to the number of anomalies detected at a given NPP on a daily basis, a significant number of flagged anomalies usually await examination for days or weeks. A primary cause of this backlog is that the methods used by the tools generate many false positives. Though this is usually attributed to oversensitive model settings due to very narrow normal operation bands, it can also be associated with the model development being inadequate for the process being monitored, or with missing model inputs that could have explained misclassified positives.

The performance of anomaly detection tools impacts their plant acceptance and utilization, especially when the effort to address false positives generated by the tool depletes the value or cost saved by using that tool. Thus, means to advance anomaly detection performance have been investigated by the Department of Energy’s Light Water Reactor Sustainability program. Previous and ongoing efforts have targeted unsupervised machine-learning (ML) methods, which do not require the labeling of any data fed into the ML model. By contrast, in supervised anomaly detection methods, every data point is labeled as either a normal or abnormal process condition, and the model is trained to replicate the classification process. Supervised methods usually outperform unsupervised methods, due to the added value in differentiating normal from anomalous states of the monitored process.

An NPP’s corrective action program requires it to track and document, via a dedicated report, the resolution of any issues that occur within the plant. Once created, each report is reviewed by a plant screening committee, and several classifications and decisions are made. Recently, a collaborating NPP developed an artificial intelligence and ML-based classifier to categorize a condition report (CR) into classes that can serve to label the data as normal or anomalous. Applying CRs as labels represents a semi-supervised use case. Semi-supervised ML assumes that labels exist for some data points (i.e., labeled anomalies, in this case) but not for the rest.

In this effort, semi-supervised ML methods were used to fuse data from CRs with anomaly detection methods in order to test the hypothesis that partially labeled anomalies would improve the accuracy of the anomaly detection methods. Specifically, two methods were used. The first is the deep Semi-supervised Anomaly Detection (deep SAD) method, which can handle labels ranging from fully unsupervised to fully supervised cases. The second is a newly designed ML method developed specifically for this effort and referred to as the high-order feature (HOF)-based method.

To evaluate these two methods in controlled environments, synthetic data generators were developed and used. The first datasets used a spring-mass-damper (SMD) system simulator commonly found in mechanical engineering references. This was used to create two use cases: a one- and a three-mass system. Anomalies were introduced by changing the spring and damper coefficients while the system was actuated by random forces. The second datasets used the commercial Dymola-Modelica software to build a simplified nuclear reactor model. Anomalies were added in the form of corrupted sensor readings and/or control commands. The deep SAD method was tested using the SMD system, while the HOF method was tested using both datasets.

Application of the deep SAD semi-supervised ML method demonstrated that labels can generate increased confidence in detecting true anomalies. This helped increase the number of true positives and decrease the number of false negatives—something that would aid in addressing the backlog of possible anomalies. Application of the HOF method demonstrated that labels can aid in down selecting from a candidate set of features to a more optimal subset in order to better differentiate between normal and anomalous conditions.

ACKNOWLEDGEMENTS

The authors wish to thank the Light Water Reactor Sustainability (LWRS) program for funding this effort. They also wish to thank Florida Power and Light company, part of NextEra Energy, for collaborating on this effort.

CONTENTS

FIGURES

ACRONYMS

CAP	corrective action program
CNN	convolutional neural networks
CR	condition reports
FNN	feedforward neural networks
GAP	global average pooling
HOF	high-order feature
LWRS	Light Water Reactor Sustainability
M&D	monitoring and diagnosis
ML	machine learning
NPP	nuclear power plant
PWR	pressurized water reactor
RWD	Randomized Window Decomposition
SAD	Semi-Supervised Anomaly Detection
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TSVM	Transductive Support Vector Machine

PROCESS ANOMALY DETECTION FOR SPARSELY LABELED EVENTS IN NUCLEAR POWER PLANTS

1. INTRODUCTION

One of the numerous responsibilities of nuclear power plant (NPP) operators in ensuring safe, reliable plant operations is to monitor and respond to plant alarms. Alarms are generated when plant process anomalies occur and are significant enough to exceed the particular threshold specified in the plant designs. Operators do not search for and detect subtle anomalies in the plant data. However, anomaly detection could help plants prevent such anomalies from escalating into unexpected equipment failure, especially since alarms may not provide sufficient lead time for the plant to act (Figure 1). This introduces a significant avoided cost saving to the plant. Often, the role of subtle anomaly detection—as well as that of investigating the cause and severity of such anomalies—is shared with dedicated monitoring and diagnosis (M&D) centers. An M&D center monitors multiple plants simultaneously and has dedicated staff whose primary task is to perform monitoring. Because the vast majority of anomalies require some level of investigation, including some that require time-consuming examination before they are passed over to the engineering organization for further analysis and actions, M&D centers are often equipped with anomaly detection tools that assist the staff in performing this function.

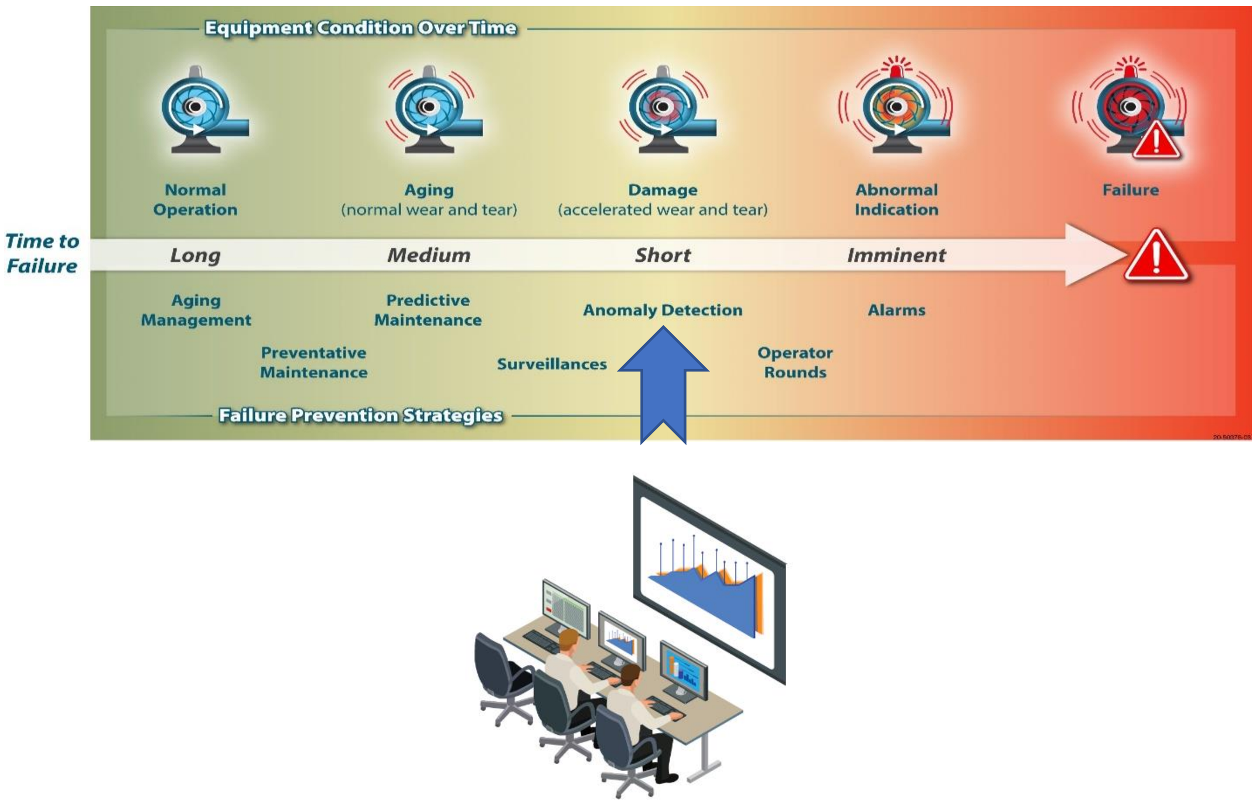

Figure 1. The M&D center’s role is to detect subtle anomalies in order to increase the detection lead time for equipment failures.

Anomaly detection tools operate as black boxes and are often based on statistical methods that establish sensor correlations using preconfigured mathematical models and flag any deviation from the pattern as being anomalous. Due to the number of anomalies flagged at a given NPP on a daily basis, the M&D center typically has a backlog of tens to hundreds of anomalies pending examination. A primary cause of this backlog is that the methods used by the tools tend to generate a large number of false positives. This is usually attributed to oversensitive model settings due to very narrow normal bands, but can also be associated with the model development being inadequate for the process being monitored, or with missing model inputs that could have explained misclassified positives.

The performance of anomaly detection tools impacts their utilization, especially when the effort of addressing false positives exceeds the value or cost saved by using those tools. Therefore, the Light Water Reactor Sustainability (LWRS) program launched efforts to explore means of advancing anomaly detection tools by leveraging state-of-the-art machine learning (ML) methods. Since the early 90s, ML techniques have been applied for condition monitoring in regard to fault detection,[1,2] system state identification,[3,4] and remaining life prediction.[5,6] Previous and ongoing LWRS program efforts have targeted unsupervised ML methods,[7,8] which do not require that the data fed into the ML model be labeled. The basic concept of such methods is based on grouping the monitored process state into multi-dimensional ranges. Any value falling outside those ranges is labeled anomalous. In supervised anomaly detection methods, every data point is labeled as either normal or anomalous, and the model is trained to replicate the classification process. Supervised methods are known to outperform unsupervised ones, due to the added value in training the model to distinguish normal states of the monitored process from anomalous ones. Figure 2 shows a simple example of unsupervised and supervised ML cases, the key distinction being that, in the unsupervised case, methods must rely on the data having natural splits in order to separate them. By contrast, in the supervised case, the labels make it much easier to distinguish between the known classes. These sample data show a relatively clear split, which may be hard to find in real data—particularly higher-dimensional data. This represents a key challenge for unsupervised learning, and helps explain why supervised learning outperforms unsupervised approaches.

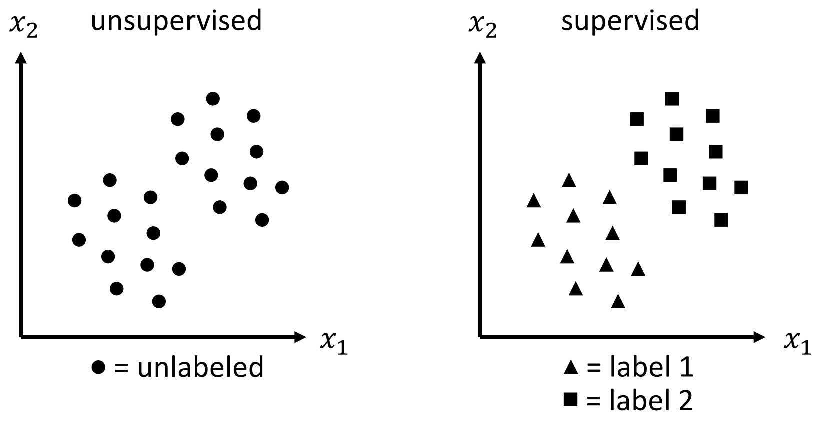

Figure 2. Simple example of unsupervised and supervised ML.

Given the great amount of sensor information now available, and the fact that supervised learning requires that all the data be labeled, the cost of human labor in applying supervised learning techniques becomes considerable, and the expert knowledge required in distinguishing fault signals is not always available. In an NPP, events are labeled using various data formats. Additionally, the labels can be generated using physics-based models[8] for failures never before observed. An NPP's corrective action program (CAP)[9] requires the plant to track and document the resolution of issues that occur in the plant[10] and is thus considered a valuable source for labels. Issue tracking/resolution is conducted through the issuance of a dedicated report, often referred to as a condition report (CR), action report, or issue report. Once created, such reports are reviewed by a plant screening committee, and several classifications and decisions are made regarding how the issue or condition should be addressed. However, the reports do not explicitly classify the condition into an equipment or process anomaly indicator. Recently, a collaborating NPP developed an artificial intelligence and ML-based classifier for categorizing a CR into classes that can serve to label the data as normal or anomalous.

Applying CRs as labels presents several challenges. First, any misclassified normal/anomaly conditions fed into the training process of the anomaly-detection ML model would impact the performance of the end-result model. This necessitates omitting from the training data any CR normal/anomaly classifications that carry low confidence. Second, a CR for a specific piece of equipment can impact several other, indirectly related pieces of equipment. This implies that anomaly patterns might exist for equipment, despite the missing anomaly labels. Third, some anomalies might be logged late in the anomaly's progression, resulting in parts of the anomaly pattern being mislabeled as normal. These three scenarios imply that, if CRs are used as a basis for labeling anomalies in an NPP, some anomaly labels will be missing, resulting in some anomaly data being mislabeled as normal. In ML terminology, this represents a partially labeled dataset, which would then be a candidate for semi-supervised ML. Semi-supervised ML methods assume that labels exist for some data points (i.e., labeled anomalies, in this case) but not for the rest (Figure 3). The unlabeled data do not negate the labeled data, but could correspond to either a normal or anomalous state.

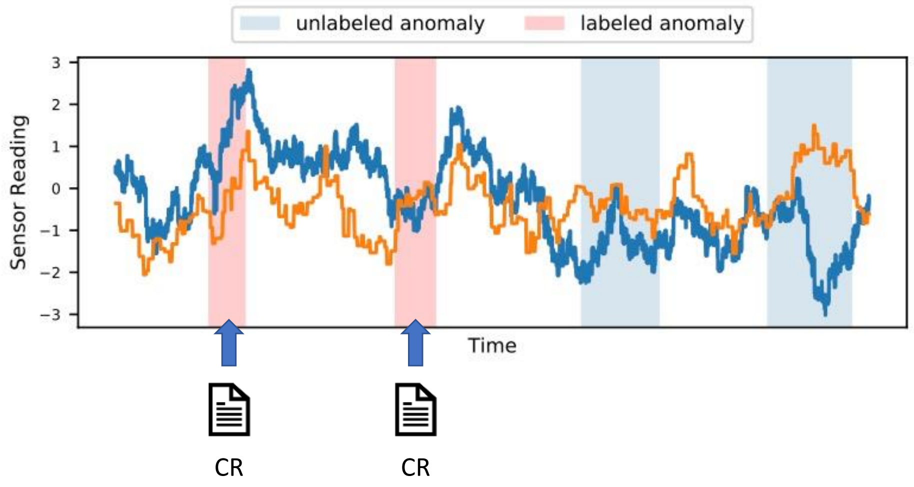

Figure 3. CRs present labels for parts of the sensors' time series data and can be used as input for anomaly-detection methods.

In this effort, semi-supervised ML methods were used to fuse CR data with anomaly-detection methods in order to test the hypothesis that partially labeled anomalies would improve the accuracy of the anomaly-detection methods and reduce false positives. This hypothesis was tested in other applications. For example, Dimla[11] reviews neural network (NN) solutions for tool condition monitoring in metal cutting, revealing that both the supervised and unsupervised architectures applied in that area achieved fairly similar results.[12]

In this effort, two semi-supervised methods were used to detect anomalies. These methods used some prior knowledge of a few sparsely labeled anomalous events, though no prior knowledge of whether the rest of the data were normal or anomalous. To evaluate the hypothesis in a controlled environment, synthetic data generators were created and used (see Section 2). The first ML method to be applied was the deep Semi-supervised Anomaly Detection (deep SAD) method, discussed in Section 3.1. The second method used to test the hypothesis was a high-order feature (HOF)-based ML method discussed in Section 3.2. The conclusions are summarized in Section 4.

1.1 Semi-supervised ML Literature Review

Various semi-supervised ML methods have been developed for a wide variety of application areas and are often heuristically customized for the respective problems. Nevertheless, semi-supervised ML algorithms can generally be divided into generative-model-based, density-based, graph-based, and heuristic approaches. Zhu[13] provides a comprehensive survey on semi-supervised ML algorithms. In this section, several illustrative sketches are provided to help explain each concept.

Generative models assume that the distribution of a given class takes some form of identifiable mixture distribution and use Bayes' rule to determine the prediction rule of the classifier, in the form of probabilistic distribution.[14] The generative-model-based semi-supervised ML algorithms use the more plentiful unlabeled data to identify the mixture distributions, and then label the distributions with the classes from the labeled data. See Figure 4 for a sketch of generative-model-based semi-supervised ML. The figure shows two clusters that are identified using Gaussian mixture models. Then, a few labeled instances of the two classes are used to correlate the clusters to specific classes. Many fault/anomaly detection practices employ generative-model-based semi-supervised ML algorithms. For example, the Skip Deep Generative Model is employed for fault detection in photovoltaic systems by training a model using the joint probability of labeled and unlabeled data with feature variables and state variables (classes), with anomalies being registered if the specific data point has an entropy far below that of the unlabeled data[15]; a deep generative model is used to predict remaining useful life[16]; and a variational-autoencoder-based deep generative model is used for bearing anomaly detection using a small subset of labeled vibration signals.[17] Typically, the generative approach is effective when the unlabeled data are clearly separable, but less effective for detecting subtle anomalies.

Density-based semi-supervised ML attempts to find a decision boundary at the low-density region that best separates one class of data from the other under the assumption that the unlabeled data from different classes are separated with large margin. Transductive support vector machine (TSVM), also called semi-supervised support vector machine (S^3VM), is the most commonly used low-density separator in semi-supervised ML. It learns a large margin hyperplane classifier by using the labeled data while forcing the hyperplane away from the unlabeled data.[18] See Figure 5 for a sketch of density-based semi-supervised ML. In the left half of the figure, the line represents the best-fit discriminator when using only a few labeled points. In the right half, the line represents the best-fit discriminator when using both the few labeled points and all the unlabeled ones. TSVM is a natural extension of SVM, which is applied to labeled data only; however TSVM is limited by the non-convex objective function. Approaches have been developed to solve this optimization difficulty[19]. Since TSVM is applicable wherever SVM is applicable, it is popularly employed to predict the boundaries between different fault/health classes in condition monitoring activities involving both labeled and unlabeled data. Reference 20 extends the semi-supervised Support Vector Data Description method by using negative samples and is applied for detection faults in rotating. Reference 21 introduces a safe semi-supervised S4VM approach for updating the hyperplane using successive online unlabeled data for detecting anomalies in vibration signals. Reference 22 investigates the performance of several density-based semi-supervised ML approaches (e.g., one-class SVM and Support Vector Data Description) for the condition monitoring of marine machinery systems for which large volumes of unlabeled operation data are available, though labeled fault samples usually are not.

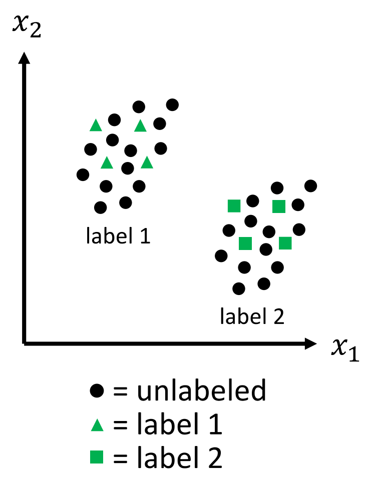

Figure 4. Sketch of generative-model-based semi-supervised ML.

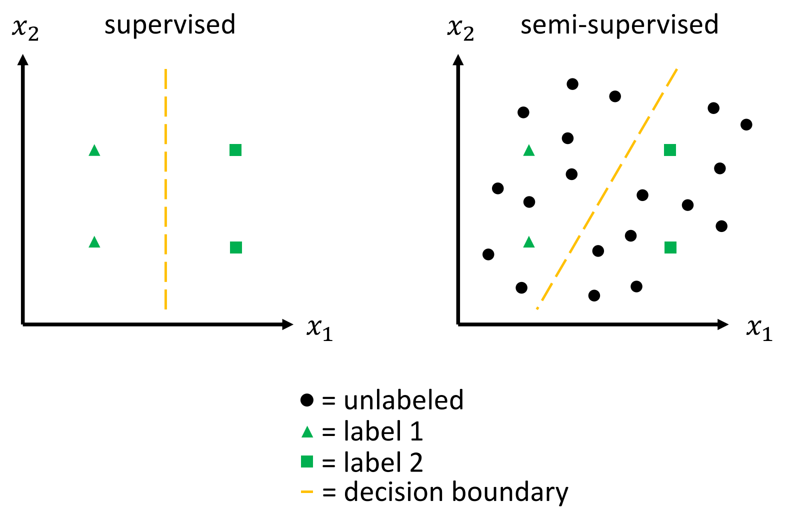

Figure 5. Sketch of density-based semi-supervised ML.

The graph-based semi-supervised ML algorithms construct a graph with labeled and unlabeled data represented as nodes, as well as weighted edges connecting the nodes to reflect the similarity between them.[23] The unlabeled data help with label propagation while minimizing the propagation energy. The graph-based semi-supervised ML guarantees good performance if the constructed graph fits the task with graphic interpretation of the results.[24, 13] See Figure 6 for a sketch of graph-based semi-supervised ML. In the figure, the lines represent connections between different data points. This can be an iterative algorithm depending on how the graph is created. The figure on the left shows the connections and labeled/unlabeled data points, the figure in the middle shows an early iteration as the algorithm tries to determine which points to cluster into which group, and the figure on the right shows a late iteration that correctly identifies all points to their respective classes. In condition monitoring, graph-based semi-supervised ML applications attempt to construct a problem-specific graph. For example, Reference 25 constructs an undirected weighted graph to guarantee the intrinsic structure of the data and accomplish the complex task of detecting and diagnosing faults with nonlinear traits, while Reference 26 employs manifold regularization, using the graph Laplacian of a graph-based representation to exploit the geometric structure of the marginal distribution of the condition monitoring data in the feature space, thereby outperforming the supervised classifications using information from the unlabeled data. Reference 2 provides another algorithm, using manifold regularization to extend SVM into Laplacian SVM in order to conduct multi-class fault detection vibration signals.

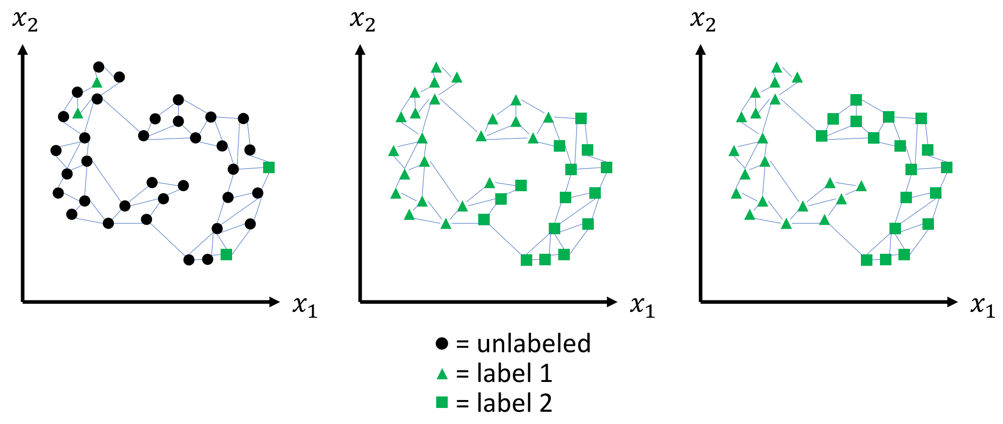

Figure 6. Sketch of graph-based semi-supervised ML.

Some semi-supervised ML methods (e.g., self-training and co-training) are heuristic, using unlabeled data within a supervised learning framework. For both these methods, the first step is to train the supervised classifier using only the available labeled data. In self-training, the classifier is then applied to the unlabeled data to generate more labeled samples and update the classifier with more labeled data. See Figure 7 for a sketch of semi-supervised ML using self-training. As with the graph-based example above, this is an iterative algorithm. The figure on the left shows the labeled and unlabeled points, the figure in the middle shows an early iteration as the algorithm begins labeling the unlabeled data points nearest the labeled data points, and the figure on the right shows the final iteration, with all the points now classified. Self-training can be applied in a preliminary investigation because it is so simple to implement. However, as self-training uses its own predictions (classifiers) to teach itself (i.e., classify the unlabeled data), early misclassifications may thus reinforce themselves. Different self-training techniques for semi-supervised ML classification are introduced and evaluated in a survey paper.[27] Reference 28 proposes a self-organizing feature map to soft label and update unlabeled data in order to detect changes/damages in image data.

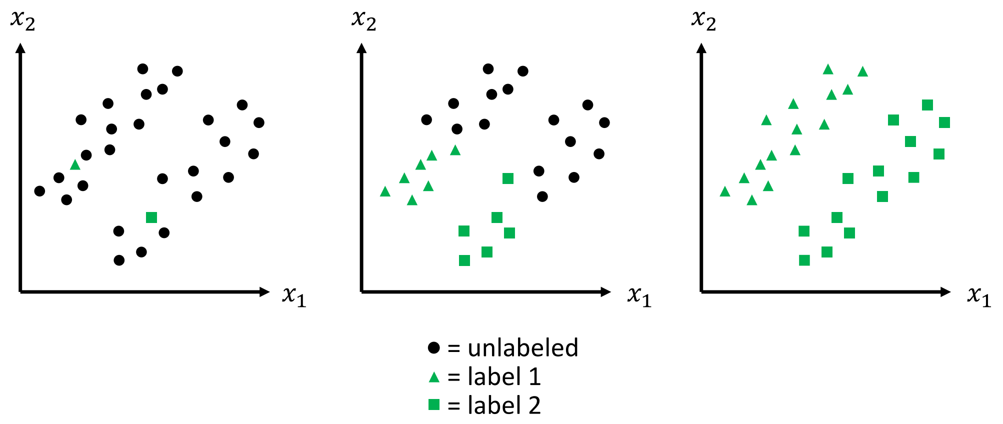

Figure 7. Sketch of semi-supervised ML using self-training.

Co-training is an extension of self-training that requires that features can be split into two views as two independent sources of information.[29] First, two classifiers are trained using labeled data from the two feature views. Then, each classifier classifies the unlabeled data and updates the other classifier using its most confident estimated labels. See Figure 8 for a block diagram of how co-training works. In this figure, X is the input data; $X^{(1)}$ is the 1st view of the data; the L and U subscripts are labeled and unlabeled, respectively; Y is the output data; and f is some classifier function. Efficiency and accuracy are improved as the classifiers from the two views teach each other. However, one limitation to the approach is that a natural feature split might not exist,[30] resulting in only a few applications of co-training in condition monitoring. Reference 31 provides an example of equipment remaining life prediction using co-training regression with a few failure units (labeled) and many suspension units (unlabeled).

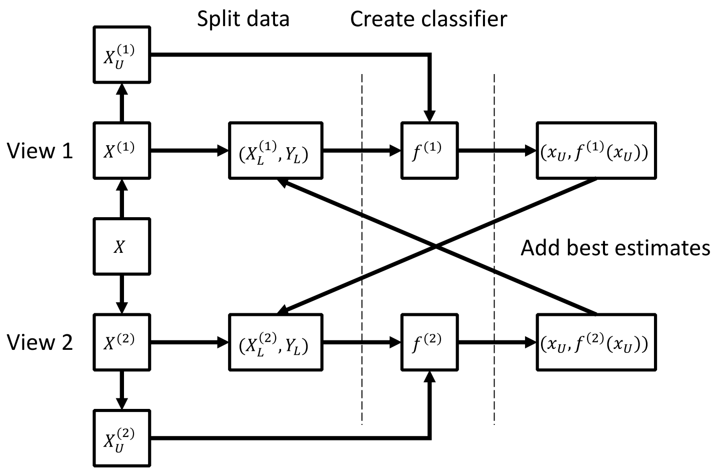

Figure 8. Sketch of semi-supervised ML using co-training.

2. SYNTHETIC DATA GENERATION

As discussed in Section 1, the anomaly detection methods developed and/or evaluated as part of this research required labeled time series data that contained both normal and anomalous behavior. For this pilot study on semi-supervised anomaly detection, using simulated data provided several advantages over using real data. First, it reduced the burden of cleaning the data, increasing the focus on the anomaly detection approaches. Second, since all anomalies were known, this provided good benchmark datasets for comparing the different approaches.

Two different approaches were used to generate simulated data: a spring-mass-damper (SMD) simulator, and a simplified pressurized-water reactor (PWR) simulation created using the Dymola-Modelica software package.

2.1 Spring-Mass-Damper Simulator

The first simulator that was developed used the idea of the SMD system, commonly found in mechanical engineering references. The basic building blocks of this system are springs, masses, dampers, sensors, and actuators: masses respond to forces according to Newton’s Second Law, expressed as force equals mass times acceleration; springs apply restorative forces to the mass, and these forces are a function of the displacement of the spring; dampers apply damping forces to the mass, and these forces are a function of the damper velocity; actuators apply forces directly to masses; and sensors measure some property of the mass, here, the position of the mass. Example one- and three-mass systems are sketched in Figure 9 and Figure 10, respectively; the sketch for the three-mass system shows how the simulator can be scaled up to generate more complicated system by combining the basic building blocks into larger configurations.

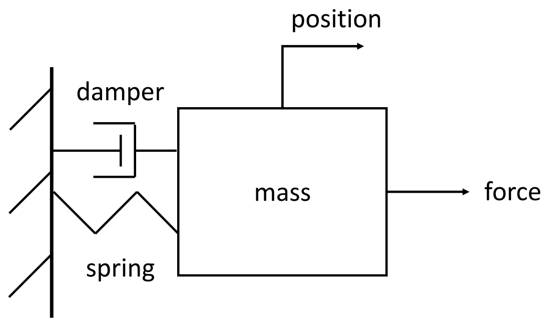

Figure 9. Sketch of a one-mass SMD system.

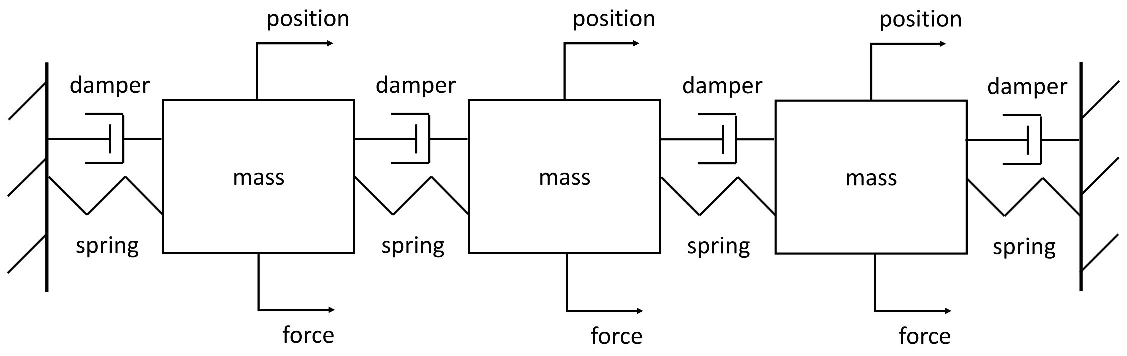

Figure 10. Sketch of a three-mass SMD system connected in series.

There are several reasons why the SMD system was selected for this research. First, it is conceptually simple, with just a few basic components; however, those components can be combined to generate high-order (HO) systems with coupled variables. Second, the system is easily scalable to include many sensors and actuators. These two characteristics are important in emulating the large-scale HO systems in NPPs. Third, the system allows the straightforward incorporation of process anomalies by modifying system parameters (here, the spring stiffness and damping coefficients). These modifications result in slowly changing anomalies that are difficult to see during manual inspection but are detectable using advanced analytical approaches.

For the one-mass system of Figure 9, the dynamics are described by a differential equation, as per Newton's Second Law. In this case, the sum of the forces on the mass is the sum of the spring force, damper force, and actuator force. The basic differential equation assuming a linear spring and damper can be written as follows:

$$m\ddot{x} + c\dot{x} + kx = F \tag{1}$$

where m is the mass, c is the damping coefficient, k is the spring stiffness, F is the actuator force, x is the displacement of the mass, $\dot{x} = \frac{dx}{dt}$ is the velocity of the mass, and $\ddot{x} = \frac{d^2x}{dt^2}$ is the acceleration of the mass.

The above differential equation is for a system with just one mass and one actuator. But, as mentioned, one benefit of using these basic building blocks is that they can be combined into larger system sizes (e.g., the three-mass system). When including multiple masses, each mass can either be attached to a fixed "ground" or to another mass. When masses are connected, the displacement and velocity of the spring and damper forces (see above) become the relative displacement and relative velocities between the two connected masses. This results in a differential equation for each mass present in the system, and the resulting set of equations represents a highly coupled and interacting dynamic system.

The above differential equation is for a system with a linear spring and damper, but this system is easily extendable to include nonlinear components. One method is to make the spring nonlinear, and this is often modeled by using a cubic polynomial with spring force equal to $k_1x + k_2x^3$. This nonlinearity is important because real systems are, to various extents, nonlinear, so the additional term makes the simulation more representative of NPP systems.

To further emulate real systems, this simulation also accounts for both process and measurement noise. Process noise introduces uncertainty in the model dynamics, because the true underlying dynamics can never be perfectly known. Measurement noise introduces uncertainty in the sensor readings, since sensors cannot perfectly capture the true parameter of interest they are estimating.

Finally, this simulation platform can easily insert anomalies by changing the system parameters (here, the spring stiffness k and damper coefficient c). These changes occur as a function of time, so during normal operation, the parameters are set to their nominal values. During anomalous operation, they are slowly varied, causing changes to the overall system behavior that affect the correlations between the variables, though these changes are subtle enough to go unnoticed by human operators.

Once the differential equations for the system have been determined, they must be solved for the mass positions. For some closed form forcing functions, the differential equations may be solved analytically; however, the forcing functions used here do not have a closed form. Thus, the best way to solve them is to use numerical integration approaches to provide an approximate solution. One of the most popular numerical integration methods is the Runge-Kutta solver. Most major programming languages have extensive functionality for numerical integration using robust, accurate integration routines. The simulation results are the mass positions and actuator forces. From a dynamic system theoretic point of view, these represent both input and output variables, and both are important for detecting anomalies in dynamic systems. In an NPP, examples of input variables include heater power, motor power, and valve position, while examples of output variables include temperature, pressure, and fluid level.

This report considers two different system configurations: a one-mass system and a three-mass system connected in series. In the one-mass system (Figure 9), the data are comprised of the mass position and actuator force for a total of two variables. Similarly, in the three-mass system (Figure 10), the types of data are the same, but correspond to three masses and three actuators for a total of six variables. The simulations covered a span of five years, with two anomalies per year, for a total of ten anomalies.

These one- and three-mass systems were simulated, but because they share such similar patterns, simulation plots are only shown for the three-mass system. For each of these plots, both the mass positions and actuator forces are shown. Figure 11 shows the simulation results for the full five years in order to reveal any long-term trends in the system behavior[a]. This plot shows the general scale of the variables and the noise, and gives a high-level overview of the interactions between the variables. Figure 12 shows the results for a two-month period, affording a closer view of the individual transients and the variable correlations. This figure shows the interactions between the variables and the time-constants of the transients, which were selected to be slow in order to emulate the slower transients seen in NPPs. Figure 13 shows the results for a three-month period in which operation transitioned from normal to anomalous after the black dotted line. This figure shows how subtle the anomalies are, which makes them difficult to detect without the use of advanced analytics.

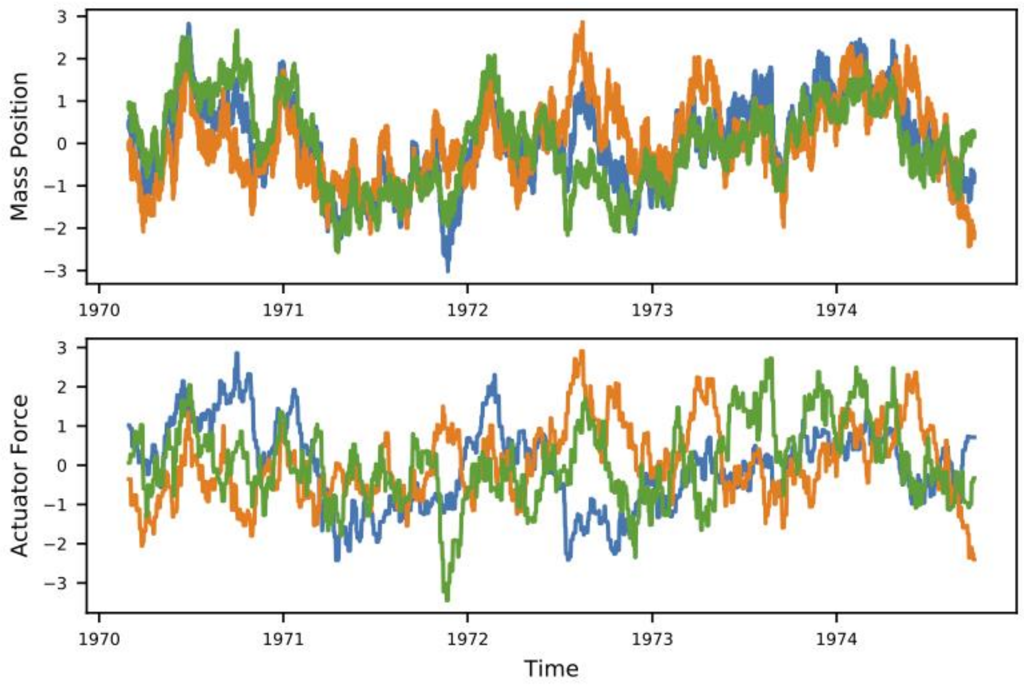

Figure 11. Simulation results for the three-mass system for the full five years. The top plot shows the mass positions, and the bottom plot shows the actuator forces.

[a] These years are shown because the Python package's default dates spanned the early 1970s.

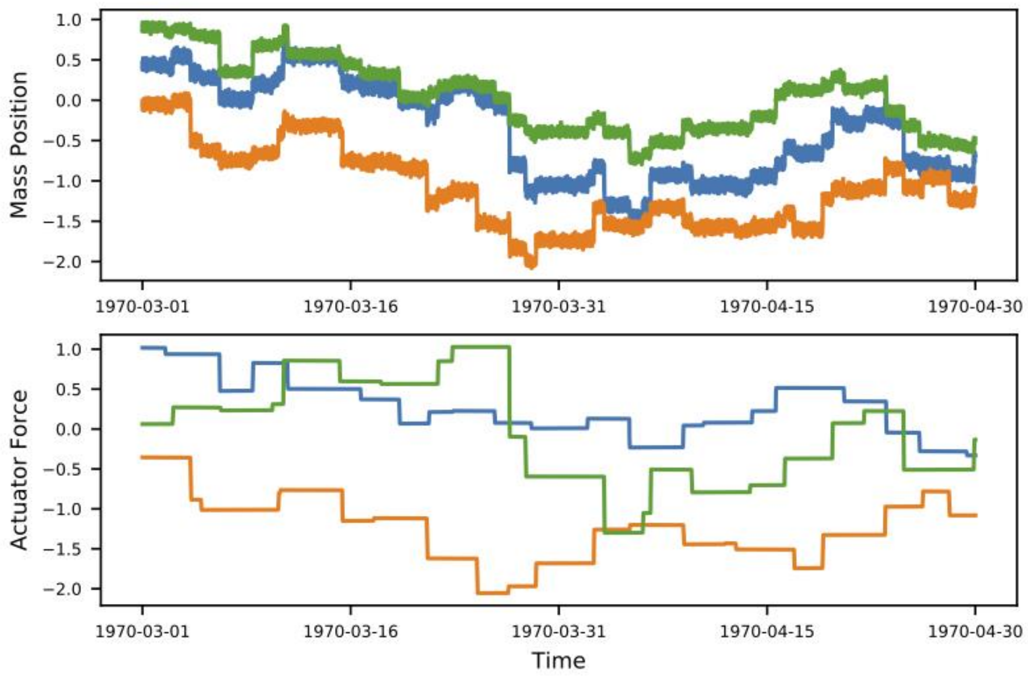

Figure 12. Simulation results for the three-mass system for two months.

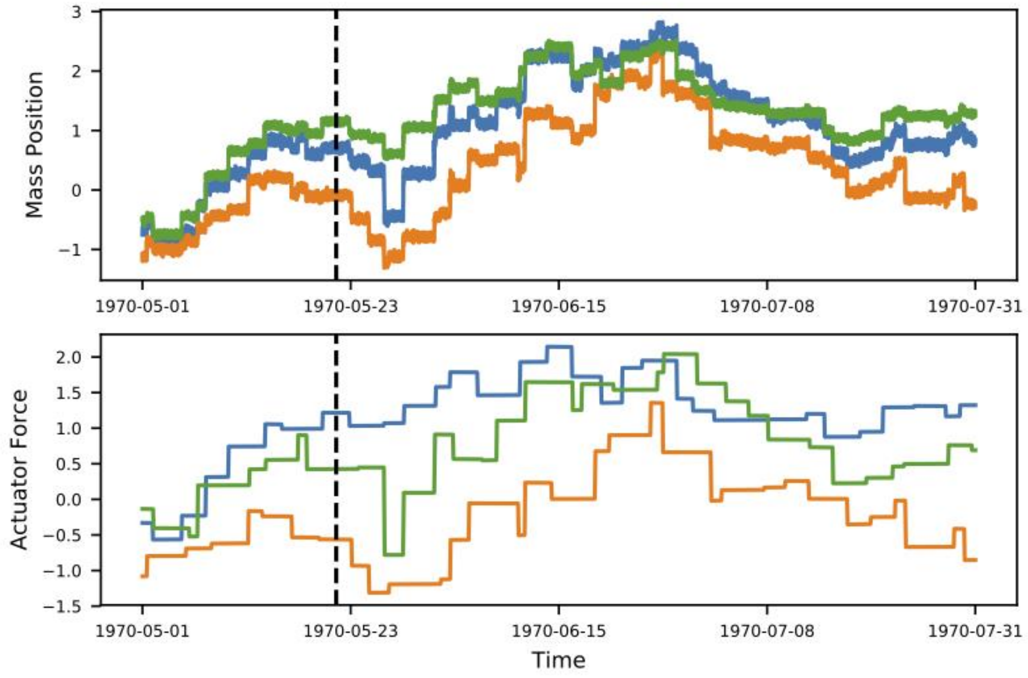

Figure 13. Simulation results for the three-mass system transitioning from normal to anomalous operations.

In addition to the position and force results, it is helpful to plot the magnitudes of the anomalies. These are the percent changes in the spring stiffnesses and damper coefficients that modify the underlying differential equations. Each anomaly is inserted as a ramp anomaly, meaning it starts from zero and increases to some maximum value (see Figure 14 for a plot of the magnitudes over time). Note that each anomaly has a different maximum magnitude and a different duration.

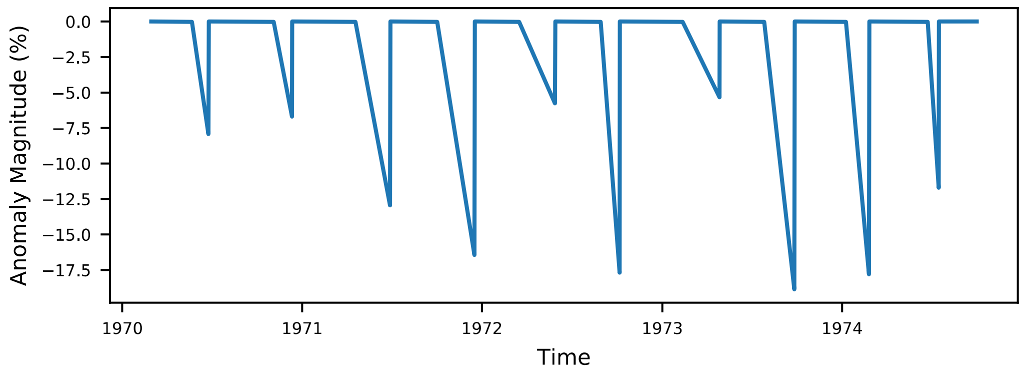

Figure 14. Plot showing the varying magnitudes and durations of each anomaly.

One important point regarding this system is that the "detectability" of anomalies is a function of two variables: the anomaly magnitude (mentioned above) and the spring displacement magnitudes. To understand this second variable, the differential equation for the one-mass system can be simplified by assuming the mass is as rest, then incorporate some anomaly-induced change to the stiffness. This results in:

$$(k + k_{anom})x = F \quad (2)$$

In this equation, the detection algorithms have access to noisy measurements of x and F in order to infer whether the spring stiffness is nominal or anomalous. However, when the value of x is close to zero (i.e., the magnitude of the mass displacement is close to zero), the effect of the anomaly on the system approaches zero. In other words, as the mass displacements shrink, the anomalous effect becomes smaller and therefore harder to detect. This means two important variables must be considered in order to determine the difficultly of detecting a given anomaly. Consideration of both of these variables plays an important role in explaining some of the results in later sections, and is also representative of NPP systems. For example, consider a case in which vibration data are used to detect anomalies in a pump motor. Even if the anomalous vibrations are apparent at normal running speeds, they may shrink or change patterns at slower pump speeds, potentially confounding an anomaly detection algorithm.

2.2 Dymola PWR Simulator

The second simulator generated virtual NPP data using the commercial Dymola-Modelica software. This software is used to build representative nuclear reactor models that employ control loops to regulate operation (e.g., a Proportional Integral Derivative controller is used to determine the optimal control actions, based on collected sensor data). The PWR model for simulating the system's transient behaviors is shown in Figure 15. The PWR consists of primary and secondary loops, the components of which are connected by blue solid lines. The reactor core, shown as the orange block on the lower left, has its inlet and outlet temperatures measured and then sent to the control systems, as represented by the red dashed line. The pressurizer, which is connected to the hot leg, features safety valves and a system-controlled pressurizer heater. The coolant in the hot leg flows into the steam generator and is pumped back to the core by the primary pump. For the secondary loop, the feedwater is pumped, heated, and consequently converted into steam in the boiler drum, then released into the atmosphere. The steam generation amount and feedwater flow rate were measured, and the results delivered to the control system for making actuation command calculations. The layout of the control system for a normal operation state is shown in Figure 16. The model used standard controllers (e.g., Proportional Integral Derivative and T_{ave}) to set the mass flow rate of the primary pump, the steam generator feedwater pump, the core reactivity, and the pressurizer's heater. Six different sensors were used as input parameters to the controller: total core power, coolant temperature of the core inlet and outlet, pressurizer pressure, steam generation amount, and feedwater mass flow rate. The results of the simulation were used to test the anomaly detection algorithm.

As with the SMD simulator discussed above, both the noise and anomalies were incorporated into the simulations. Noise was dynamically added to the control commands and sensor signals to simulate real behavior that included process and sensor noise. Anomalies were added in the form of corrupted sensor readings (e.g., level indicators and thermocouple readings) and/or control commands (e.g., valve alignment), with the resulting changes being subtle in comparison with the noise level.

Given the wide range of anomalies expected in an NPP, three different types of anomalies were used for this initial study: wide anomalies, which gradually develop over longer time periods and are then gradually removed; narrow anomalies, which are similar to wide anomalies but develop over shorter time periods; and persistent anomalies, which gradually develop but are not removed. The first two anomaly types represent situations in which the anomalies are discovered and removed by the regular maintenance work orders, whereas the third represents undiscovered anomalies.

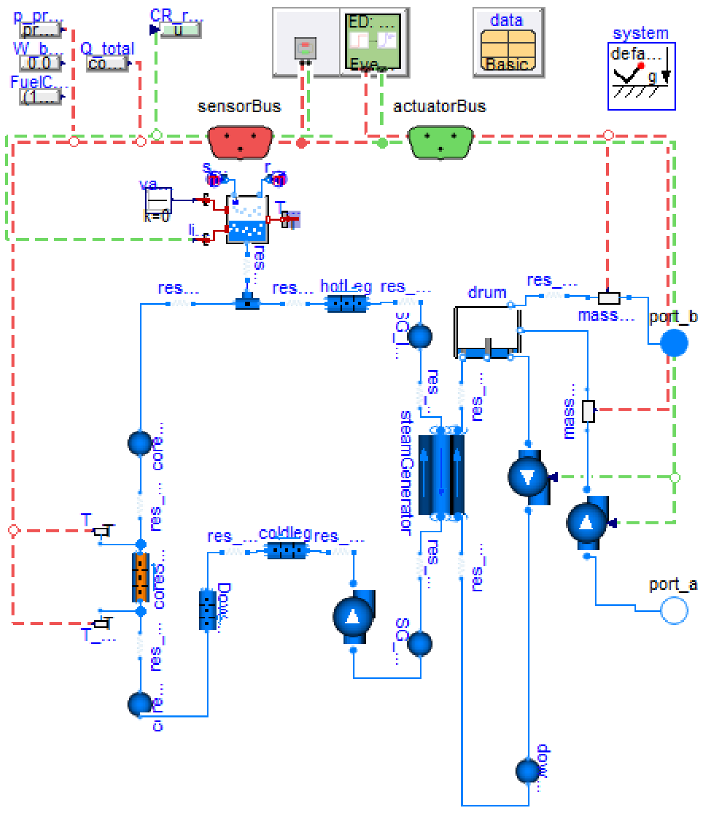

Figure 15. Layout of the Dymola-simulated PWR.[32]

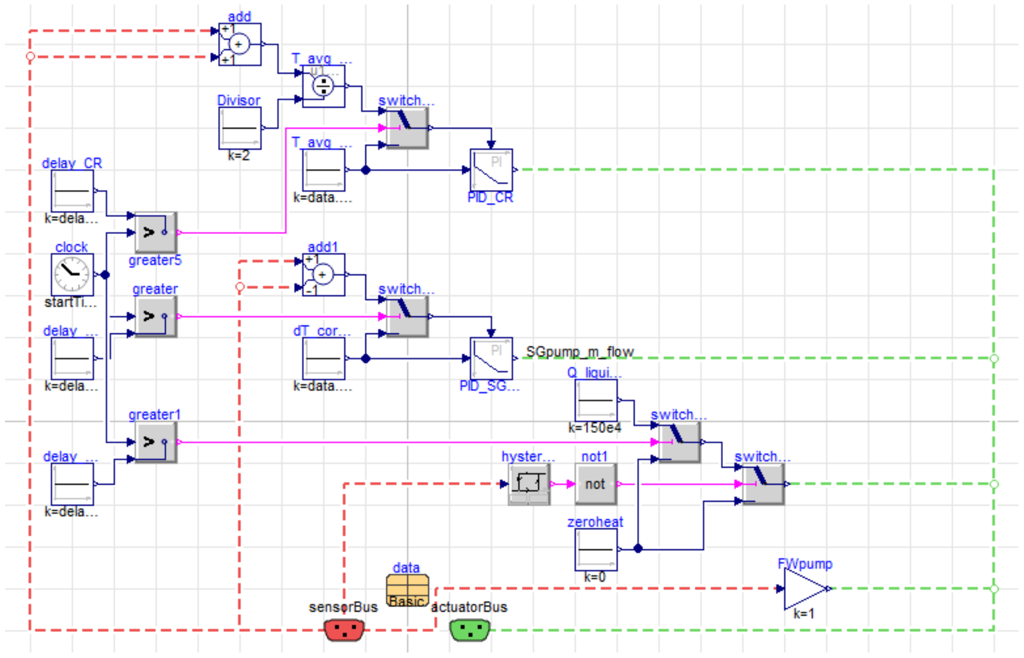

Figure 16. Control system layout.[32]

The selected wide anomalies were introduced in the primary loop core flow rate. To simulate noise, Gaussian noise of a magnitude representative of flowrate measurements in a typical NPP was added in a dynamic manner. This was achieved by adding the noise to the sensor readings before sending them to the controller. This ensured that the noise propagated dynamically throughout the actuated commands to the rest of the reactor, as shown in Figure 17. The temporal evolutions of different sensors are shown in Figure 18–Figure 20. Two simulations were completed, both with process noise, though one had no anomalies and the other had two wide anomalies. The simulation with no anomalies is shown for illustration purposes only and is not used by the anomaly detection algorithm. For the simulation with anomalies, one is introduced between 1000–2000[b] seconds and the other is introduced between 3000–4000 seconds. These graphs show that, when noise is present, the anomalies have little impact on the system behavior after they are removed, as would be expected during operation. For example, the first wide anomaly was gradually introduced at 1000 seconds and then completely removed by 2000 seconds. The normal behavior (blue) and anomalous behavior (orange) are indistinguishable over the ranges in which no anomalies were introduced. Similarly, the temporal evolutions of the total power for the narrow and persistent anomalies are shown in Figure 21 and Figure 22, respectively. For the persistent anomaly scenario, an additional narrow anomaly was added at 6000 seconds, and a persistent (i.e., not removed) anomaly was introduced at 8000 seconds.

[b] The time scale of the simulation is shorter than the typical time needed for anomalies to grow. This was not considered an issue, since the short time scale of simulation was proportional to the anomaly development rate, regardless of time scale.

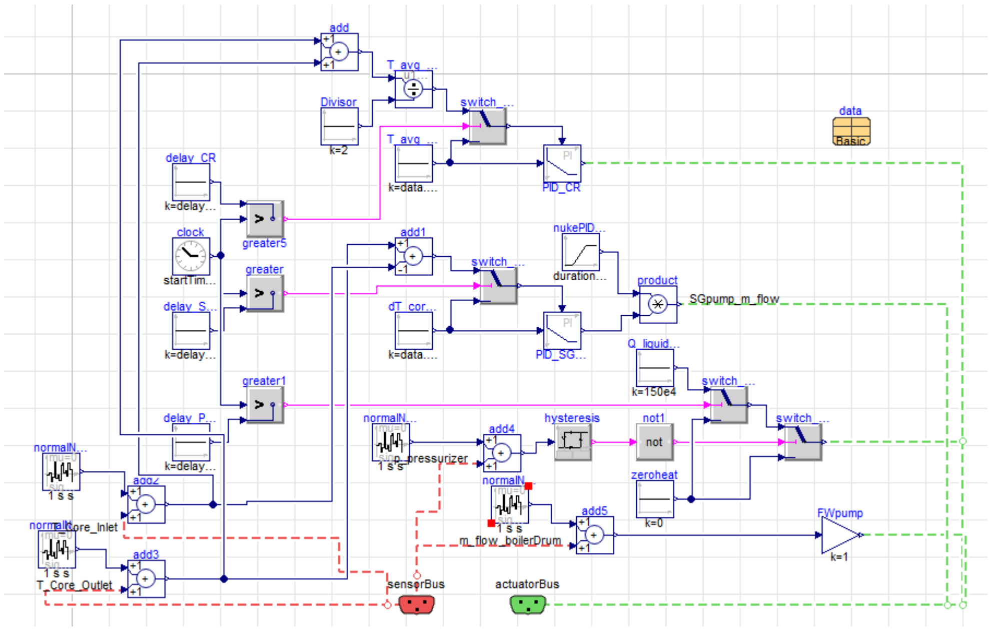

Figure 17. Control system layout of the anomalous primary loop.

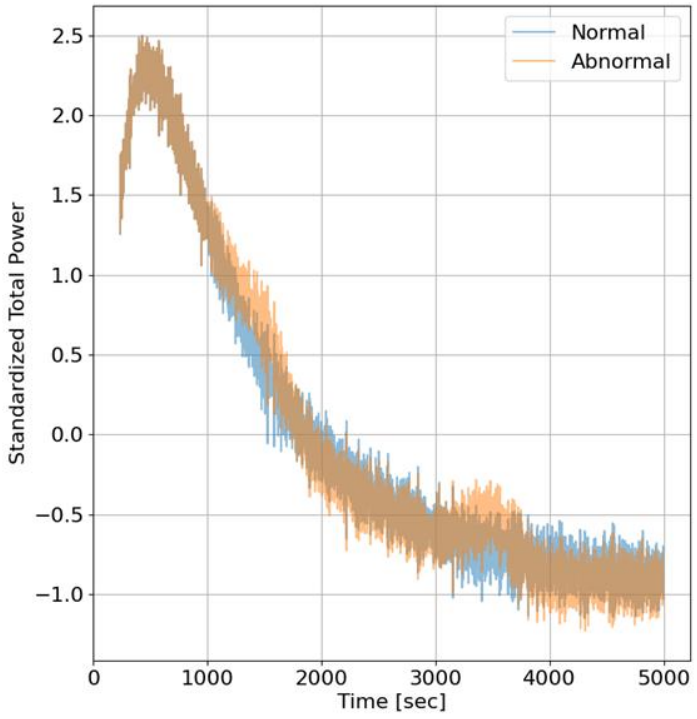

Figure 18. Standardized total power with two wide anomalies.

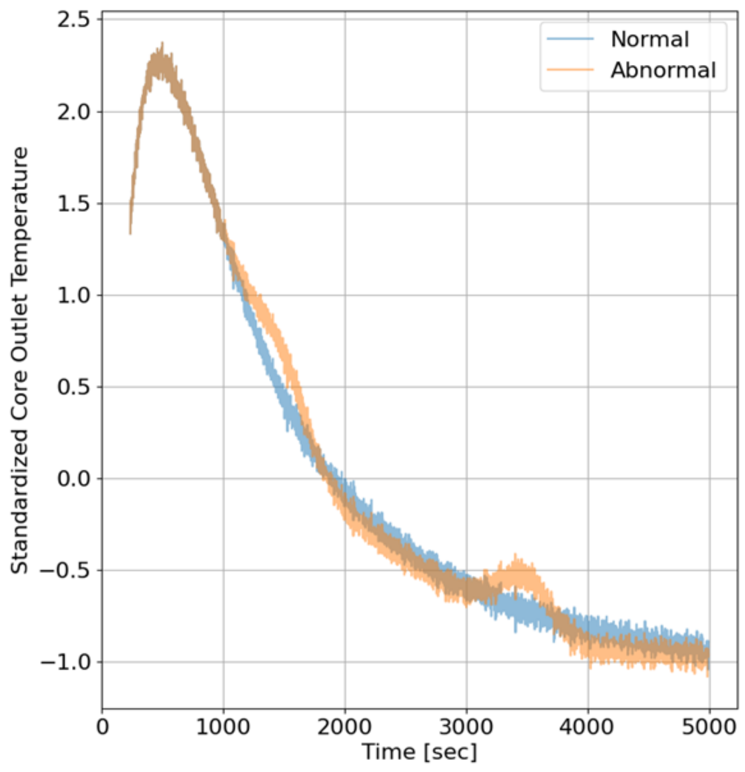

Figure 19. Standardized core outlet temperature with two wide anomalies.

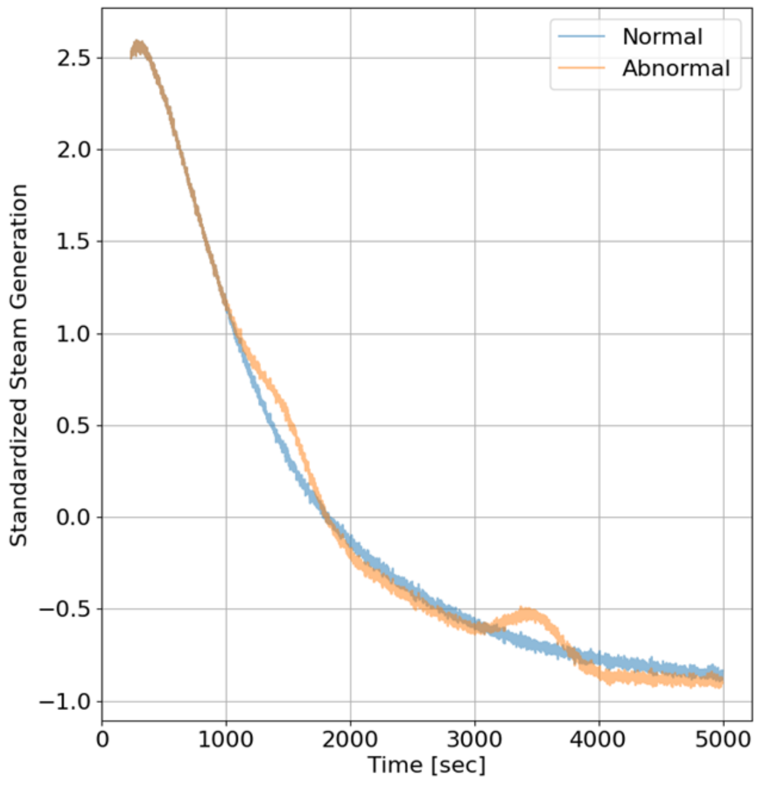

Figure 20. Standardized steam generation with two wide anomalies.

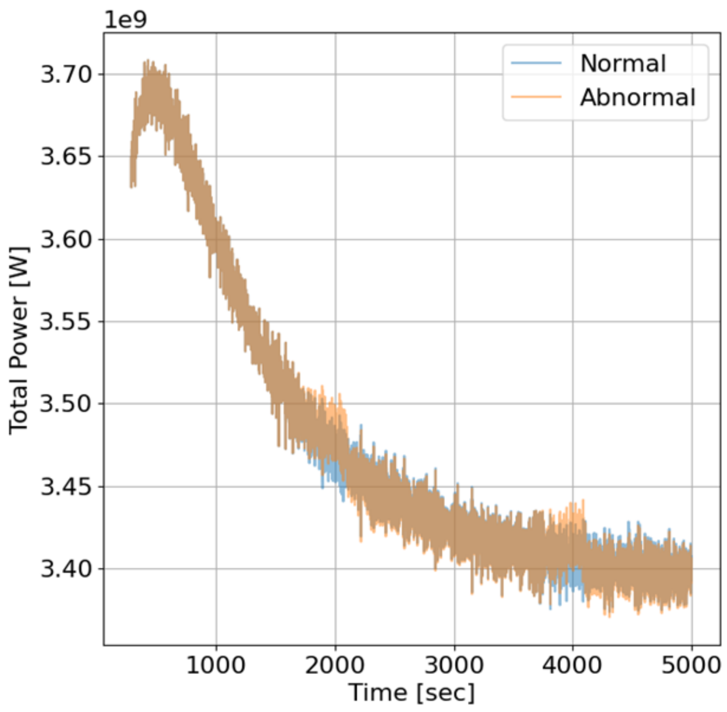

Figure 21. Abnormal scenario with two narrow anomalies.

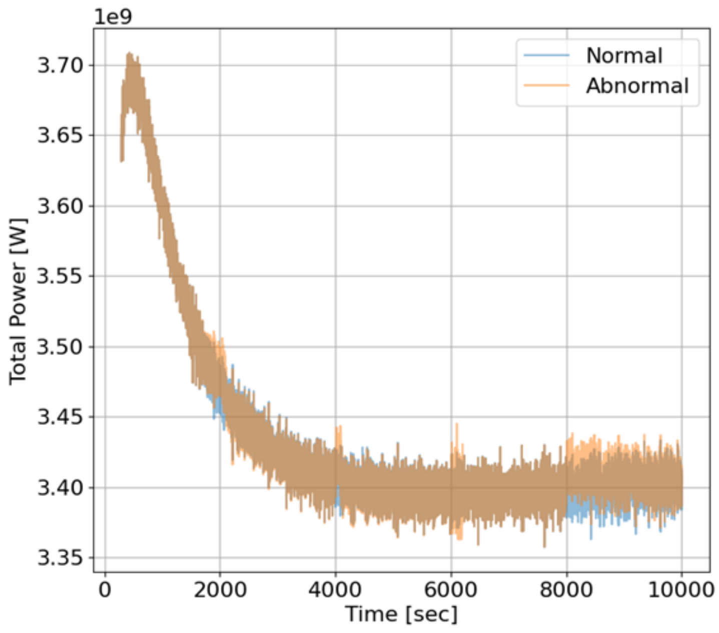

Figure 22. Abnormal scenario with three narrow anomalies and one persistent anomaly.

Furthermore, similar types of anomalies were implemented into another component: the steam generator feedwater pump. The previous analysis was repeated for all the narrow and persistent anomalies. The modified control system layout for this anomalous scenario is shown in Figure 23. The transient behavior of the feedwater flow rate under normal operation is shown in Figure 24, and the anomalies propagated via the control system are shown in Figure 25. Note that these figures do not show the included noise. The light blue curve represents the normal transient behaviors in both figures, whereas the orange line shows the transient behavior, with narrow anomalies occurring between 1800–2200, 3800–4200, and 6000–6200 seconds. The persistent anomaly was introduced at 8000 seconds.

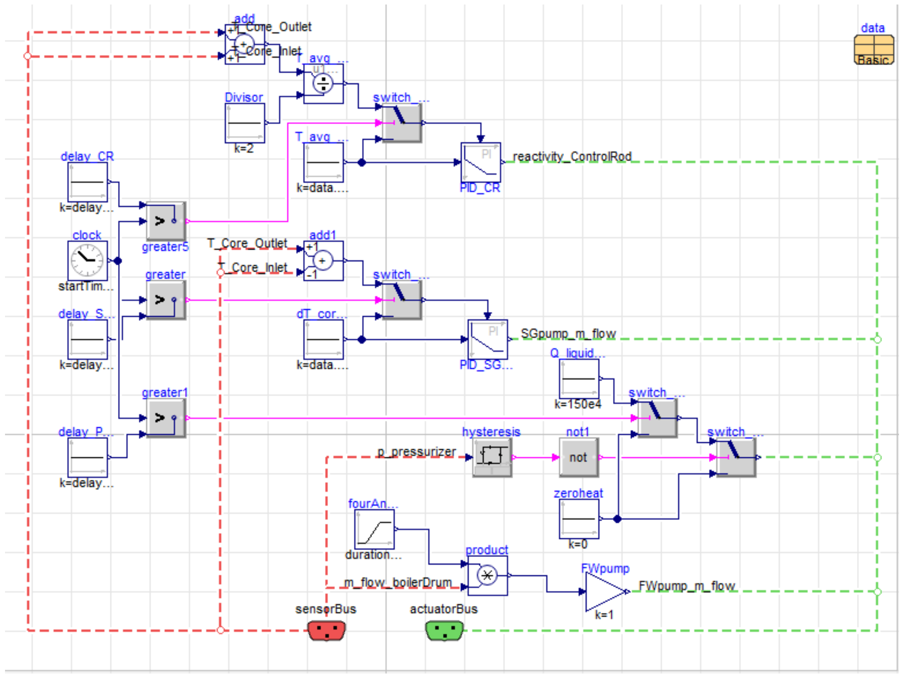

Figure 23. Control system layout of the anomalous secondary loop.

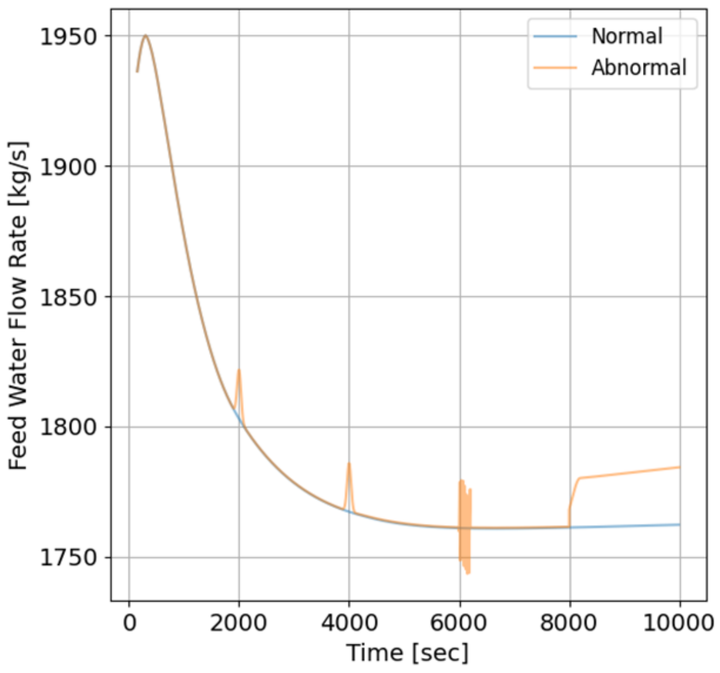

Figure 24. Anomalies in feed water flow rate.

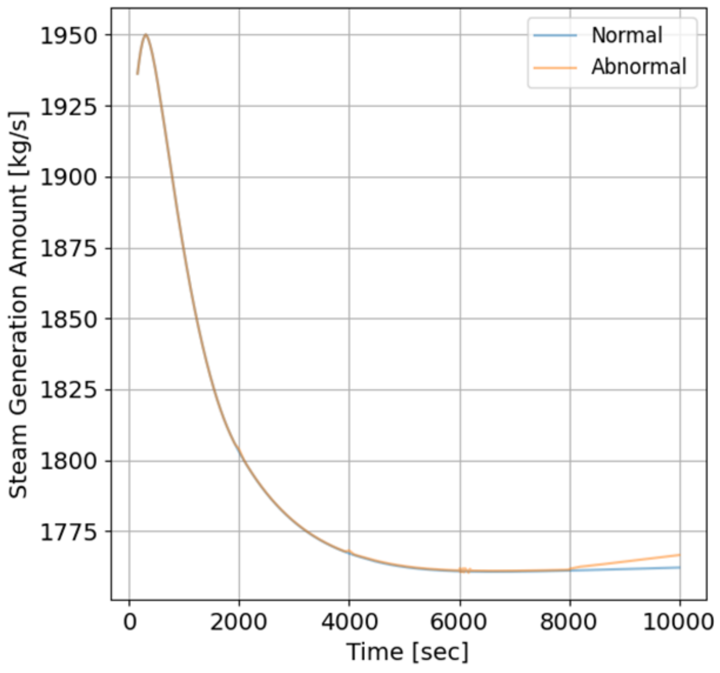

Figure 25. Anomalies propagated to the steam generation amount.

3. METHODS

In this report, two different methods of semi-supervised ML were used to fuse data from CRs with anomaly detection methods. As mentioned in Section 1, the ultimate objective is to test the hypothesis that, if partially labeled anomalies in a time series sensor dataset are used to train an ML model, that model would demonstrate superior anomaly detection performance (i.e., higher accuracy and fewer false positives) than if no labels were provided in the model training process. Two methods were used to test this hypothesis: the deep SAD method, which can handle labels ranging from fully unsupervised to fully supervised cases; and the HOF method developed specifically for this effort.

These two methods were selected because they both use features, though the HOF method creates them and deep SAD learns them. Another major difference between the two methods relates to the use of sparsely labeled data. The deep SAD approach actively uses the labels to develop a detector, whereas the HOF approach uses the labels passively. The deep SAD method actively learns the labeled anomalies by pushing them away from the center of the normal condition hypersphere (as discussed in the next section) while simultaneously minimizing the unlabeled distance from the center. In other words, the labeled anomalies directly influence the anomaly score function. By contrast, the HOF approach develops candidate features independent of the labels, then down selects those features that seemingly best distinguish the labeled anomalies. This approach is more passive, because the labeled anomalies do not directly influence the calculation of the features; instead, they are used purely to sort through the candidate features.

In the following sections, the deep SAD method was used to evaluate the aforementioned hypothesis, using a one- and a three-mass SMD system, whereas the HOF method was tested using both the three-mass SMD system and the Dymola-Modelica simulation datasets.

3.1 Deep Semi-supervised Anomaly Detection

3.1.1 Method Description

Deep SAD is an anomaly detection approach that can handle any range of labels, from fully unsupervised to fully supervised, though it is specifically intended for unsupervised and semi-supervised scenarios. The deep SAD method is outlined in this report; for full details, see the original work.[33] The basic concept is shown in Figure 26, as applied to a low-dimensional problem. On the left of the figure are the raw data in the input space, where the filled points are unlabeled and presumed normal and the unfilled points are anomalies. From this image, a nonlinear pattern can be observed for the normal points, and this pattern that could be used to separate them from the anomalies, though this may not be obvious to an unsupervised algorithm, particularly in a higher dimensional space. In the middle of the figure is the transformation $\phi(\cdot, w)$, which will be further defined later and represents some nonlinear function learned using the data. This function is usually based on some type of neural network (NN). Once applied, the data are transformed into a space in which the similar data from the input space have been mapped into a tight hypersphere around some center point, and the dissimilar data have been mapped further from the center (as shown on the right in Figure 26). This shape results from the algorithm's attempt to minimize the distance between all the points and the hypersphere's center point. Then, the points that fall within some user-defined radius are considered normal, and those that fall outside that radius are considered anomalous.

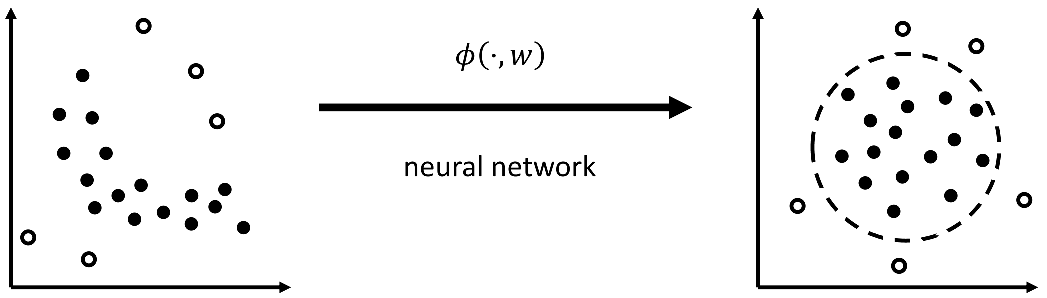

Figure 26. Conceptual drawing of the deep SAD approach.

The concept, as presented, describes an unsupervised ML process. To make it a semi-supervised ML process, the algorithm makes use of events that were presumed to have been labeled normal or anomalous. For events presumed to have been labeled normal, the algorithm has a user-defined parameter to weight the distance of normal points from the center. When this parameter is greater than one, it essentially makes the algorithm work harder to ensure that those known normal points are closer to the center. By contrast, for events presumed to have been labeled anomalous, this parameter still exists, but the algorithm also optimizes the inverse of the distance to the center of the hypersphere. In other words, it tries to push anomalies as far from the center as possible, while simultaneously pushing all the other points closer to the center.

The above concepts can be made more mathematical by looking at the objective function used in the deep SAD algorithm. The goal is to optimize some NN using the following objective function:

$$\min_{w} \frac{1}{n+m}\sum_{i=1}^{n}\|\phi(x_i; w) - c\|^2 + \frac{\eta}{n+m}\sum_{j=1}^{m}\left(\left\|\phi(\tilde{x}_j; w) - c\right\|^2\right)^{\tilde{y}_j} + \frac{\lambda}{2}\sum_{l=1}^{L}\left\|w^l\right\|_F^2 \tag{3}$$

where w is the vector of all NN weights, n is the number of unlabeled points, m is the number of labeled points, $\phi(\cdot, w)$ is the NN function, c is the hypersphere center, η is the user-defined parameter to weight labeled points, x_i are the unlabeled points, $\tilde{x}_j$ are the labeled points, $\tilde{y}_j$ are the labels equal to 1 if the point is normal and -1 if the point is anomalous, and the last term is a standard weight regularization term used in many common ML algorithms, with λ equal to the weight decay parameter. Comparing the objective function to the descriptions above, the first summation term minimizes the distance from the unlabeled points to the center, the second summation term both minimizes the distance from the points labeled normal to the center and maximizes the distance from the points labeled anomalous to the center, and the third summation term is a regularization term used to reduce model overfit.

One general challenge of using NNs with this objective function is their ability to define very complex functions. An NN with enough degrees of freedom will be able to map all points very close to the center. As such, several important steps are taken to prevent so-called hypersphere collapse. Some strategies employed in this report are discussed below; see the original paper for additional details.

To use deep SAD with time series data, a window approach was applied, meaning that each input to the NN contained the data for all relevant variables at multiple time steps. These windows can either be overlapping or adjacent to each other. Figure 27 demonstrates the window approach on a generic two-dimensional time series. For multi-variate time series data, the network input is of size n_t timesteps by n_v variables per window.

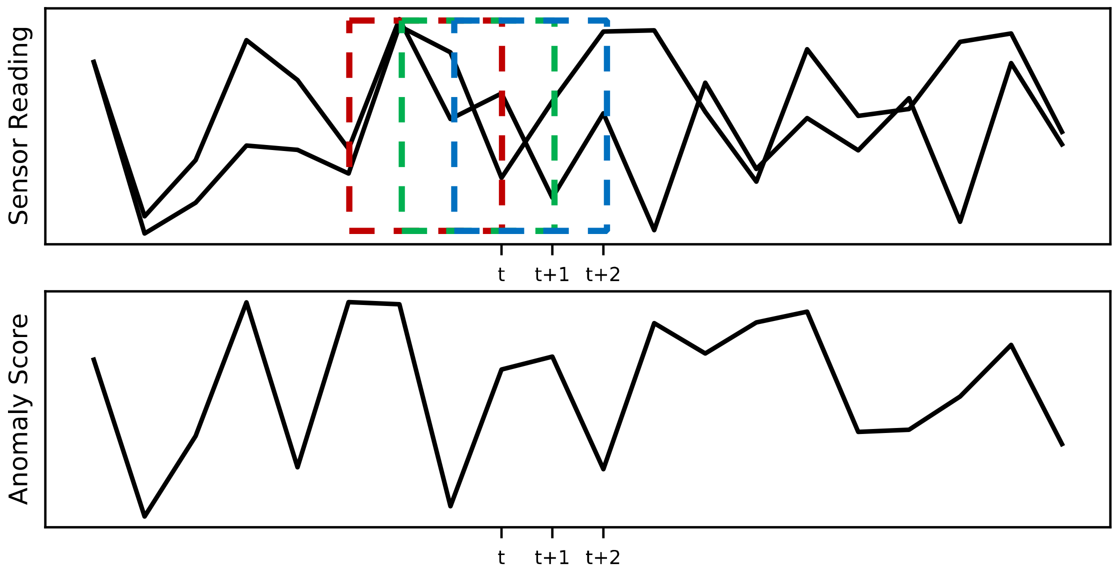

Figure 27. Visualization of the window approach to analyzing time series data. Each window (shown in different colors) includes the variables over a window of timesteps, as shown.

In this effort, convolutional neural networks (CNNs) are used as the function $\phi(\cdot, w)$ to handle the time series windows. CNNs take advantage of temporal and spatial dependencies in data in order to dramatically reduce the complexity of NNs. As such, they have seen significant use in image and video processing, as well as in time series analysis. A convolution operator slides a kernel of size n_k by n_v variables along the time series window, producing a new feature in accordance with:

$$S(k) = (X * W)(k) \overset{\text{def}}{=} \sum_{m=1}^{n_k} X(k-m)W(m) \tag{4}$$

where S is the generated feature map, X is the input to the convolution layer, W is the kernel weight matrix, $*$ is the convolution operator, and k indexes the time axis in the input. Using this convolution operator, a typical convolution layer consists of the convolution operator, a batch normalization, and a nonlinear activation function. The batch normalization is simply a normalizing layer to aid in training. For the activation function, the deep SAD algorithm suggests using the popular leaky Rectified Linear Unit function, so that is the function used in this effort. Combining all of this, a typical convolution layer consists of n_k kernels, batch normalization, and a leaky Rectified Linear Unit function. In addition, a network may consist of one or more convolution layers (Figure 28).

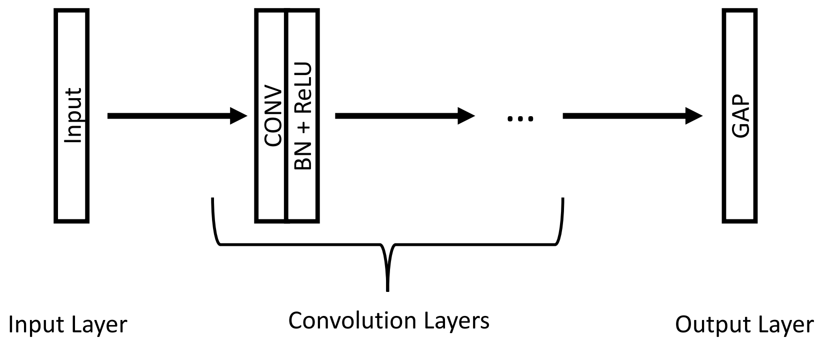

Figure 28. Architecture of the CNN used with deep SAD.

This work follows the CNN architecture discussed in[34] and combines convolution layers with a final global average pooling (GAP) layer. Prior to the GAP layer, the output of the final convolution layer has n_k feature maps that can each be thought of as a filtered time series specially crafted for its intended purpose (here, anomaly detection). Using a GAP layer, which calculates the average value over each feature map, these filtered time series are reduced to a useful lower-dimensional representation of the input window. This means the output of the GAP function is an n_k-dimensional array. This array is the output of $\phi(x; w)$ in the deep SAD algorithm. A schematic of the CNN is shown in Figure 28.

Combining CNN layers with a GAP layer offers several advantages over standard feedforward NNs (FNNs). First, the CNN layers use weight sharing to take advantage of the temporal dependence of time series data, thus reducing the complexity of the network and lowering the chances of overfit. This weight sharing is a result of the fixed kernel sliding over the window and reusing the same weights throughout the window; by contrast, an FNN would have separate weights for each value in the window, resulting in far more weights. Second, combining the CNN layers with a GAP layer means that the windows are invariant of start time. In other words, if the window start points were shifted over by a few steps, the network would analyze the data in an identical manner, making the network architecture robust to window placement. This statement stems from both the convolution operator and the GAP layer being invariant of start time. By contrast, an FNN could have different weights for both the start and end of a window, meaning it would analyze the beginning and end differently, making it more sensitive to window placement.

After the CNN architecture was designed, the network was trained using the objective function in Equation (3). This process involved solving for the weights that minimize the objective function. This is a standard process that uses a procedure called forward and backward propagation, along with stochastic optimization methods, to solve for the weights. These are standard procedures for most NN applications.

Once trained, the final step is to apply the NN for anomaly detection. As mentioned above, points that fall within some user-defined radius from the center are considered normal, and those that fall outside the radius are considered anomalous. Thus, distance from the center can be directly applied as an anomaly score, with smaller values indicating windows less likely to have anomalies, and larger values indicating windows that are more likely to have them. This anomaly score is used in the results shown in the next section.

To implement the deep SAD algorithm, this work uses an existing code that was made open source. The code was modified to accommodate this effort's datasets and network architecture.[c]

3.1.2 SMD Results

The deep SAD approach was applied to the one- and three-mass datasets described in Section 2.1. For each dataset, the cases analyzed were the unsupervised case, semi-supervised cases with varying numbers of labels, and the fully supervised case. When presenting results, an anomaly is successfully detected if the anomaly score rises above the determined threshold before the anomaly ends, even if there is some delay before the anomaly is detected. In addition, a false alarm occurs if the score rises above the threshold during normal operation.

In both the one- and three-mass datasets, 10 anomalies occurred over the course of five years. The first four years are used for training and the last is used as a test set. However, anomaly scores are given for the entire five years to show how the method works, even on the training data. This is particularly relevant for the unlabeled anomalies, since they are still trained to minimize the anomaly scores.

[c] The code is available under the MIT license by the authors of Reference 33. The original source code is available at https://github.com/lukasruff/Deep-SAD-PyTorch. The deep SAD algorithm was written using the PyTorch Python library, an open-source ML library designed for deep learning applications.

The threshold is calculated by plotting the number of detected events as a function of varying threshold. The selected threshold will be the smallest value at which the number of detected events plateaus. This concept is shown for each case, with the selected threshold identified. The results are shown in Figure 29 for the unsupervised, semi-supervised, and supervised cases, each of which featured its own optimal detection threshold. In addition to the number of events detected, this figure also shows the slope of the number of events detected using a smoothing filter. The plateau in the number of detected events is calculated as the first candidate value at which the smoothed slope reaches zero, and is marked on each plot in the figure. One takeaway from these plots is that they enable an estimation of the boundary between the majority of the events and the handful of larger events that get marked as anomalies.

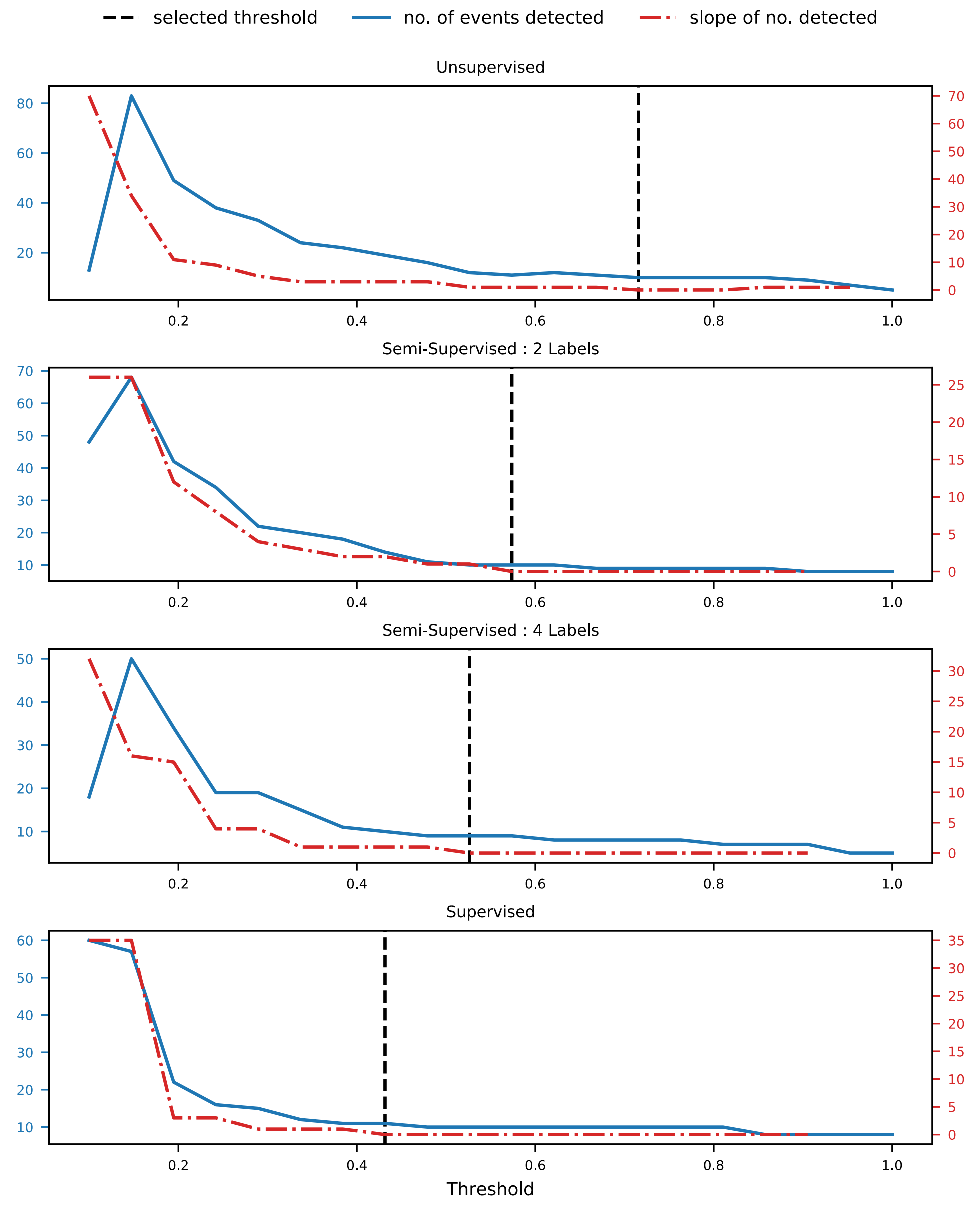

Figure 29. Candidate decision thresholds for the one-mass system.

Next, the anomaly scores are calculated for each of the cases, as shown in Figure 30. In the figure, the blue shaded regions mark unlabeled anomalies, red shaded regions mark labeled anomalies, the red horizontal line marks the pre-determined threshold, and the black vertical line differentiates the training and test sets. Starting with the unsupervised case (top subplot), the detector does catch many of the anomalies, including those in the test set. However, there are also some large anomaly scores during normal operation that result in false alarms. Moving down the plot, as the algorithm trains on more labeled anomalies, the general trends are that (1) the anomaly scores become more dominant and (2) the previous false alarm scores get progressively smaller up until the fully supervised (bottom subplot) case, which detects all the anomalies. Overall, the deep SAD algorithm was very successful in detecting anomalies in this dataset.

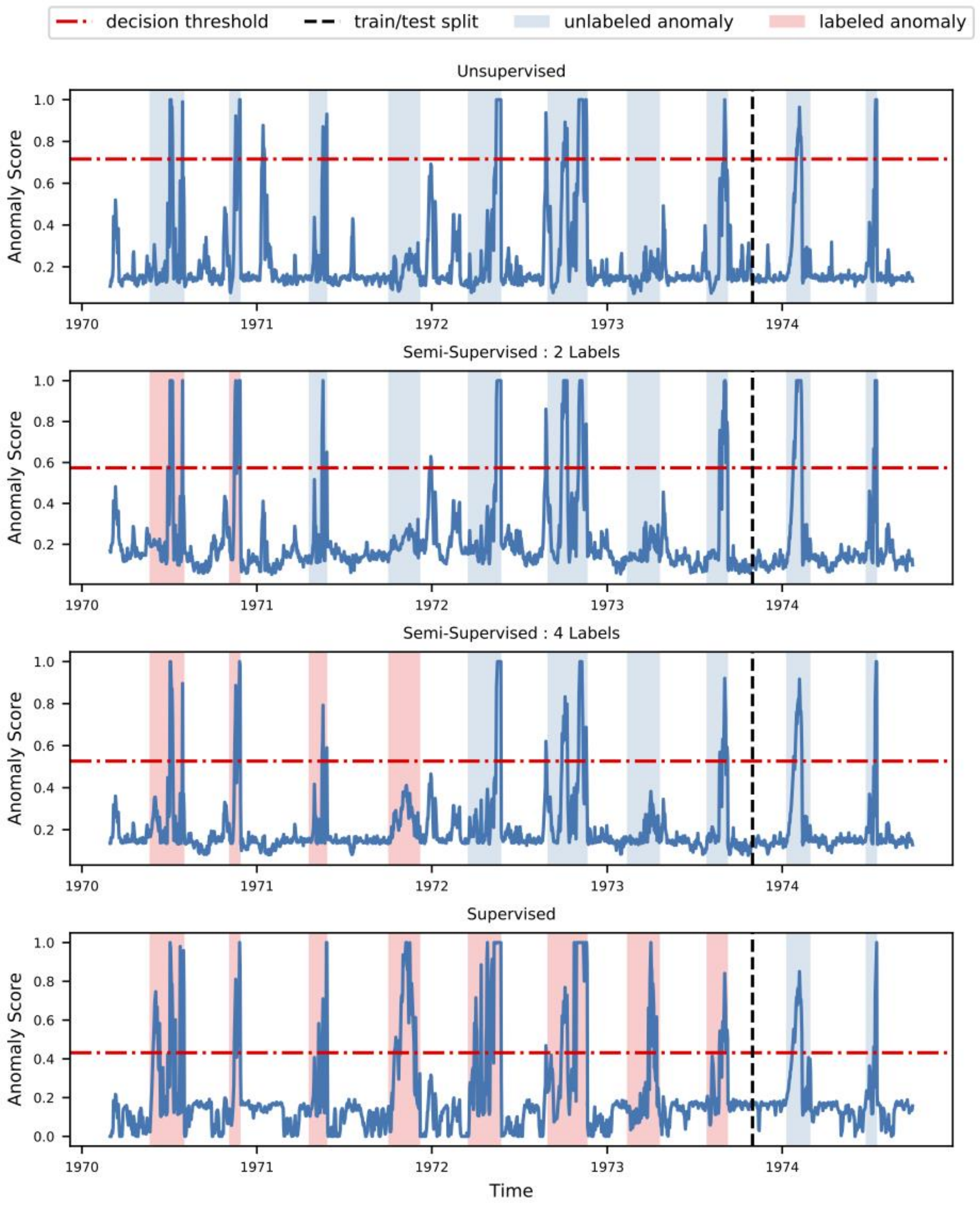

Figure 30. Anomaly scores for the one-mass system.

For the one-mass dataset shown here, the labeled anomalies help the anomaly detection algorithm in two distinct ways. First, they improve the detector's ability to assign high scores to true anomalies. Consider, for example, the fourth anomaly, which is undetected in the unsupervised case, increases in the semi-supervised cases, and is easily detected in the fully supervised case. Second, they decrease the anomaly scores that the detector assigns to periods of normal operation. This is clearly shown for the peak in the unsupervised case at the start of 1971. As more labels are included, the peak decreases until becoming indistinguishable in the fully supervised case. There are several other peaks in the unsupervised case that slowly disappear as additional anomalies are labeled. Both benefits of labeled anomalies improve the detector's ability to increase true positives and decrease false negatives.

As with the one-mass system, the threshold for the three-mass case is calculated by plotting the number of events detected as a function of varying threshold. These results are shown in Figure 31 for the unsupervised, semi-supervised, and supervised cases. The conclusions from this plot are very similar to those reached when using the one-mass dataset.

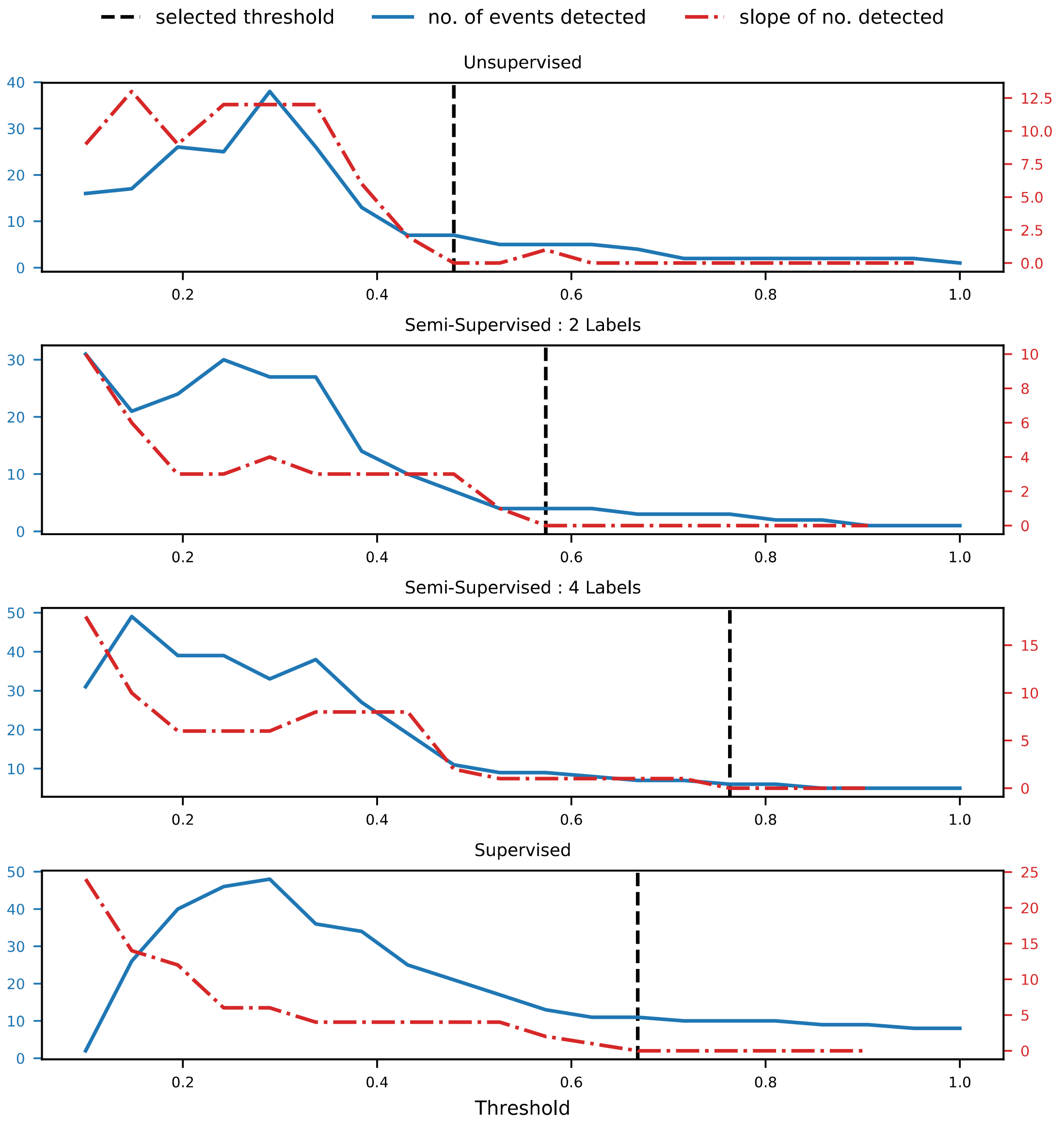

Figure 31. Candidate decision thresholds for the three-mass system.

Next, the anomaly scores are calculated for each of the cases. Figure 32 shows the results, which are not as clear as for the one-mass dataset. For the unsupervised case, the detector catches many of the anomalies. However, the second case, featuring just two labeled anomalies, seemingly performs worse than the unsupervised case. But, by adding more labels to the fully supervised case, the detector catches almost all the anomalies. The only exception in the fully supervised case is the second anomaly, which was a particularly difficult anomaly to detect; this is discussed in more detail below.

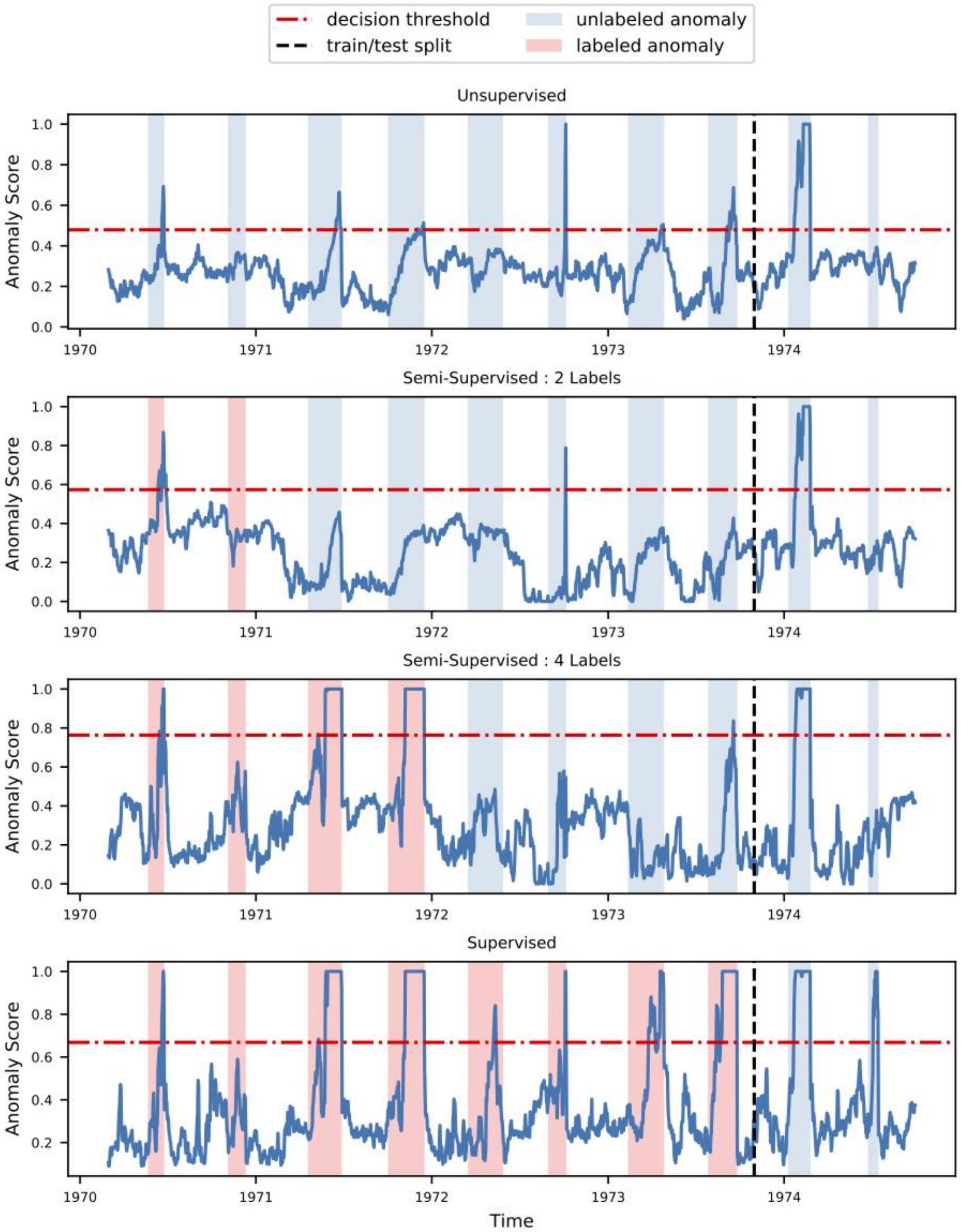

Figure 32. Anomaly scores for the three-mass system.

Looking at the results for the three-mass system, the second anomaly is difficult to detect in all four cases. To better understand this, it is helpful to remember the two factors that affect the detectability of such anomalies: anomaly magnitude and the mass displacement magnitude. Looking at the anomaly magnitudes from Figure 14, the magnitude of the second anomaly is comparable to those of several other anomalies, so this is unlikely the problem. However, looking at the mass displacements in Figure 33, they are small for the duration of the second anomaly, as compared with other anomalies. This means that the measurable effects of the second anomaly are smaller than for other anomalies, making it more difficult to detect.

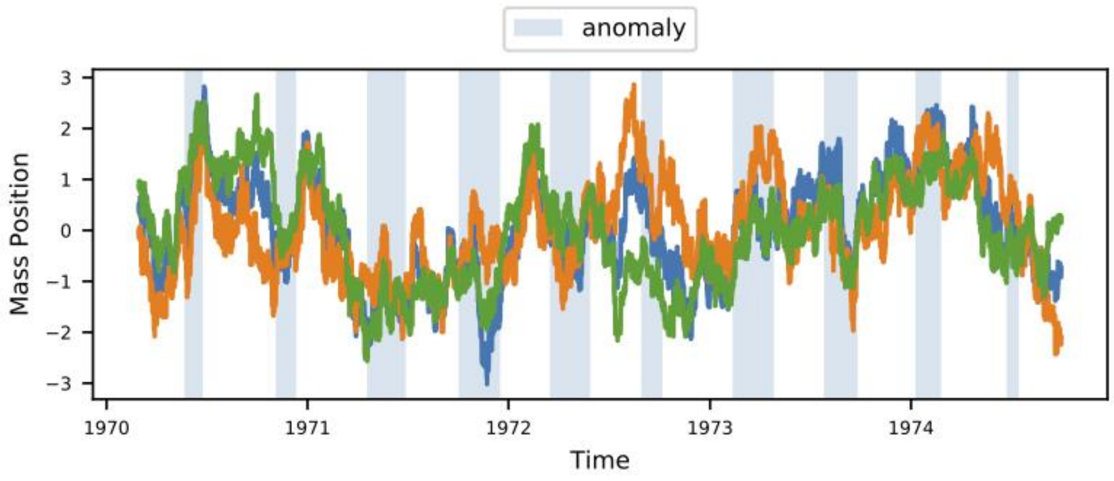

Figure 33. Mass displacements highlighted for the second anomaly in the three-mass dataset.

There are several reasons why this three-mass dataset may have performed worse than the one-mass dataset when using the deep SAD approach. First, it is a more complicated dataset with more variables and more sophisticated interdependencies; therefore, it is intuitively a more challenging problem. Second—and possibly more important—there are more possible types of anomalies, making it harder to generalize from the known examples. To better explain this, the one-mass system only has a single failure mechanism: failure of the single spring and damper. Thus, given a few labels, these labeled instances are indicative of all possible failure modes, making it easier to generalize to new unseen. By contrast, the three-mass system has three failure mechanisms, one for each mass. As a result, when given a few labels, these may not be indicative of all possible failure modes, thereby increasing the difficulty in generalizing to unseen anomalies.

3.2 HOF-Based Anomaly Detection

3.2.1 Method Description

This work features a new semi-supervised ML algorithm, which encompasses three functions:

1. Uses unsupervised decomposition techniques to identify a set of candidate features denoted by HOFs.
2. Ranks which HOFs are most sensitive to the available labeled anomalies.
3. Acts as a standard supervised or unsupervised ML algorithm in order to design a classifier.

To describe these steps, a brief summary is presented of how general anomaly detection algorithms work. The core idea is to split the signal into two components: a regular pattern and an unexplained residual. The regular pattern represents the structured variations in the data and is considered to represent normal behavior. The residual part is assumed to be unexplained (i.e., without structure and attributable to random noise). Mathematically, the regular pattern is identified by minimizing the norm of the unexplained residual using a distance-type metric (e.g., least-squares minimization). The idea behind this approach is that, during normal operation, the regular pattern will manifest itself and the unexplained part will remain approximately bounded within some norm calculated during the training phase. Consider, for example, a split of the following form:

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + a_3 x_{t-3} + \epsilon_t \tag{5}$$

This linear estimation implies that the value of variable x at time t is approximately determined during normal operations by the weighted sum of the last three lagged values of x—with the weights being constant (i.e., time independent) in this simple example—and the residual ϵ represents the unexplained part of the signal. In this context, the three-component vector containing these weights is referred to as a feature vector. The implication is that the predicted value of x at time t is simply the inner product (i.e., projection) of the three lagged values with the feature vector:

$$x_t = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ x_{t-2} \\ x_{t-3} \end{bmatrix} + \epsilon_t \tag{6}$$

The idea of feature extraction is general: it applies to simple linear models such as the one above, or to general nonlinear functions such as those trained by deep NNs, wherein the feature vectors are the weight vectors associated with the various neuron functions, i.e., $g_i(w_i^T x)$, which can be described by a sigmoid function or other activation function.

An effective anomaly classifier is one that can identify the features most sensitive to the sources of the anomalies, rather than the dominant (i.e., normal) behavior. A prominent approach for extracting the dominant features from time series data is the windowed Singular Value Decomposition (SVD) approach, which has been employed as a basic ingredient in many feature extraction algorithms such as Principal Component Analysis, Dynamic Mode Decomposition, and Proper Orthogonal Decomposition. [35,36] These techniques construct a matrix X of snapshots whose columns contain the time series data over the selected window, with the number of columns representing the total number of snapshots. Next, a rank-revealing decomposition is performed, producing X = USV^T, where U is a matrix of left singular vectors, S is a rectangular diagonal matrix containing the singular values, and V is a matrix of right singular vectors. The matrix U is *k* x *k*, where *k* is the window size. The first *r* columns of U represent the low-order components, which correspond to the dominant features in the time series. The remaining *k–r* components (hereafter denoted as the HO components) are noise-dominated over the range of normal behavior.

To detect anomalies, the first step of this proposed semi-supervised ML algorithm employs a randomized-window-placing SVD algorithm called Randomized Window Decomposition (RWD)[35,36] to identify the HOFs that are orthogonal in the L_2 sense to the low-order (i.e., dominant) features. These HOFs are similar to a matrix's null space components, denoting the directions in the input space that are multiplied by zero in a matrix-based linear transformation. Similarly, the RWD-extracted HOFs are expected to be zero when no noise exists. Under normal operation, they are expected to have the same characteristics as normal process noise. This makes them ideal for detecting subtle variations because, when an anomalous behavior develops, both the low-order and HO features are generally expected to be impacted—with the latter expected to be more sensitive, since its pre-anomalous values are well characterized. If the normal process noise is well understood, it is possible to employ a denoising algorithm to preserve the HOFs' sensitivity to the subtle anomalies.

The idea of using information contained in the HOFs is not new; previous work has looked at this for anomaly detection, although not for semi-supervised ML.[37] In this earlier work, the authors called the features null-space features. A primary difference between this earlier work and the current effort is that the earlier work used the Euclidean norm of all the noise-dominated null-space features. However, the Euclidean norm is known to dampen the impact of outliers, which may make it less sensitive to subtle variations. To overcome that, the method developed here analyzes the HOFs individually as indicators for subtle variations. One additional potential benefit of this is that it may be possible to tie a given anomalous source to a particular HOF, providing a strategy for both anomalous behavior detection as well as classification, although more work is needed to demonstrate this.

With access to the known past anomalies encoded in the algorithm, the second step of the HOF algorithm selects those HOFs that show the highest sensitivities to the labeled anomalies. Finally, in the third step of the algorithm, the selected HOFs are employed to train a classifier with the labeled anomalies. A detailed description of the three steps of the HOF algorithm is provided below:

Step 1: Identification of HOFs using the RWD algorithm

In conjunction with the RWD algorithm, a previously developed denoising algorithm[36] is employed to smooth out the noise expected in operational data. This denoising algorithm employs multi-level passes of the RWD algorithm over the original time series data and features extracted therefrom, ensuring that all features in the data are preserved, both low- and high-order.

After denoising, the RWD algorithm, which uses SVD, identifies the low-order features during normal operation. Because the window-placing is random, the extracted low-order features are insensitive to the presence of rare anomalies. This is important because, in practical applications, most anomalies are unlabeled. After the low-order features are extracted, the HOFs are defined as the higher-order components of the SVD decomposition (i.e., those associated with low singular values) and, when plotted vs. time, display random behavior akin to noise. If the window size is k, and r represents the number of dominant components, this algorithm produces k–r possible HOFs to serve as candidates for identifying anomalous behavior.

Step 2: Ranking of HOFs using labeled anomalies

This step represents the key aspect of the semi-supervised algorithm, as it employs the known labeled anomalies to down select those HOFs deemed most sensitive to the labeled anomalies. In this initial study, a simple metric was used to rank the HOFs extracted in the first step, measuring the ratio of the peak value to the noise level. Examples are shown in Figure 34–Figure 36, representing different HOFs with different levels of sensitivity to the anomalous data[d]. The red dots represent the value of a given HOF in the anomalous range, and the blue represents the corresponding value over the normal range. Figure 34 indicates that the particular HOF is not sensitive to the anomalous behavior, hence it is discarded from the candidate set of HOFs selected in Step 1. The HOF in Figure 35 demonstrates a moderate level of sensitivity to the anomalous behavior, and Figure 36 shows the highest level of sensitivity identified in the candidate set.

Selection of the HOF that exhibits the highest sensitivity to the anomalous behavior represents the best feature set to use in a classifier. One idea is to fuse multiple features via multiplication or linear superposition of their corresponding profiles, making the anomalies' impact more pronounced. Given that the number of features is small, one can attempt simple fusion ideas such as forming a set of binary-fused features (i.e., two at a time), tertiary features, etc., then applying the proposed metric to identify the binary set with the highest sensitivity. Figure 37 shows an example resulting from fusing one HOF with one dominant feature. In our study, this fused feature is used for classification in the next step.

[d] Note that at the tops of these figures, the window size is given in number of samples.

Step 3: Classifier Training

With the identified candidate HOFs from Step 2, one could use a wide range of unsupervised ML methods (e.g., clustering) or supervised ML methods (e.g., SVM-based classification). This step is not shown in this report.

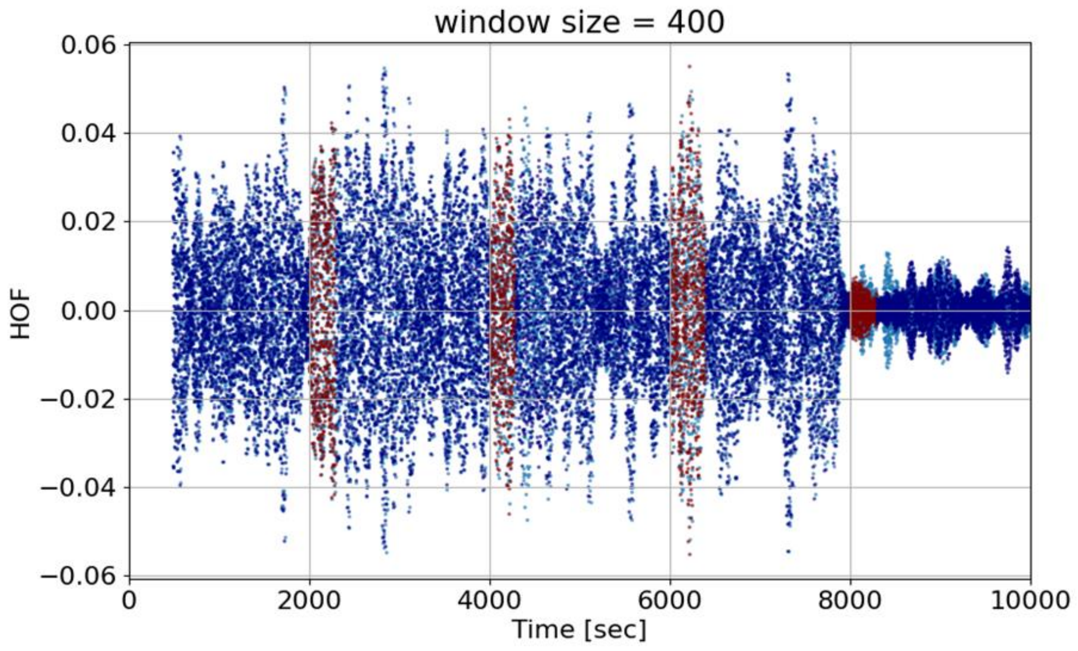

Figure 34. HOF without sensitivity.

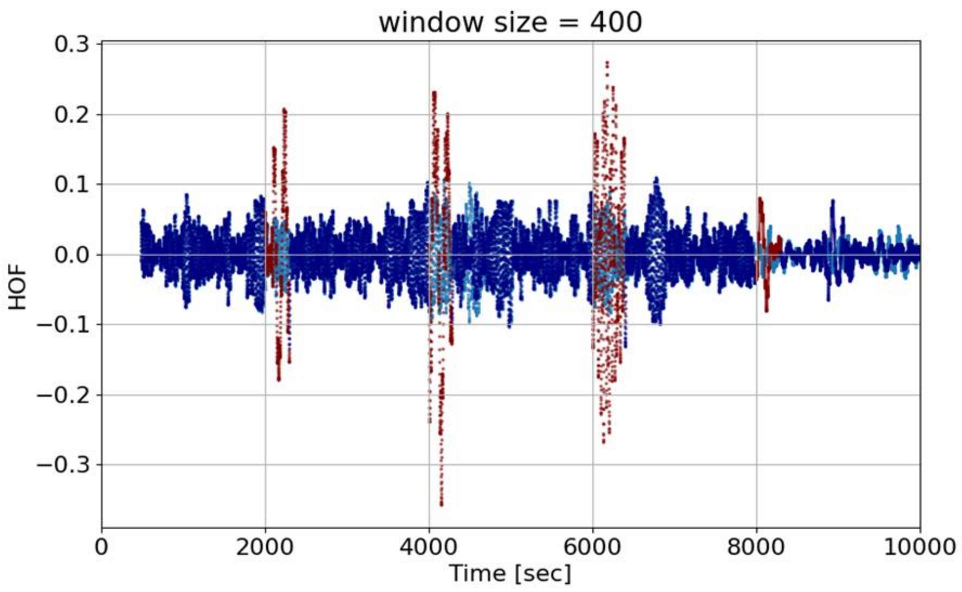

Figure 35. HOF with moderate sensitivity.

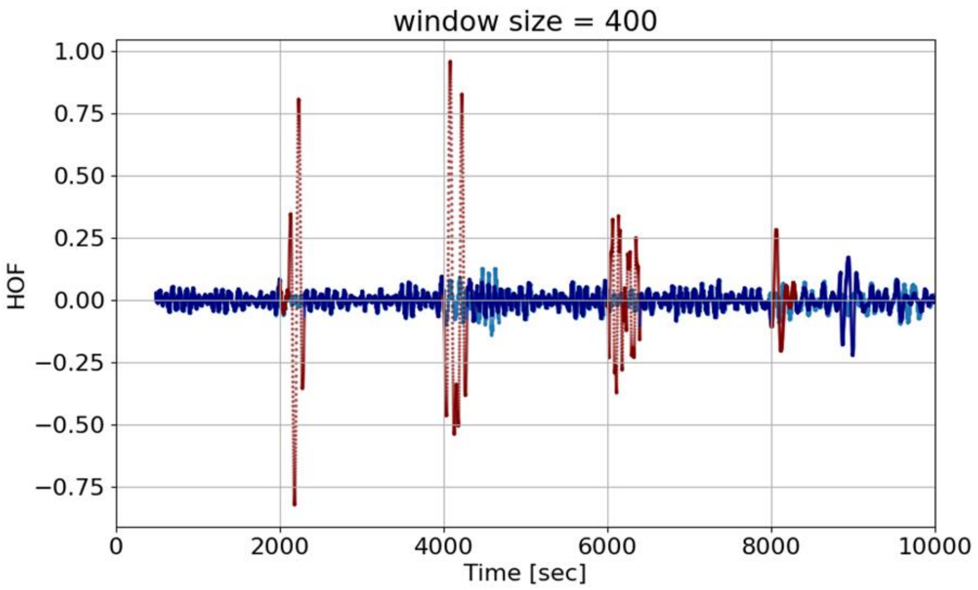

Figure 36. HOF with high sensitivity.

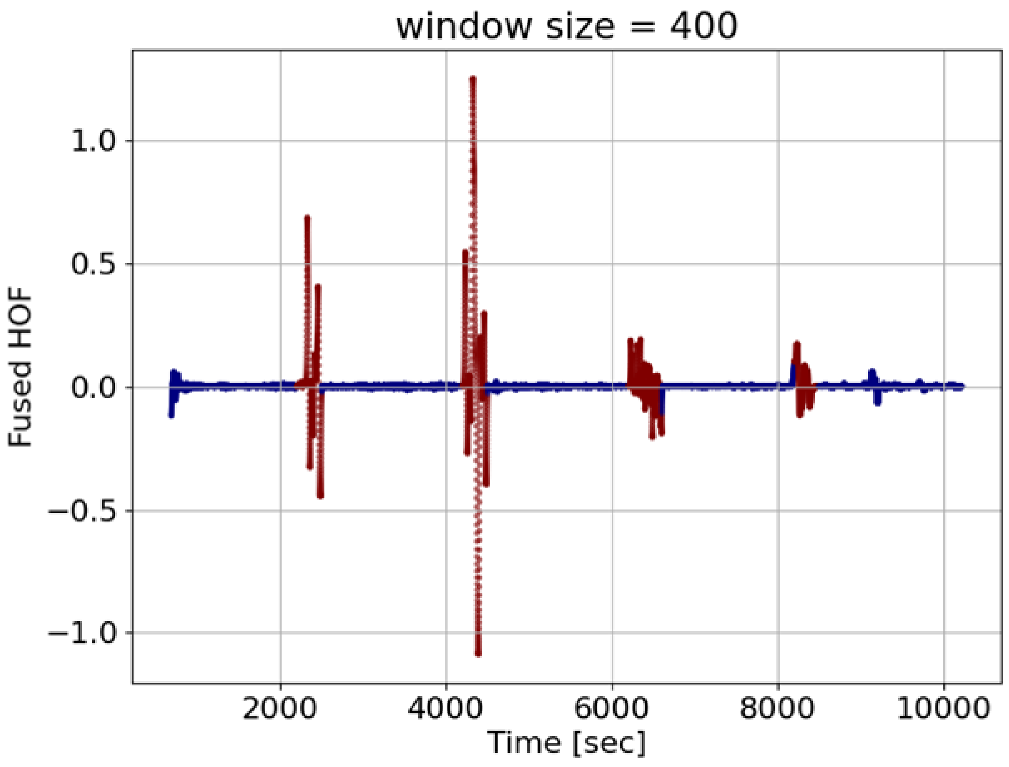

Figure 37. Fused HOF.

3.2.2 Dymola Results

The HOF anomaly detection approach was applied to the Dymola dataset described in Section 2.2. The first set of results are for the wide anomalies. In the first step of the proposed algorithm, the RWD

HOFs are extracted and plotted vs. time (see Figure 38–Figure 41), showing different levels of sensitivity to the anomalies. Figure 40 and Figure 41 show the features with the maximum sensitivity, and these features could be employed to train the SVM classifier. The classification is based on a label of the first anomaly only, with the results indicating that the second anomaly is perfectly visible to the classifier.

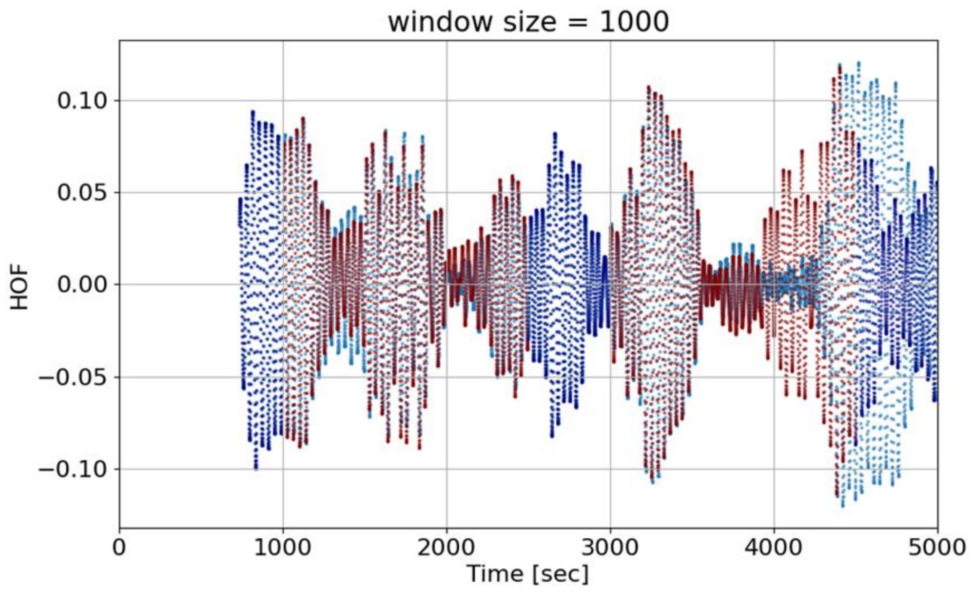

Figure 38. HOF with low sensitivity for wide anomalies.

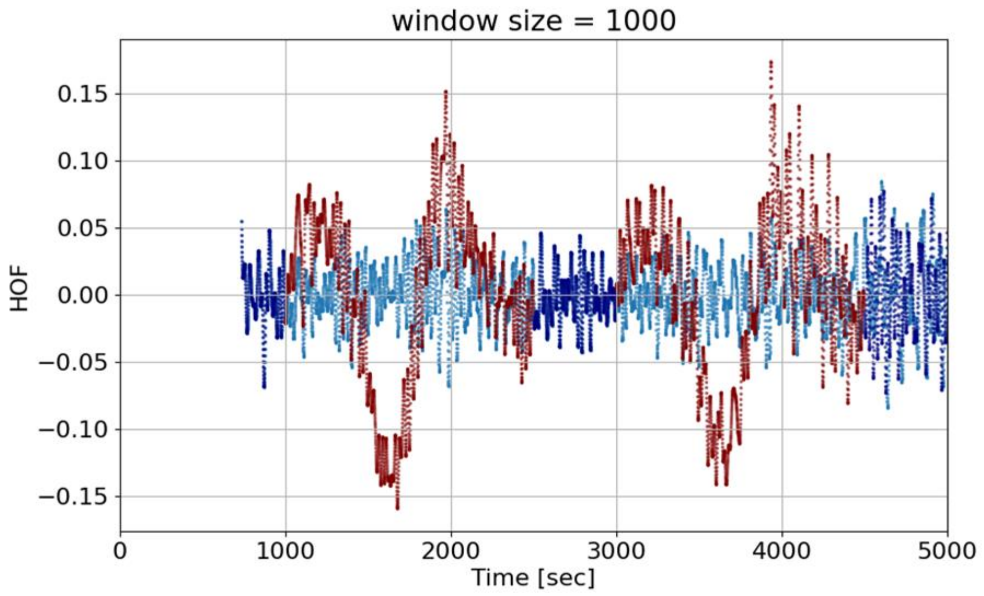

Figure 39. HOF with moderate sensitivity for wide anomalies.

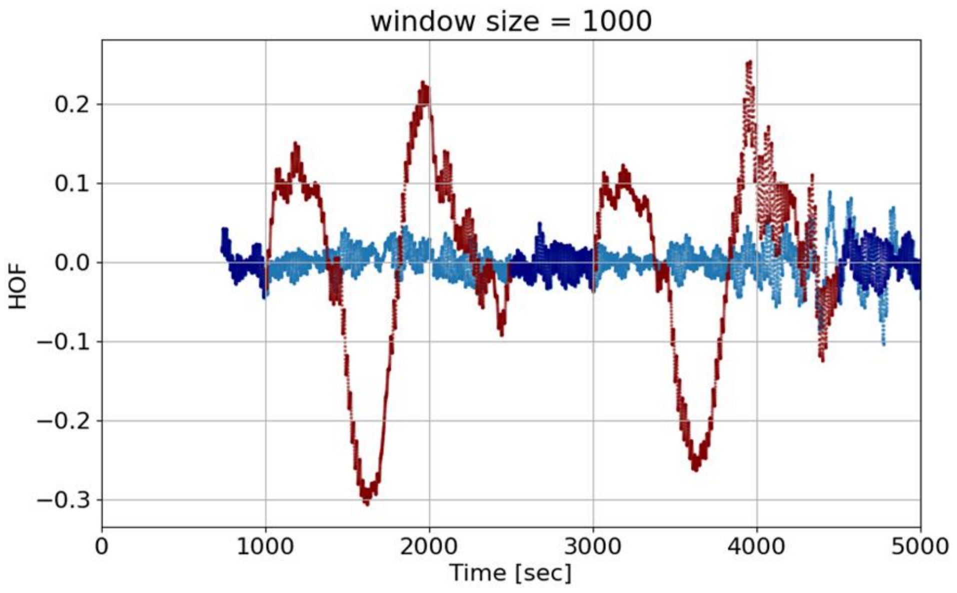

Figure 40. HOF with high sensitivity for wide anomalies.

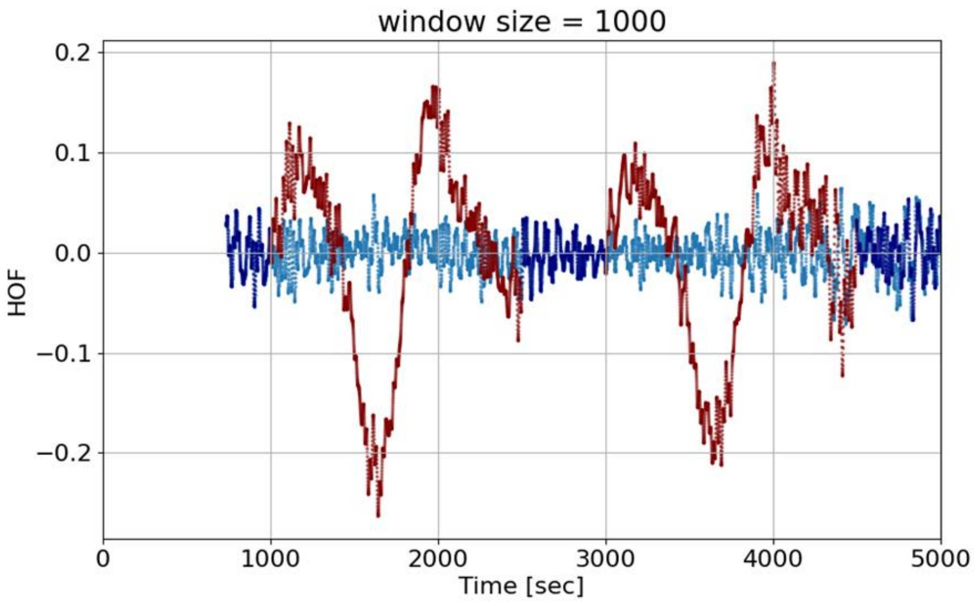

Figure 41. A second HOF with high sensitivity for wide anomalies.

Figure 42–Figure 45 show representative HOFs similar to those for the case of two narrow anomalies. The features in Figure 44 and Figure 45 are used for classification, and the results show the effectiveness of the selected HOFs in detecting subtle narrow anomalies.

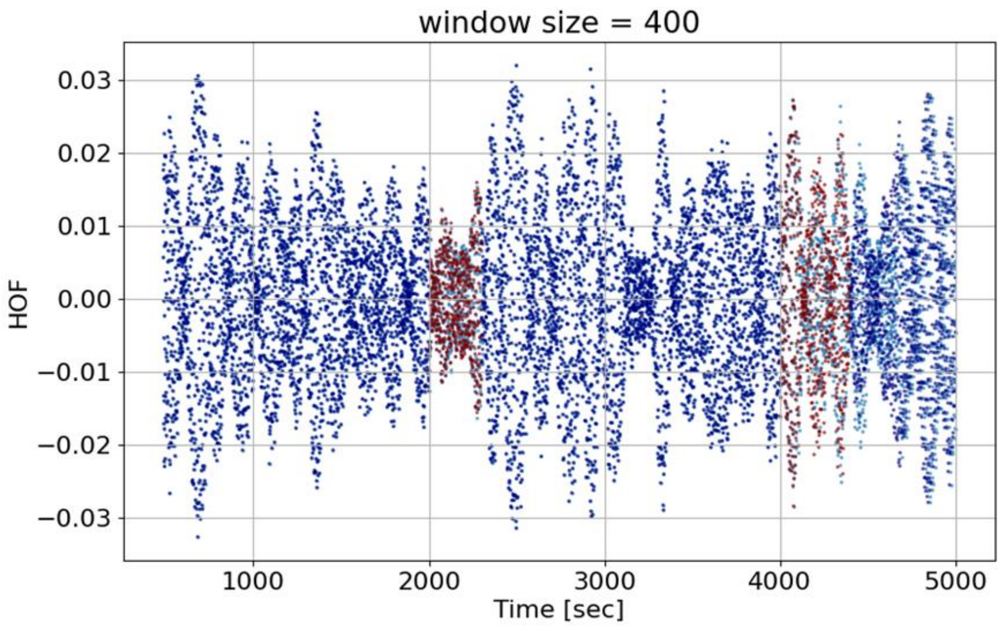

Figure 42. HOF with low sensitivity for subtle narrow anomalies.

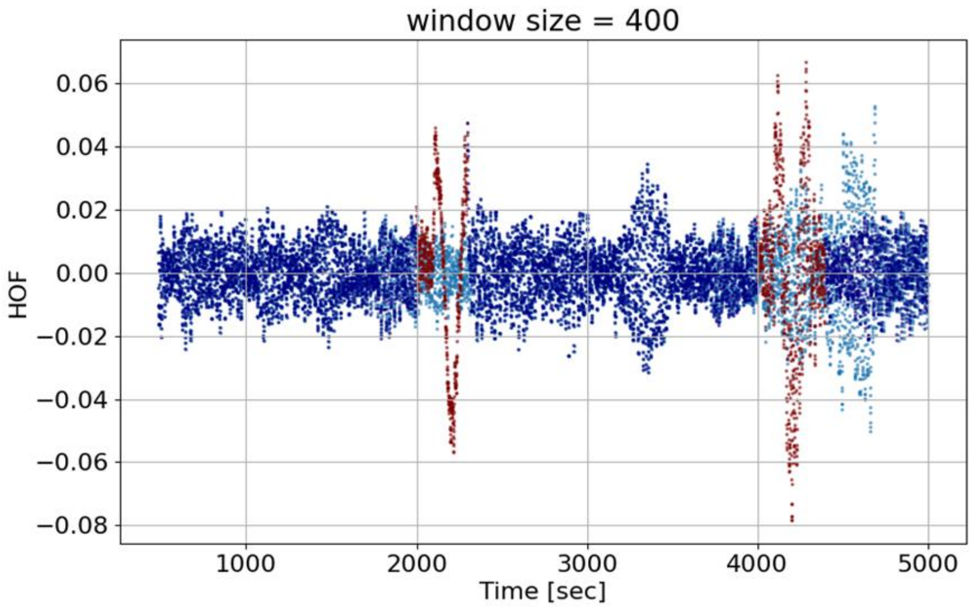

Figure 43. HOF with moderate sensitivity for subtle narrow anomalies.

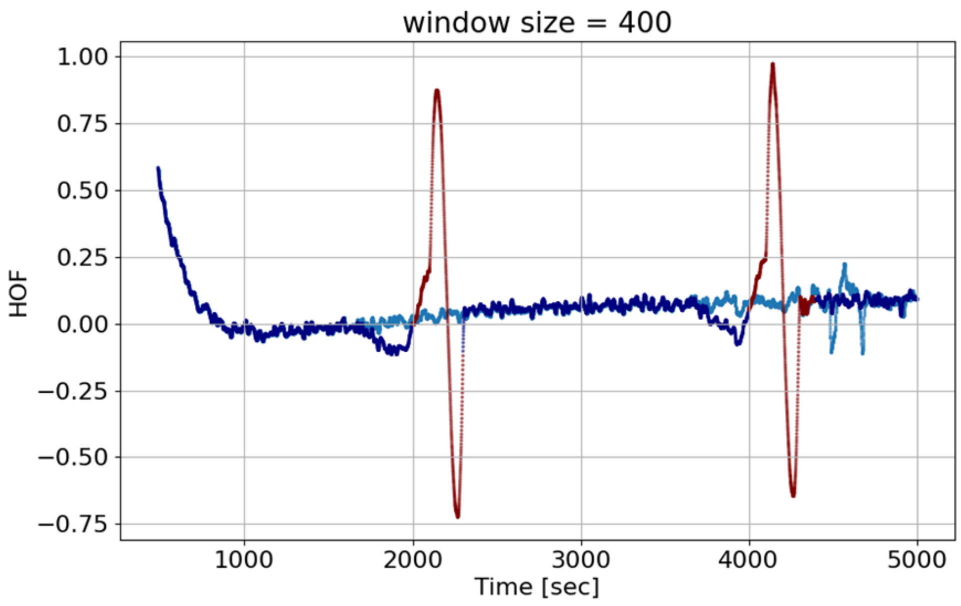

Figure 44. First HOF with high sensitivity for subtle narrow anomalies.

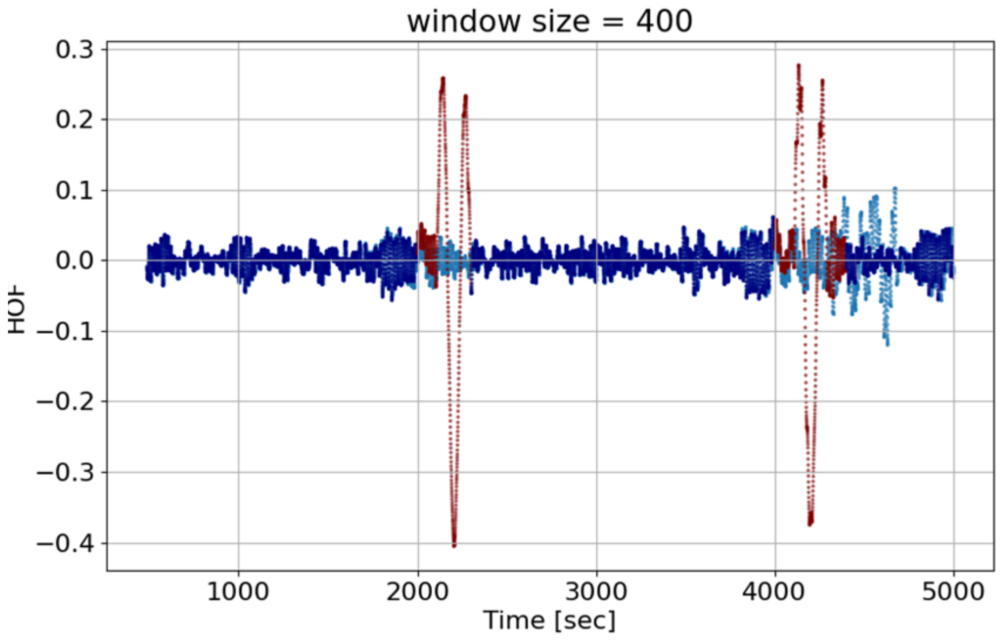

Figure 45. Second HOF with high sensitivity for subtle narrow anomalies.

The next set of results are for the narrow and persistent anomalies. Figure 46 and Figure 47 show candidate HOFs for classification. Note that, while the HOF in Figure 47 is very sensitive to the narrow anomaly at 6000 seconds, it is completely blind to the persistent anomaly at 8000 seconds. Using feature fusion, two additional features were constructed and employed for classification, as shown in Figure 48 and Figure 49. The fused feature in Figure 49 shows higher sensitivity to the first labeled anomaly, and so it was selected.

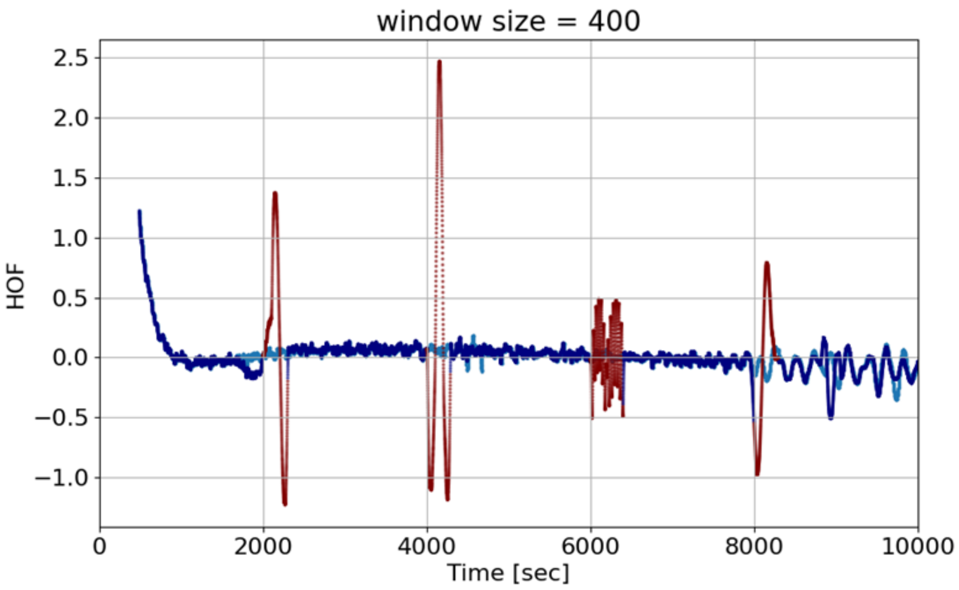

Figure 46. First HOF for the narrow and persistent anomalies.

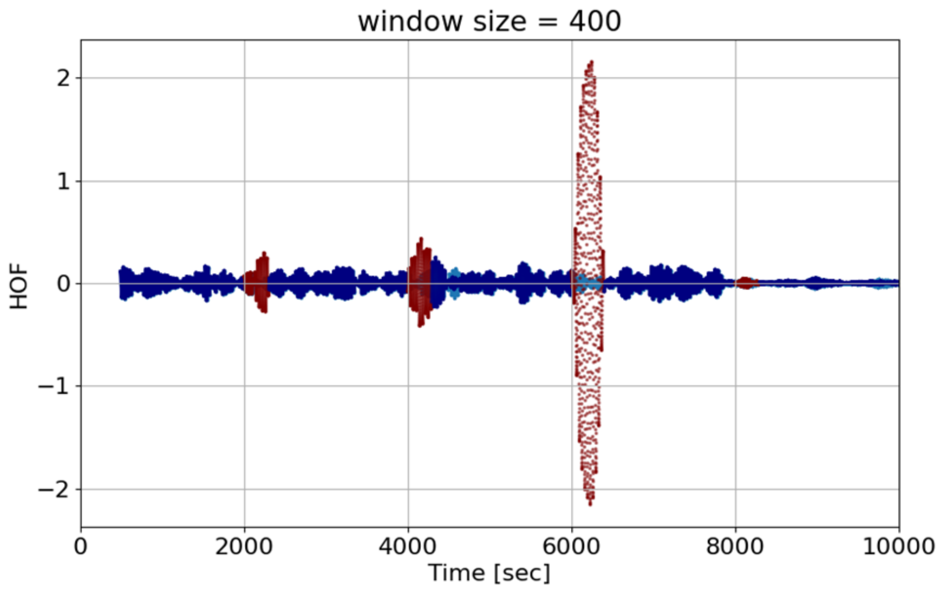

Figure 47. Second HOF for the narrow and persistent anomalies.

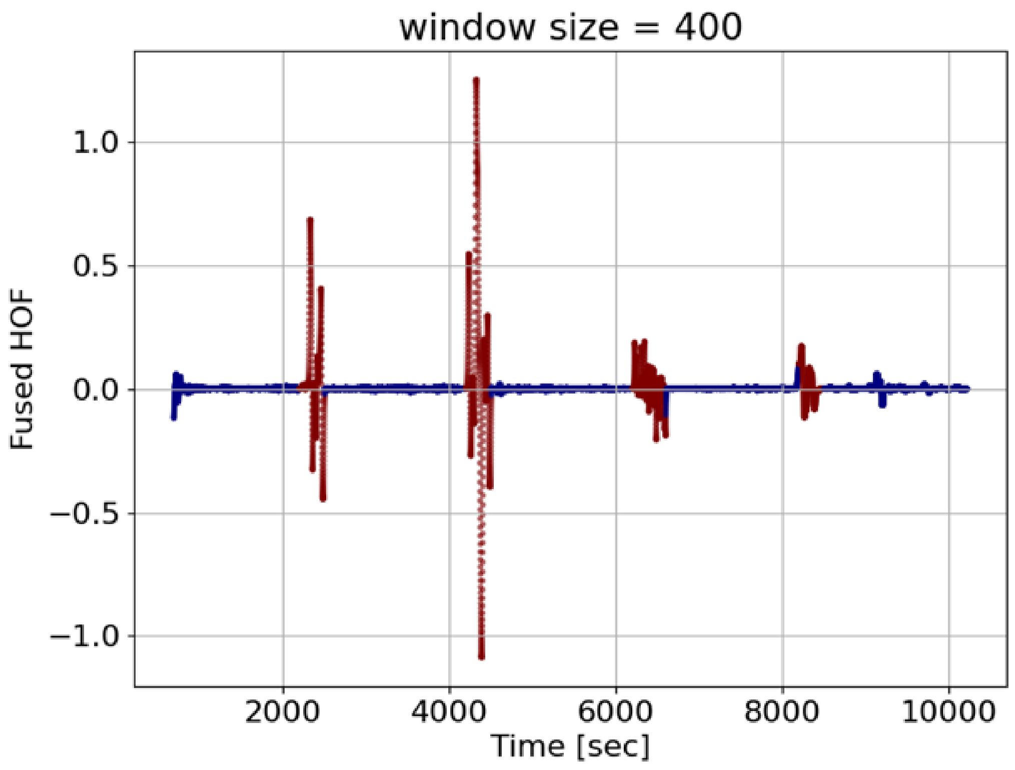

Figure 48. First Fused HOF for the narrow and persistent anomalies.

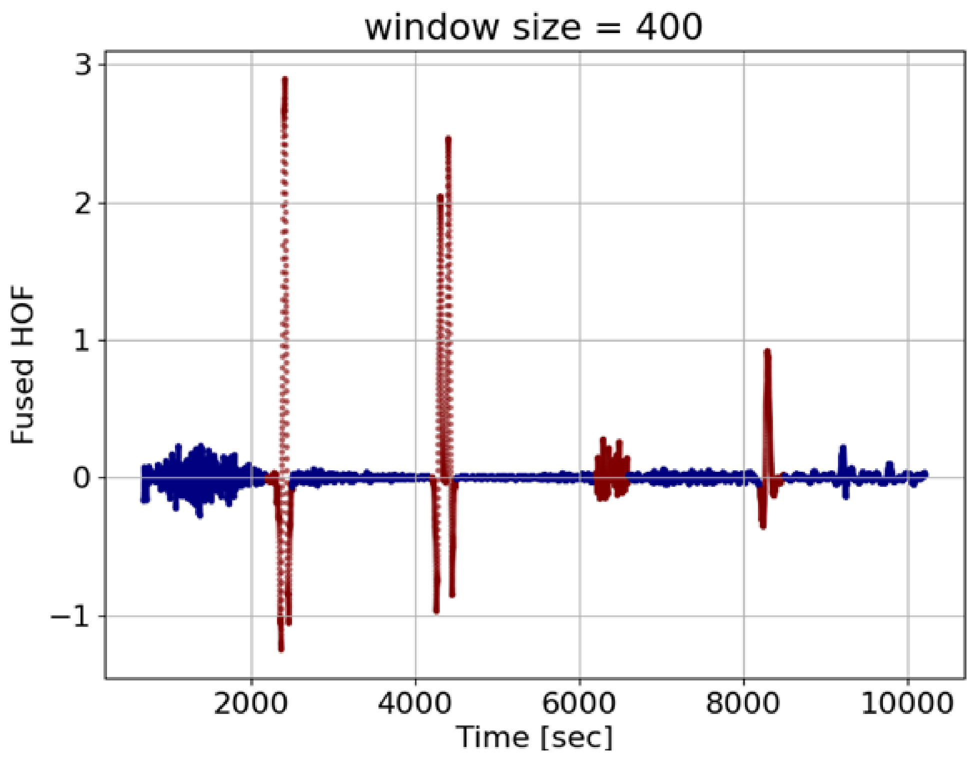

Figure 49. Second fused HOF for the narrow and persistent anomalies.

Finally, the results are shown for when the anomalies were implemented into the steam generator feedwater pump. The behavior is similar to what was already observed, with the proposed semi-supervised ML algorithm able to capture the anomaly shortly after its inception (see Figure 50). Note that for these results, the values have been transformed into a magnitude, so are strictly positive.

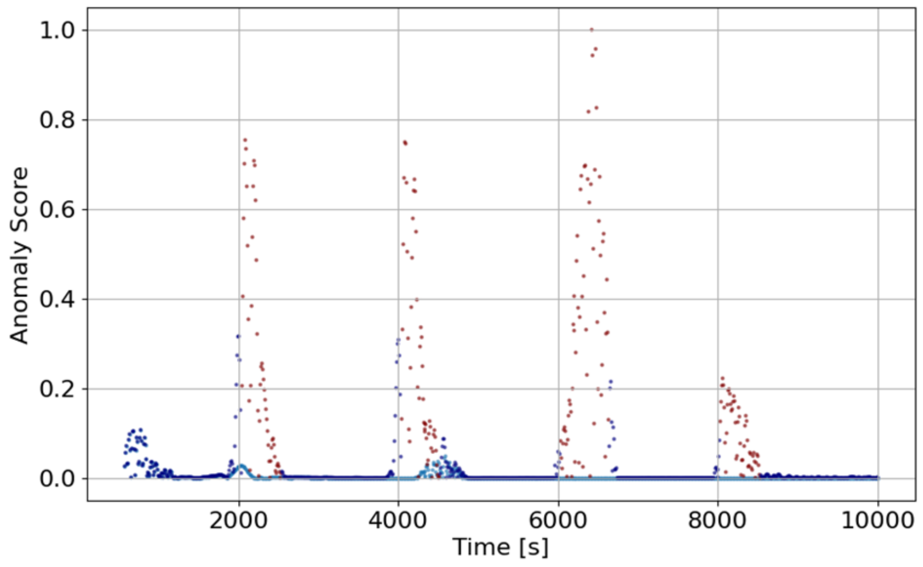

Figure 50. HOF with high sensitivity for the steam generator feedwater pump anomaly.

3.2.3 SMD Results

The HOF algorithm was applied to the three-mass dataset described in Section 2.1. For this problem, the data volume is extremely large. Since the HOF algorithm does not require large data volumes, the data are down-sampled initially by employing a smoothing window, reducing the size of the data by two orders of magnitude.

The analysis was performed for the three-mass problem, with the selected HOFs plotted in Figure 51. For a three-mass problem without any labels, an HOF is selected randomly. Here, there is no simple approach to selecting a good candidate feature. However, any labels will enable the selection of better candidate HOFs. As more labels are used, different HOFs are selected, but this has little impact on the results. The results generally indicate that, after adding a small number of labels, little improvement is obtained by adding more. The results, however, show some single-point false positives (i.e., the HOF has a very high anomaly score at a single time step), represented as sharp blue lines. In typical anomaly detection, an additional criterion is often employed to guard against such peaks (e.g., declare an anomaly when the anomaly score exceeds the threshold a few times in a row).

There is an important distinction to be made here in regard to semi-supervised ML techniques. In some existing techniques, the time series data associated with the labeled anomalies are employed to train classifiers to learn the signature of the anomaly. In the HOF algorithm, the anomalous data are not used to learn the anomalies. Instead, their locations are used to down select the HOFs obtained from the unlabeled time series data. If the unlabeled time series data contain too many anomalies, the HOFs are expected to fail to capture the anomalies, since they become tuned to the shapes of the anomalies. This, however, is unlikely to happen in realistic applications.

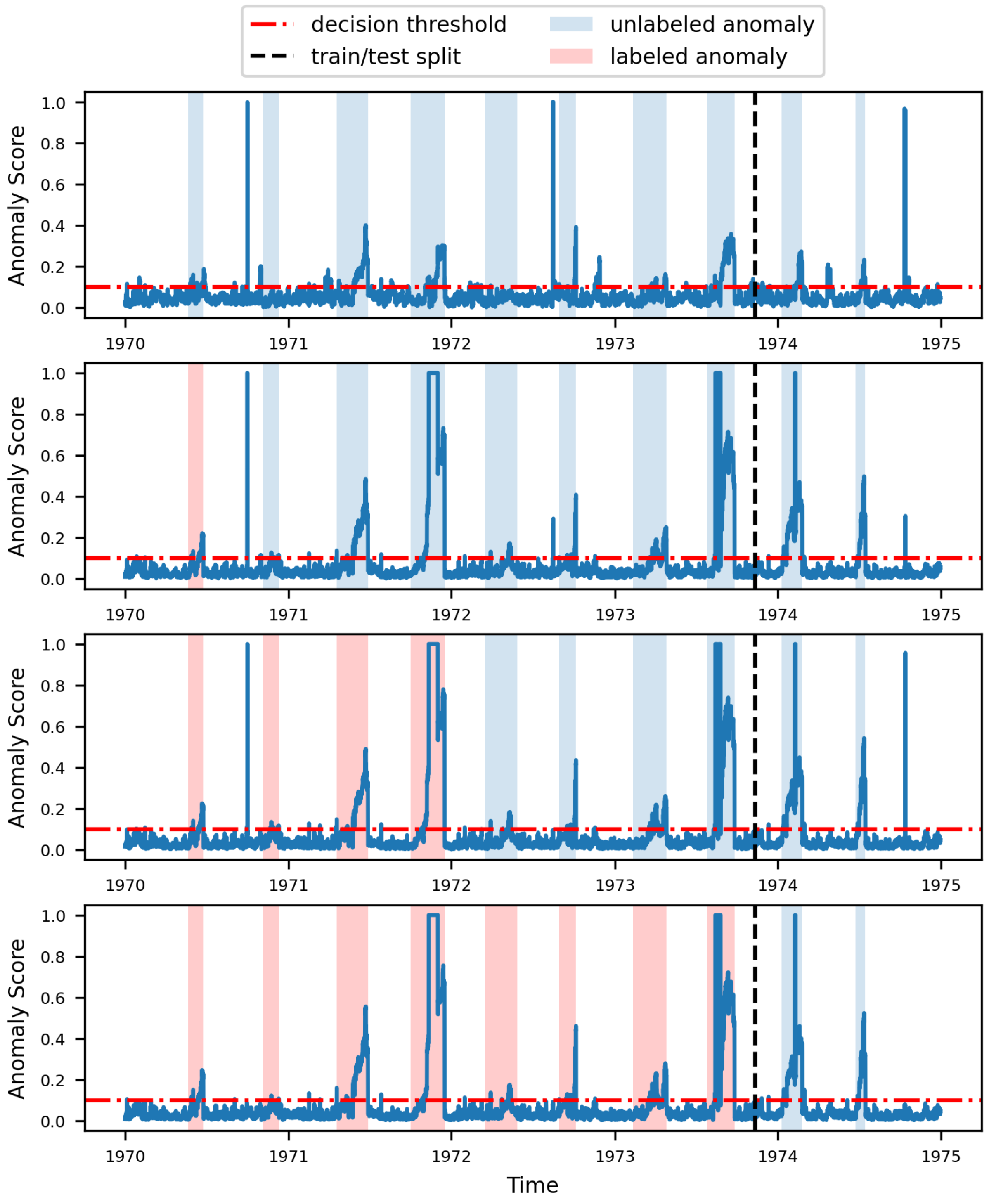

Figure 51. Three mass-spring anomaly detection using the HOF method.

4. CONCLUSIONS

Detection of subtle process anomalies can increase detection lead times for equipment failures, enabling plants to mitigate unexpected partial or full outages, and generating significant cost savings. Many previous and ongoing efforts have focused on unsupervised ML methods, despite supervised methods often outperforming them due to the added value of knowing the true labels. The challenge in implementing supervised ML methods is that it remains difficult and expensive to compile a full set of reliable labels. As an alternative, this effort proposes using CRs, which are developed by the plants as part of their corrective action programs, to achieve partial labeling of anomalies. This approach (i.e., some data labeled, but most remaining unlabeled) represents a semi-supervised ML problem, and this current effort focused on implementing semi-supervised ML methods to fuse partial labeling with sensor data in order to test the hypothesis that partially labeled anomalies can improve anomaly detection performance.

In this effort, two semi-supervised ML methods were used. The first was the deep SAD method, which can handle labels ranging from fully unsupervised to fully supervised cases. The second was a newly designed ML method developed for this effort and referred to as the HOF-based method. To evaluate the two methods in controlled environments, synthetic data generators were developed and used. An SMD system simulator commonly found in mechanical engineering references was used to create two use cases: a one-mass and a three-mass system. Anomalies were introduced by changing the spring and damper coefficients while the system was actuated by random forces. The second dataset used the commercial Dymola-Modelica software to build a simplified nuclear reactor model. Anomalies were added in the form of corrupted sensor readings inside of control loops. The deep SAD method was tested using the SMD system, while the HOF method was tested using both aforementioned datasets.

Starting with the deep SAD method, this effort made major modifications to an existing anomaly detection approach in order to focus on anomaly detection for time series data. Specifically, it used a state-of-the-art CNN architecture to learn critical features from the data—features potentially usable to differentiate between normal and anomalous conditions. The deep SAD approach was used to compare several different test cases: unsupervised, semi-supervised with varying numbers of labels, and fully supervised. The results of these cases demonstrated that labels improve the confidence that true anomalies are being detected, increasing the true positives and decreasing the false negatives. Such an approach would help address the backlog of possible anomalies identified at the M&D centers.

The HOF method developed features a priori rather than learn them. In this method, the RWD algorithm is first applied to capture the dominant temporal variations, which are used as the basis for defining another set of features, denoted as HOFs. Though the HOFs are noise-dominated (i.e., they carry no useful information under normal operation), anomalous behavior is expected to stimulate them. This method provides flexibility in selecting features, allowing a semi-supervised ML algorithm to select the features associated with a given type of labeled anomaly. For this effort, a simple criterion was employed to down select the HOFs, based on the available labeled anomalies. Results indicate that, through initial careful selection of features, one can strategically use the available (often small number) labeled anomalies to identify the best features with which to train a classifier for anomaly detection. This means that the semi-supervised framework considerably improved the algorithm's ability to detect subtle process anomalies.

5. REFERENCES

1. Garga, A. K., et al. 2001. “Hybrid reasoning for prognostic learning in CBM systems.” 2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542) 6 (March): 2957-2969. https://doi.org/10.1109/AERO.2001.931316.
2. Zhao, X., et al. 2009. “Multi-class semi-supervised learning in machine condition monitoring.” 2009 International Conference on Information Engineering and Computer Science (December): 1-4. https://doi.org/10.1109/ICIECS.2009.5365024.
3. Miao, Q., et al. 2017. “Condition multi-classification and evaluation of system degradation process using an improved support vector machine.” Microelectronics Reliability 75 (August): 223-232. https://doi.org/10.1016/j.microrel.2017.03.020.
4. Liu, J., et al. 2017. “A hybrid generalized hidden Markov model-based condition monitoring approach for rolling bearings.” Sensors 17 (5): 1143. http://dx.doi.org/10.3390/s17051143.
5. Gebraeel, N.Z., et al. 2005. “Residual-life distributions from component degradation signals: A Bayesian approach.” IIE Transactions 37, no. 6 (February): 543-557. https://doi.org/10.1080/07408170590929018.
6. Tian, Z. 2009. “An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring.” Journal of Intelligent Manufacturing 23 (November): 227-237. https://doi.org/10.1007/s10845-009-0356-9.
7. Al Rashdan, A., et al. 2019, “Subtle process anomalies detection using machine learning methods.” INL/EXT-19-55629, Idaho National Laboratory.
8. Al Rashdan, A., et al. 2020, “An Applied Strategy for Using Empirical and Hybrid Models in Online Monitoring.” INL/EXT-20-59688, Idaho National Laboratory.
9. U.S. NRC. n.d. “Corrective Action Program.” Accessed August 2021. https://www.nrc.gov/reading-rm/basic-ref/glossary/corrective-action-program.html
10. 2011. ML112241408, “Attributes of an Effective Corrective Action Program (Draft),” Nuclear Regulatory Committee.
11. Dimla Jr, D. E., P. M. Lister, and N. J. Leighton, 1997. “Neural network solutions to the tool condition monitoring problem in metal cutting—A critical review of methods.” International Journal of Machine Tools and Manufacture 37, no. 9 (September): 1219-1241. https://doi.org/10.1016/S0890-6955(97)00020-5.
12. Burke, L. I. and S. Rangwala. 1991. “Tool condition monitoring in metal cutting: a neural network approach.” Journal of Intelligent Manufacturing 2, no. 5 (October): 269-280. https://doi.org/10.1007/BF01471175.
13. Zhu, X. J. 2005. “Semi-supervised learning literature survey.” University of Wisconsin. http://digital.library.wisc.edu/1793/60444
14. Nigam, K., et al. 2000. “Text classification from labeled and unlabeled documents using EM.” Machine learning 39 (May): 103-134. https://doi.org/10.1023/A:1007692713085.
15. Maaløe, L., et al. 2020. “Condition Monitoring in Photovoltaic Systems by Semi-Supervised Machine Learning.” Energies 13, no. 3 (January): 584. https://doi.org/10.3390/en13030584
16. Yoon, A. S., et al. 2017. “Semi-supervised learning with deep generative models for asset failure prediction.” arXiv preprint. https://arxiv.org/abs/1709.00845.

17. Zhang, S., et al. 2019. “Semi-supervised learning of bearing anomaly detection via deep variational autoencoders.” arXiv preprint. https://arxiv.org/abs/1912.01096.

18. Collobert, R., et al. 2006. “Large scale transductive SVMs.” Journal of Machine Learning Research 7, (December): 1687-1712.

19. Chapelle, O., V. Sindhwani, and S. S. Keerthi. 2006. “Branch and Bound for Semi-Supervised Support Vector Machines.” In NIPS (January): 217-224.

20. Liu, C. and K. Gryllias. 2020. “A semi-supervised Support Vector Data Description-based fault detection method for rolling element bearings based on cyclic spectral analysis.” Mechanical Systems and Signal Processing 140 (June): 106682. https://doi.org/10.1016/j.ymssp.2020.106682

21. Mao, W., et al. 2020. “Online detection of bearing incipient fault with semi-supervised architecture and deep feature representation.” Journal of Manufacturing Systems 55 (April): 179-198. https://doi.org/10.1016/j.jmsy.2020.03.005.

22. Tan, Y., et al. 2020. “A comparative investigation of data-driven approaches based on one-class classifiers for condition monitoring of marine machinery system.” Ocean Engineering 201 (April): 107174. http://dx.doi.org/10.1016/j.oceaneng.2020.107174.

23. Zhu, X., Z. Ghahramani, and J. D. Lafferty. 2003. “Semi-supervised learning using gaussian fields and harmonic functions.” In Proceedings of the 20th International conference on Machine Learning (ICML-03) (August): 912-919.

24. Jaakkola, M. S. T. and M. Szummer, 2002. “Partially labeled classification with Markov random walks.” Advances in neural information processing systems (NIPS) 14 (January): 945-952. https://dl.acm.org/doi/10.5555/2980539.2980661.

25. Wang, X., H. Feng, and Y. Fan. 2015. “Fault detection and classification for complex processes using semi-supervised learning algorithm.” Chemometrics and Intelligent Laboratory Systems 149, no. 5 (December): 24-32. https://doi.org/10.1016/j.chemolab.2015.10.019.

26. Yuan, J. and X. Liu. 2013. “Semi-supervised learning and condition fusion for fault diagnosis.” Mechanical Systems and Signal Processing 38, no. 2 (July): 615-627. https://doi.org/10.1016/j.ymssp.2013.03.008.

27. Triguero, I., S. García, and F. Herrera. 2015. “Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study.” Knowledge and Information systems 42 (November): 245-284. https://doi.org/10.1007/s10115-013-0706-y.

28. Ghosh, S., M. Roy, and A. Ghosh. 2014. “Semi-supervised change detection using modified self-organizing feature map neural network.” Applied Soft Computing 15 (February): 1-20. https://doi.org/10.1016/j.asoc.2013.09.010.

29. Blum, A. and T. Mitchell. 1998. “Combining labeled and unlabeled data with co-training.” In Proceedings of the eleventh annual conference on Computational learning theory (July): 92-100. http://dx.doi.org/10.1145/279943.279962.

30. Nigam, K. and R. Ghani. 2000. “Analyzing the effectiveness and applicability of co-training.” In Proceedings of the Ninth International Conference on Information and Knowledge Management (November): 86-93. https://doi.org/10.1145/354756.354805.

31. Hu, C., et al. 2015. “A co-training-based approach for prediction of remaining useful life utilizing both failure and suspension data.” Mechanical Systems and Signal Processing 62 (October): 75-90. https://doi.org/10.1016/j.ymssp.2015.03.004.

32. Greenwood, M. S., et al. 2017. “TRANSFORM - TRANsient Simulation Framework of Reconfigurable Models. Computer Software.” 005452IBMPC00, Oak Ridge National Laboratory. https://github.com/ORNL-Modelica/TRANSFORM-Library.

33. Ruff, L., et al., 2020. “Deep Semi-Supervised Anomaly Detection.” 2020 International Conference on Learning Representations (ICLR). https://arxiv.org/abs/1906.02694.

34. Wang, Z., W. Yan, and T. Oates. 2017. "Time series classification from scratch with deep neural networks: A strong baseline." 2017 International Joint Conference on Neural Networks (IJCNN), (May): 1578-1585, https://doi.org/10.1109/IJCNN.2017.7966039.

35. Li, Y. and H. S. Abdel-Khalik, 2021. “Data trustworthiness signatures for nuclear reactor dynamics simulation.” Progress in Nuclear Energy 133 (March): 103612. https://doi.org/10.1016/j.pnucene.2020.103612.

36. Sundaram, A., Y. Li, and H. S. Abdel-Khalik, 2021. “A Multi-level Feature Extraction and Denoising Approach to Detect Subtle variations in Industrial Control Systems.” Proceedings of M&C October 2021.

37. Zugasti, E., M. Iturbe, I. Garitano, and U. Zurutuza. “Null is Not Always Empty: Monitoring the Null Space for Field-Level Anomaly Detection in Industrial IoT Environments.” Global Internet of Things Summit 2018.