 doi:10.20944/preprints202012.0058.v1

Article

Sensor and Component Fault Detection and Diagnosis for Hydraulic Machinery Integrating LSTM Autoencoder Detector and Diagnostic Classifiers

Ahlam Mallak [1,*], Madjid Fathi [1]

[1] Department of Electrical Engineering and Computer Science, Knowledge-based Systems and Knowledge Management, University of Siegen, Siegen, Germany; fathi@informatik.uni-siegen.de

* Correspondence: Ahlam.mallak@Ymail.com

Abstract: Anomaly occurrences in hydraulic machinery may lead to massive systems shut down, jeopardizing the safety of the machinery and its surrounding human operator(s) and environment, and the severe economic implications succeeding the faults and their associated damage. Hydraulics are mostly placed in ruthless environments, where they are consistently vulnerable to many faults. Hence, not only the machines and their components are prone to anomalies, but also the sensors attached to them, which monitor and report their health and behavioral changes. In this work, a comprehensive applicational analysis of anomalies in hydraulic systems extracted from a hydraulic test rig is thoroughly achieved. Firstly, we provided a combination of a new architecture of LSTM autoencoders and supervised machine and deep learning methodologies to perform two separate stages of fault detection and diagnosis. The two phases are condensed by: the detection phase using the LSTM autoencoder. Followed by the fault diagnosis phase represented by the classification schema. The previously mentioned framework is applied to both component and sensor faults in hydraulic systems, deployed in the form of two in-depth applicational experiments. Moreover, a thorough literature review of the past decade related work for the two stages separately is successfully conducted in this paper.

Keywords: Deep Learning; LSTM Autoencoder; Supervised Learning, Hydraulic Test Rig; Sensor Faults; Component Faults.

1. Introduction

Mechanical machines are considered a vital part of the industrial operation. Hence, they play a tremendous role in the production and manufacturing processes. Due to their major importance in the production line, they are usually placed in tough locations and tough environments, which make them susceptive to the occurrence of various faults and malfunctions. Nowadays, the industrial applications are getting more complicated and scalable than ever, which contributed tremendously to the complexity of fault detection in those systems, as well as making those tasks quite challenging [1]. The study in [2] indicated that 70-90% of the incidents associated to industrial operations are caused by human workers or operators. Consequently, the need for computer-aided diagnosis emerged, to ensure highly accurate fault detection, prediction, and diagnosis of systems with extreme complexities. Moreover, computer-aided diagnosis may also contribute to the speed and precision of the recovery actions deployment required following the fault appearance. The main goal of automated fault detection is to capture the anomalies accurately as soon as they manifest, to ensure deploying the necessary maintenance procedures, and to dodge economical, humanitarian, and environmental tragedies. Creating a solid fault detection and diagnosis systems, not only contribute to reducing the risk, and providing safety to human operators and the environment. But also, they play a major role in cutting down the costs related to unnecessary maintenance.

Thereafter, fault detection and diagnosis in mechanical devices placed in complex systems like industry is always a hot research topic.

Automated Fault Detection and Diagnosis (FDD) algorithms and systems are usually dependent on the training and analysis of datasets, in which they are extracted from numerous sensors attached to the industrial equipment and its components. Those sensors continuously send signals essential to monitor each component of the mechanical machine. In other words, sensor readings are most often the modalities, or the source of row data associated to automated FDD systems. The health of these sensors is the key to monitor those components properly, which leads to accurate component diagnosis results. Although these sensors are substantial for computer-aided diagnosis, they are mostly ridiculously cheap and perform under extreme environmental conditions due to their attachment to the machinery device. Therefore, sensors in mechanical machines are prone to malfunctions and variety of faults themselves. Smart FDD systems should monitor both the mechanical devices and their components, as well as the sensors' health status of those who are responsible of reporting the health indications of mechanical components. Hence, it is undeniably important to establish sensor FDD systems along with the component FDD ones when monitoring industrial operations.

One of the most essential mechanical equipment for industrial processes are hydraulic systems. The hydraulic system's data applied to this study is gathered from a hydraulic test rig. The applied dataset [3] represents real measurements of multivariate, time-series sensors, placed in a hydraulic test rig. The purpose intended for the data collection is to monitor and assess the hydraulic system health condition. A test rig can be defined as a piece of mechanical devices that is mainly utilized to assess, evaluate, and test the capacities and performance of other mechanical machines, or just certain components of them. Test rigs can be called by various terminologies including test bench or test pay, and testing station. Test rigs are common in a wide range of industrial fields from hydraulic systems to aerospace. They literally have a vast scope of testing methods, and analytical parameters such as, manual, cyclical, brake and burst testing.

Hydraulic systems obtained from a hydraulic test rig are the focus of this work, due to their importance and limited FDD resources in the past decade, in which a comprehensive FDD system of both component and sensor faults is included.

Recently, an effective FDD framework has been trending, which evolves around preforming the detection and diagnosis phases separately to ensure detecting rare fault occurrences in various systems. The first phase is the detection one, where it is often represented by a healthy signal reconstructed schema using different Machine Learning (ML) and Deep Learning (DL) methodologies. i.e. autoencoders. The autoencoder is trained using the fully efficient inputs of the dataset, which represents the healthy form of the training data. Eventually, the autoencoding model should be able to reconstruct the healthy version of the input data at any given point of time. Comparing the reconstructed healthy signal with the given one, provides an indication of any fault presence. The more the given signal is identical to the healthy reconstructed one, the most likely it is a healthy signal and lacks the presence of anomalies, and vice versa. Meanwhile, in the fault diagnosis phase, the faults or deviations captured in the first phase (detection phase), are then used to train a certain ML or DL classification model.

Our Contribution

In this work, a comprehensive FDD approach for hydraulic systems is proposed. where an additional step is added in advance to the diagnosis using the classification phase, to overcome the weaknesses of supervised diagnostic approaches in capturing rare and beyond the existing labels faults. To overcome this challenge, in this section the detection and diagnosis phases are performed separately. Where the detection phase is done by applying a Long Short-Term Memory (LSTM) autoencoder to detect rare and unprecedented faults. Followed by the diagnosis phase using the ML and DL classifiers to analyze the nature of the captured faults in phase one.

This approach has been already created in the literature; however, our work is beyond the state-of-the-art by the following: (1) It is the first time this schema is applied to hydraulic test rigs

 doi:10.20944/preprints202012.0058.v1

data. (2) This work was applied to both sensor and component faults in a hydraulic test rig, in two separate thorough experiments. (3) In the detection phase represented by the LSTM autoencoder, we presented a new criteria to calculate the deviation between the predicted signal and the input one, which is proven more effective than the traditional method in computing more accurate diagnostic thresholds. (4) In the fault diagnosis phase proposed by the classification, we provided a full comparison results between numerous ML and DL classifiers of different functionality and techniques. (5) In the same phase, we also provided a behavior analysis of each ML and DL classifiers used in the diagnosis phase with a bunch of time-domain feature selection methods, to help further research in the future mapping each classifier with their best or least suitable time-domain features to achieve either component or sensor FDD in hydraulic systems. (6) In this work, a comprehensive literature review for FDD in hydraulic systems based on autoencoders is thoroughly presented.

The rest of paper is as follows: In section 2, an in-depth analysis of the related work is presented, where a list of FDD research in hydraulic systems, based on autoencoding techniques for the past century is discussed. Section 3 explains the overview of the FDD system utilized in this work. In section 4, experimental results are showcased and analyzed in two main experiments, where one is applied on injected sensor faults, while the other is conducted to achieve FDD for component faults in hydraulic test rigs. Finally, the discussion and future work is talked in section 5.

2. Related Work

Autoencoder Approaches for FDD in Hydraulic Machinery

The work in [4] shows a combined approach to achieve component fault detection and diagnosis of rare events occurring in chemical factories. The proposed method joints LSTM autoencoder as the detection phase, followed by the diagnosis phase using LSTM classifier. This approach is used to detect and diagnose faults of the Tennessee Eastman benchmark [5], which represent a dataset extracted from a simulator of actual chemical processes that includes various components: reactors, condensers, vapor-liquids… and so on. In the detection phase, the sequence comparison between the reconstructed sequence and the given one is achieved by applying the traditional signal difference. In the diagnosis phase, no feature selection or extraction approach is used prior to the classification using LSTM classifiers. Moreover, a solo comparison to Convolutional Neural Network (CNN) is made, but no comparisons with other DL or ML classifiers are conducted. Lu et al. [6] introduced a novel autoencoder called Stacked Denoised Autoencoder (SDA) that is used to detect component faults in rotary machineries. The method is applied to a dataset extracted from a physical simulation of a bearing test-rig. SDA implicitly feature engineer the data, which is compared to Principal Component Analysis (PCA) and regular stacked autoencoders (SAE). Moreover, the classification results provided by SDA are then compared to SAE, Support Vector Machines (SVM), Random Forest (RF) and regular autoencoders. The work proposed in [7] shows a novel approach of creating a new type of autoencoders, in which it combines stacked autoencoders and LSTM network. The work is separated into two-phases: (1) Feature transformation using LSTM stacked autoencoders. (2) Apply LSTM for fault identification. The proposed method focuses on detecting injected component faults to a Bently Nevada Rotor Kit RK3, which is designed to physically simulate rotating equipment and its conditions. The raw vibrational signals are directly collected from the RK3 kit, then Wavelet Packet Decomposition (WPD) method is used to select features in both time-domain and frequency-domain, to ensure a wide investigation in both domains, followed by transforming the selected features using the stacked autoencoders in account to their mean square error calculated, which helps in generating a threshold for each feature. Finally, the fault detection accuracy for each feature is validated using five-fold cross validation after classification using K-Nearest Neighbour (KNN) method. No comparisons of other feature selection methods to WPD, or additional classifiers besides KNN are used in the mentioned work. According to [8] a component fault diagnosis system of rolling bearings using stacked autoencoders is introduced, as well as compared to two other deep learning

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

schemas: (1) Deep Boltzmann Machines and (2) Deep Belief Networks. Four experiments are conducted using various data pre-processing schemas using time-domain, frequency-domain, and time-frequency domain.

As stated in [9] a deep autoencoder is developed to diagnose vibration signals in both gearboxes and electrical locomotive roller bearings. The novel approach proposed consist of two steps: (1) the design of the deep loss function in the autoencoder using maximum correntropy. (2) Applying artificial fish swam algorithm to optimize the autoencoder's parameters and its ability to extract valuable features. Similarly, the approach proposed in [10] demonstrates a new method of combining wavelet transform and stacked autoencoders, to diagnose faults occurring in roller bearing systems. Furthermore, a deep autoencoder in [11] is used to develop the quality of feature fusion, which contributes in aiding the diagnosis of faults in rotating machinery. The applied autoencoder is a collaboration between denoising autoencoders and contractive autoencoders. Where the deeply extracted features from both methods separately are then fused together using locality preserving projection (LPP). The fused features are then applied to SoftMax function to train the diagnosis process. In addition, another architecture of sparse autoencoders is performed in [12] to monitor and diagnose the component faults in motors and air compressors. The application of regular ML classifiers such as SVM requires intensive understanding and expertise in feature engineering. Thus, the application of autoencoders can massively facilitate the feature engineering process and perhaps outperform the regular feature engineering approaches. For that matter, sparse autoencoders are compared to other ML fault diagnosis methods such as, SVM and SoftMax regressor to classify faults in motors and air compressors. Accordingly, in [13] a multivariant fault diagnosis and health monitoring approach in rotating machines is introduced. This method is called "SAE-DBN" as a combination of a two-layered sparse autoencoder (SAE) to perform data fusion between the features of multi-sensors followed by the application of Deep Belief Networks (DBN) for the diagnosis. In [14] another approach using sparse autoencoders is proposed. The method is applied to induction motors monitoring and fault diagnosis purposes. The autoencoder application in [15] shows an ensemble, and deep approach of autoencoders designated to fault diagnosis in rolling bearings. Various activation functions are deployed at the same time, to create multiple autoencoders that are going to be combined later using a novel strategy. Finally, the work in [16] investigates fault detection and feature extraction schema for motors using an autoencoding schema of Recurrent Neural Networks (RNN). The explained schema for fault classification is applied directly on time-domain vibrational data then compared to the results conducted by a two-layered Artificial Neural Network (ANN) model. On a different note, the feature selection capacities of the RNN autoencoder was compared to PCA and Linear Discriminant Analysis (LDA) for dimensionality reduction. The vibrational signals used in this work were obtained form an actual motor positioned with different accelerometers in various locations. The table demonstrated below is created to conclude all the autoencoding FDD approaches in mechanical machinery performed in the past decade.

Table 1. Autoencoding-based Methods for FDD in Hydraulic Machinery.

Reference	Autoencoding Method	Mechanical Equipment	Fault Type/ Purpose	Dataset
[4]	LSTM Autoencoder+ LSTM Classifier	Chemical Reactor	Component Faults of Tennessee Eastman benchmark.	Tennessee Eastman benchmark [5].
[7]	Stacked Autoencoder LSTM + KNN	Rotating equipment	Injected Component faults to a physical simulation	Data collected from Bently Nevada Rotor Kit RK3 to simulate rotating device.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

[6]	Stacked Denoised Autoencoder	Rotary machinery	Component faults in a bearing test-rig	Data extracted from physical bearing test-rig.
[8]	Stacked deep autoencoders	Rolling bearings	Component faults in rolling bearings.	Gathered from UPS.
[9]	Another architecture deep autoencoder	Gearboxes and electrical locomotive roller bearings	Component faults in rolling bearings and electrical locomotive.	From a physical test rig.
[10]	Wavelet transform + stacked autoencoders	Roller bearing systems	Component faults in rolling bearings.	From case western reserve university (CWRU).
[11]	Another architecture of deep autoencoders	Rotating machinery	Component faults in rotating machinery	Physical rotor fault test, CWRU [17] and NASA datasets [18].
[12]	Another architecture of sparse autoencoders	Motors and air compressors	Component faults in motors and air compressors	Actual air compressor and motor
[13]	SAE-DBN (sparse autoencoder + Deep Belief Networks)	Rotating machines	Component faults in rotating machinery	Extracted from an experimental system.
[14]	Another architecture of sparse autoencoders	Induction motors	Component faults in induction motors	Fault simulator.
[15]	Ensemble deep autoencoder	Rolling bearings	Component fault diagnosis in rolling bearings	CWRU [17].
[19]	Another architecture of stacked autoencoders	Hydraulic pumps	Detect component faults in hydraulic pumps	Hydraulic pump of type axial piston pump (25MCY14-1B).
[16]	Autoencoding schema of RNN networks	Motors	Component fault detection and feature extraction in motors	Physical motor.

3. Hydraulic System FDD Overview

In this experiment the data used is collected from a condition monitoring of a hydraulic test rig, which is designated to test a hydraulic system. Thereafter, the data is used to conduct two main experiments, one to analyse the provided component faults at the total failure stage. The other experiment is concerned with sensor faults, where the fault injection takes place to successfully inject three main types of sensor faults; constant fault that covers constant low, constant high and constant zero faults, as well as gain faults and bias or offset faults. These injected faults along with the healthy

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

readings would eventually act as pre-defined classes necessary for the fault classification and the healthy signal reconstruction learning, respectively.

Figure 1, shows the two main experiments to detect and diagnose a variety of sensor faults and severe component failures in the hydraulic test rig readings, as an example of hydraulic system's data. In this work, two comprehensive experiments were conducted to guarantee performing fault detection and diagnosis for each component and sensor faults in the hydraulic system tested using a hydraulic test rig. The two experiments start mutually with applying the necessary data pre-processing steps, to ensure removing the unnecessary noise in the input signals, and to arrange the inputs in a way suitable for the LSTM autoencoder. Please note that the data applied, and its pre-processing differs between the sensor FDD experiment and the component one. Moreover, the data used for both experiments are filtered and organized differently. The data description and organisation for both experiments will be described in detail in the experimental results section of this work. For Sensor FDD, the available dataset of a hydraulic test rig lacks the presence of any sensor faults, which necessitates the injection of various sensor faults into the filtered and pre-processed data. The choice of which types of sensors faults to inject is decided upon convenience and necessity. For example, stuck-at or constant sensor fault was chosen to be injected in the data due to its simplicity to apply such an effect on different periods of time comparing to other data-centric sensor faults .i.e. outliers or spikes that are not easily predicted or frequently occurring, or even possessing a regular pattern in which they could be injected in the data. Gain and bias faults are an example of system-centric faults, which by definition are complicated to diagnose relying on the data alone, that is why these sensor faults are significant to study and apply algorithms with high accuracies to diagnose, as well as both mentioned faults have a clear definition and pattern that makes it easier to inject them to the dataset.

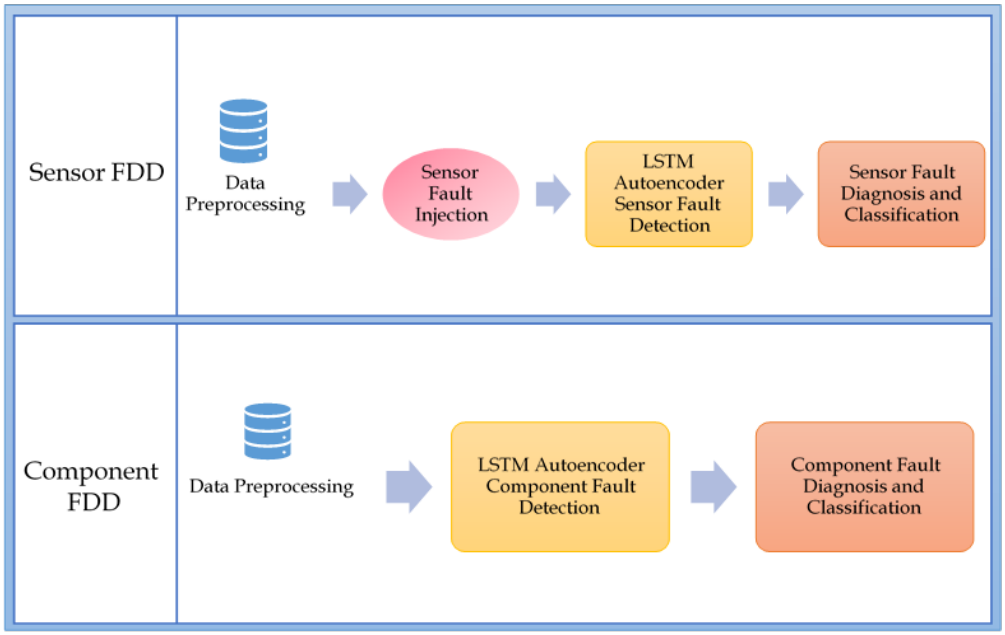

Figure 1 An Overview of the Two Experiments to Achieve FDD in Hydraulic Test Rigs for both Sensor and Component Faults.

For the component FDD experiments, the instances selected are the ones with full efficiency to be fed in the detection phase demonstrated by the LSTM autoencoder. However, the faults that are proceeded to the diagnosis stage are the ones representing total failure in the hydraulic test rig which are, cooler total failure, valve total failure, pump severe leakage and hydraulic accumulator total failure. In both experiments the detection phase is demonstrated by the LSTM autoencoder to reconstruct the healthy form of the input signal when the FDD is conducted and tested. The LSTM autoencoder is trained using healthy sensor data that shows full efficiency in both experiments.

 doi:10.20944/preprints202012.0058.v1

However, the data formulation and organisation is different between the two experiments, since the signal subject of reconstruction for sensor FDD is a window of 60 seconds values corresponding to each sensor separately in the hydraulic test rig, but in the component FDD the healthy signal used for training is organised without sliding windows, while each reading represents the values of all eleven sensors at this particular point of time, and how they altogether contributed in diagnosing the failure.

In the diagnosis phase for both experiments, the faulty data of both systems are fed separately into a classification process. The choice of traditional ML classifiers for this experiment is dependent on the selection of various supervised learning methods of entirely different functionality and mechanism as possible, to provide a broad and comprehensive analysis. The classifiers used in this experiment are LDA, Logistic Regression (LR), KNN, Decision Trees (CART), Naïve Bayesian (NB), SVM and finally RF. The DL methods chosen for this experiment are CNN and LSTM both applied in an interesting manner. The comparison includes the application of the chosen ML and DL approaches using different features extracted or selected via numerous feature extraction and selection methods such as, manually extracting time domain features for each sliding window such as the mean, variance, standard deviation and signal to noise ratio. And applying PCA directly using the raw multivariate time domain sensor data without dimensionality reduction, as well as using Recursive *k*-Means Silhouette Elimination (R*k*SE) feature selection and dimensionality reduction algorithm [20] . Finally, the trained models and saved thresholds from each experiment can be easily used to achieve run-time predictions of new samples at real-time. The FDD prediction at run-time can be done by the following: (1) The detection: a) predict the healthy reconstructed sample to the new sample using the trained model of LSTM autoencoder. b) Compare the reconstructed sample and its original form by applying the suitable sequence difference. c) Compare the calculated sequence difference to the threshold computed during the offline training stage, if the difference is greater than the threshold then a fault has been detected. When fault is detected, it needs to be passed to the next stage of fault diagnosis. (2) Fault diagnosis: this step is done by passing the new sample that has been detected as faulty, into the chosen trained classifier. After taking into consideration choosing the best features and the most optimal classifiers based on the comprehensive training and comparisons done previously in the model training offline phase.

4. Analysis and Experimental Results

4.1. Experiment One: Sensor FDD Using the Joint LSTM Autoencoder and Classifier Approach

In this section FDD of sensor faults in hydraulic test rigs using a joint approach between healthy signal reconstruction to detect sensor faults, followed by fault classification to diagnose the selected sensor faults is introduced, analyzed, and discussed.

The following subsections elucidate each step of the described approach applied on sensor faults and showcase their results. Figure 2 shows the steps included in experiment one, where below each step is elaborated in comprehensive detail.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

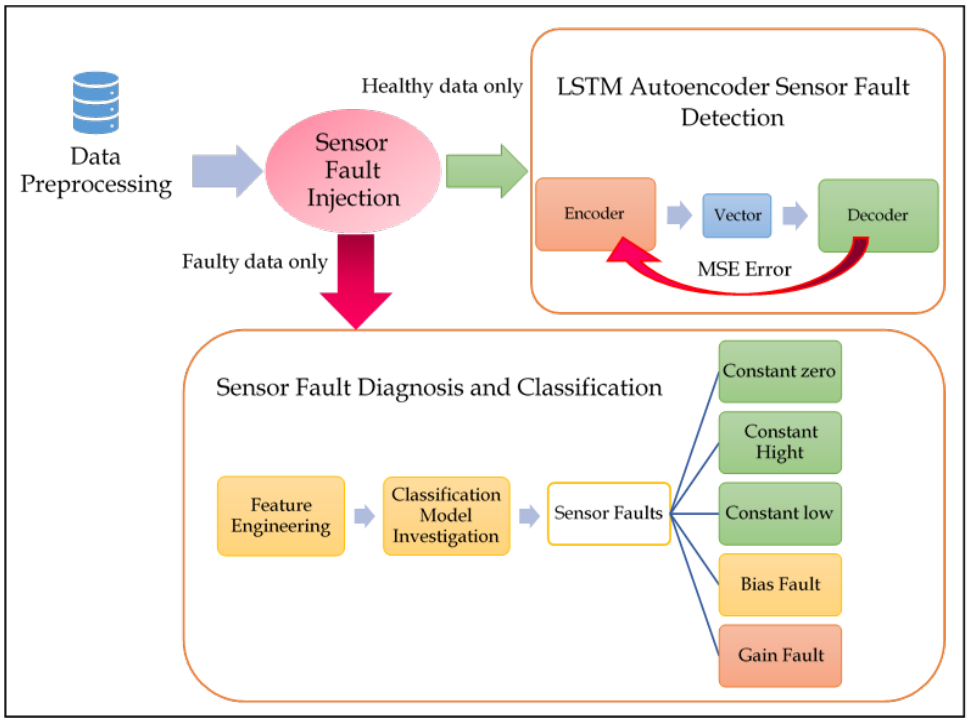

Figure 2 Sensor FDD Comprehensive Framework

The dataset used for the sensor fault detection and diagnosis is the hydraulic test rig dataset earlier. The mentioned data provided a wide range of component faults varies from slightly damaged to total failure. However, the dataset did not provide any sensor faults. Thus, it is essential to inject sensor faults to build the sensor FDD model.

Although the sensor FDD architecture navigated in this chapter is meant for multivariate time-series data, for simplicity reasons one sensor only is considered to show results for the sensor FDD process. Sensor PS1 (the first pressure sensor) is used to showcase the sensor FDD results during the fault injection, sensor FDD while using LSTM autoencoder and the sensor detection classification results.

The faults chosen to be injected are: (1) stuck-at: three main types of stuck at faults has been injected due to the fact that stuck-at or constant faults are the most common form of data-centric faults, and it shows a mighty severity of the sensor condition. Moreover, constant faults are extremely easy to inject. Consider the input sensor signal is $x(t)$ then the constant fault can be injected easily by following $x'(t) = c$ where c is a constant number representing the stationary condition of the sensor. Three main types of constant faults are added. constant zero when the sensor is stuck at zero, constant high when the sensor is stuck at the highest value in the window, and constant low stuck at the lowest point of the sensor readings during the observed window. We randomly injected 40 windows of size 60 seconds with constant zero fault, 7210 windows of size 60 readings of PS1 is injected with high and low constant faults, which make the overall number of windows injected with stuck-at fault is 7250 windows. (2) Gain fault and (3) Bias or offset faults. these faults are a type of system-centric faults; hence it is hard to observe their pattern through sensor signal's observation alone. So that, these faults are significant to study and build ML approaches to dynamically detect and diagnose them. Furthermore, both faults have a clear pattern that makes it easy to inject these types of faults in the data. Gain fault also known as amplification, where the original signal $x(t)$ is amplified with a constant w; $x'(t) = x(t) * w$. To inject this fault, randomly selected amplification number between 0.3-1.3 are selected each time, to regenerate the magnified fault signal. 7210 samples of 60 PS1 sensor readings are injected with randomly chosen gain values. Bias or offset fault is another example of calibration system-centric fault, where

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

the original signal is shifted with a constant value. Consider the original signal is $x(t)$ then the manipulated with offset signal is $x'(t) = x(t) + b$ where b is the constant number representing the bias or offset added to the signal. b value can be too small and hard to notice or observe, or too large and hard to ignore. As a result, it is essential to inject both cases of b. To achieve this, 3480 windows of size 60 were injected with a random number between 0.1-1 to represent the too tiny bias category, while the remaining 3730 windows of size 60, were injected with the comparatively larger biases that are randomly chosen between 1.1-50. Finally, the overall PS1 sensor data prepared after the fault inject process, possesses many windows of size 60 readings that consists of the following: (1) 7210 windows representing fully efficient windows as an example of healthy windows. (2) 7250 windows of constant faults (zero, high, low). (3) 7120 windows of gain fault. (4) 7210 windows of bias faults (low bias, high bias).

4.1.1. LSTM Autoencoder for Sensor Signal Reconstruction

To achieve the problem under investigation, the desired neural network should be able to perform sequence to sequence predictions. Hence, the input sequence is sliding window of the sensor PS1 and the reconstructed signal is from the same nature of the input sequence, as well as they are both having the same size of 60. Then the encoder-decoder type required to fit the problem is an autoencoder. The choice of LSTM as a type of DL algorithm is because its tendency to learn the hidden dependencies between many time points at once, which make LSTM one of the most suitable forms of DL when it comes to time-series data, especially sequence to sequence (seq2seq) operations.

The LSTM autoencoder created for this experiment, has only one batch of LSTM sequences. This batch is designed to be sequential in direction and nature, which means the input layer is directly connected to the hidden layers, then the hidden layer is connected to the output layer. The LSTM hidden layer consists of a hundred hidden LSTM neurons. The activation function applied for the designed DL model is ReLU, based on a try and error validation. The hidden layer is chosen to be fully connected by adding the dense layer of output equal to the overall output expected from the LSTM model. The optimizer chosen for the LSTM layer is Adam optimization algorithm.

In order to utilize the healthy windows of PS1 for LSTM use, it must go under a heavy pre-processing and structuring to fit the LSTM criteria. The pre-processing and restructuring including the following: (1) Flatten the data into a vector. (2) Normalize the flattened data between zero and one to be able to use in LSTM. (3) Create the target sequence $y(t)$ to reconstruct this is the most important step of all, which determines what to learn and what to predict. In our case, the input sequence is a sliding window of size 60, while the target sequence is the next sliding window. The shift or sliding step is assumed to be only one step to guarantee higher model accuracy, which means if the input point is $x(0)$ then the target point used to train the prediction model is $y(0) = x(1)$. So that in general, $y(t) = x(t + 1)$ (4) divide the flattened normalized vectors of $x(t)$ and $y(t)$ between training and testing samples, where the training windows are the 80% selected from the overall data, while the remaining 20% is divided equally between testing and prediction. (5) convert the flat, normalized vectors of $x(t)$ and $y(t)$ into a two-dimensional array of (number of samples, window size) shape. (6) convert the training and testing 2D tensor samples into a 3D tensor suitable to use in LSTM. LSTM units in Keras only accept the training and testing data in a 3D tensor shape following the size of (number of samples, time points, number of features). Where z-axis or pages or axis 0 is the number of samples, axis 1 or rows is the number of time points to store in the memory of LSTM and learn their dependencies, and finally axis 2 or columns represents the number of features inserted in the data. The $x(train)$ used in this experiment has the size of (11191, 60, 1), where 11191 of samples that has the size of (60,1) which is corresponding to one window of 60 only healthy readings of PS1.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

The previously designed LSTM model is trained and validated using the intensely pre-processed healthy data of PS1. In this experiment, the LSTM parameters are set to one hundred epochs and verbose equal one.

The validation results of the LSTM healthy signal reconstruction using the formulated testing data at the last epoch (number one hundred) has the errors Mean Square Error (MSE) and Mean Absolute Error (MAE) 0.000039871 and 0.0029, respectively, which both are considered exceedingly small loss values.

After training, evaluating, and testing the LSTM autoencoder model, it is time to start making fault detection decisions aided by the model, but the question arise, how to detect faults based on the quality of the reconstructed signal? Which brings up another important question: How to determine the fault detection threshold?

Taking a glance at the state-of-the-art methods helps answering the previous questions, as such the study researched in [4]. The approach they have applied is highly similar to our approach by having separate phases for both detection using signal reconstruction, and diagnosis applying fault classification. To find the difference between the predicted sequence and the input sequence, they used signal difference that can be easily calculated by taking the amplitude of the subtraction operation between the two sequences ($z(t) = |x(t) - x'(t)|$). Although using signal difference has shown accurate results, we propose a different signal similarity measure that showed more accuracy and performance when comparing to signal difference for fault detection.

The threshold determines what is faulty or healthy based on the value of the signal difference. If the value is higher than the designated threshold then the reconstructed signal is considered faulty, or else it is healthy. The threshold is best measured by creating a pool of various threshold values between the minimum and maximum values of the calculated signal difference. Followed by making the fault detection decisions on the prediction samples, based on each one of the thresholds in the pool. For each threshold in the pool, check if the prediction's sample signal difference is higher than the threshold to be considered faulty, while lower is detected to be healthy. Finally, calculate the precision, recall, f1-score, and accuracy for all the prediction results made by each threshold in the pool. The choice of the right threshold for the sensor fault detection, is made by choosing the threshold that guarantees the best precision to recall trade-off, also known as f1-score. In this experiment, a prediction dataset (500 windows of size 60 readings of PS1) that consists of various healthy and faulty samples, is used to determine the fault detection accuracy using the LSTM autoencoder sequence reconstruction. The 500 windows reconstructed using LSTM were then each compared to the original sequence to show how much they were deviated from the original window regarding their health status. The comparisons between the reconstructed windows and the original ones are made utilizing two main metrics: (1) signal difference: $z(t) - |x(t) - x'(t)|$. (2) our new metric using the complement of Pearson's autocorrelation.

Pearson's autocorrelation can be calculated using the formula shown below:

$$r_{xy} - \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \qquad \text{Eq. 1}$$

Where r_{xy} is the correlation between to vectors x, y. Furthermore, $x \text{ and } y$ are expected to possess the same length of n. Where the autocorrelation measures the similarity between two sequences, while subtracting the measured similarity from the highest possible value of resemblance (+1) represents another way of calculating the difference between two sequences $z(t) = 1 - r_{xy}$.

The tables demonstrated below show some of the threshold values selected to find the optimal threshold necessary for the sensor fault detection, corresponding to their precision, recall, f1-score, and accuracy using the tradition signal difference, and our proposed correlation complement.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

Table 2. Signal Difference Thresholds and Their Metrics.

Threshold	0.1	0.3	0.5	0.7	0.9	1.1
Precision	0.78	0.76	0.74	0.69	0.65	0.79
Recall	0.89	0.77	0.69	0.54	0.33	0.2
F1-Score	0.83	0.76	0.71	0.61	0.43	0.32
Accuracy	0.71	0.62	0.55	0.44	0.32	0.32

Table 3. Signal Difference using the Correlation and Their Metrics

Threshold	0.1	0.3	0.5	0.7	0.9	1.1	1.3
Precision	0.79	0.8	0.81	0.83	0.84	0.85	0.86
Recall	0.95	0.85	0.82	0.72	0.64	0.55	0.5
f1-score	0.86	0.82	0.81	0.77	0.73	0.66	0.63
Accuracy	0.76	0.71	0.7	0.66	0.61	0.56	0.53

Based on the values shown in table 2 and table 3. The optimal threshold based on each signal difference metric can be easily detected by choosing the threshold that provided the best precision, recall trade-off. It is apparent that the threshold of 0.3 is the optimal threshold when using the regular signal difference metric, and the accuracy of the LSTM autoencoding sensor fault detection when using the optimal threshold of 0.3 is 0.62. On the other hand, the optimal threshold when using the signal difference based on the correlation is 0.5, and the accuracy of the sensor detection given the optimal threshold is 0.71.

As visualized in Figure 3 and Figure 4. The optimal threshold is the one providing the best precision, recall trade-off also known as f1-score. The optimal threshold can be easily observed as the intersection point between the three metrics mentioned previously. Based on the visualization in Figure 3, the threshold selected is 0.3, which provide the detection with 0.62 accuracy. Compared to the intersection point shown in Figure 4, where the threshold is chosen 0.5, and the corresponding accuracy is observed higher at 0.71. This concludes the accuracy of the proposed signal difference measure comparing to the traditional one, to achieve fault detection using signal reconstruction technique.

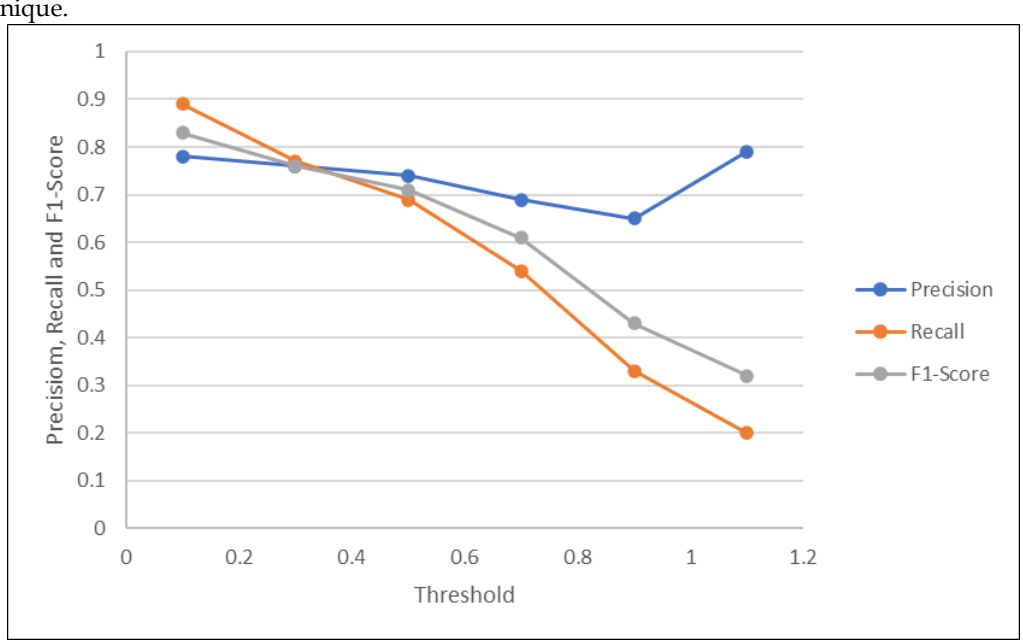

Figure 3 Optimal Threshold Selection Using Regular Signal Difference

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

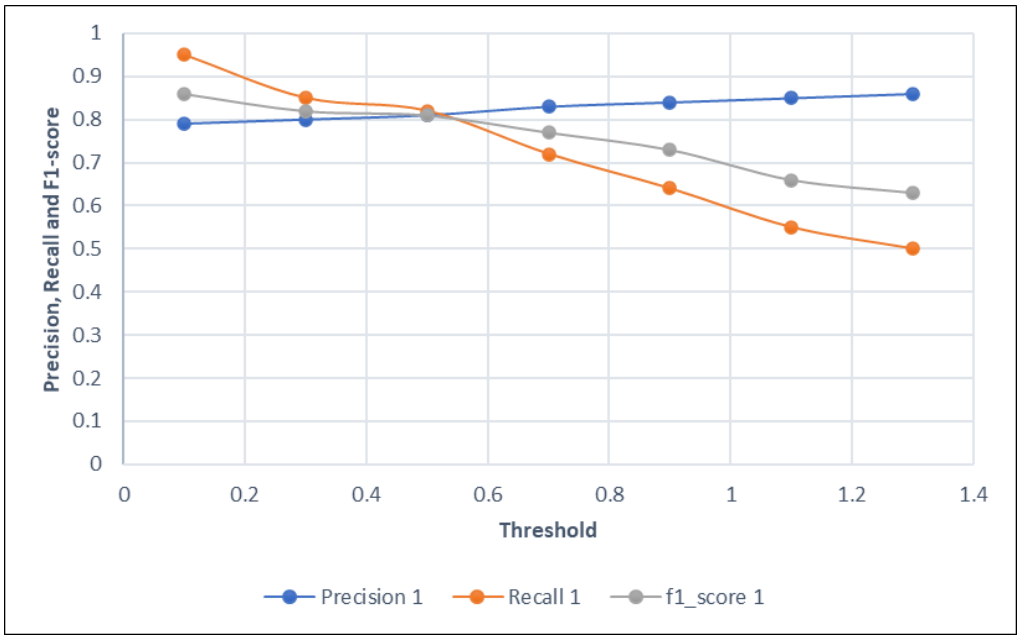

Figure 4 Optimal Threshold Selection Using Signal Difference Based on Correlation Complement.

4.1.2. Sensor Fault Diagnosis: Classification Schema

In this section, the experimental results conducting sensor fault diagnosis using a variety of supervised learning algorithms is demonstrated. As shown in Figure 2, the second phase following the detection of existing anomalies is applying the necessary means to diagnose their nature. The detected faults by the previous phase using the LSTM autoencoder, are then fed into a fault classification schema to determine the type and nature of the detected faults. In other words, to perform the fault diagnosis only faulty data is classified.

In this work, the classification results are compared when various feature engineering approaches are applied. The feature engineering approaches used for this section are: PCA, Feature Importance (FI), manually extracted time-domain features, and new cluster-based feature selection method called R*k*SE. Feature selection or extraction when applied to univariate datasets in a shape of sliding windows, is simply considered as a window compression method, to minimize the size of the readings provided by each window, and select the features with most contribution to the learning process. Therefore, the time and complexity constraints of the ML or DL models can be managed and minimized with smarter choice of features. Various ML and DL classifiers has been trained, validated, and tested, individually with the selected features using a diversity of feature engineering approaches. Then their results are documented and compared. The ML approaches used are LR, LDA, KNN, CART, NB, SVM and RF. The DL approaches selected to perform the classification tasks are CNN and LSTM.

The following experimental results tackle each of the feature engineering process and their results, when fed into the mentioned above ML and DL classifiers, to eventually achieve the FDD for sensor faults, using PS1 sensor as an example of sensors in the hydraulic test rig system.

The parameters selection for each classifier was chosen by trial and error, to ensure the highest possible accuracy when validating using 10-fold cross validation over the original data without any feature engineering applied. The table shown below describes the applied ML classifiers and their corresponding parameters using Scikit in Python. Furthermore, the mean accuracy for the 10-folds and the standard deviation corresponding to the 10-folds is calculated.

The CNN classifier has the parameters verbose, epochs and batch size of zero, hundred and 20, respectively. The parameters are chosen by try and error to provide the designed deep neural network with the highest 10-fold classification accuracy. The CNN applied is designed as sequential model (input, hidden layers, and output). The CNN convolutional layer applied here is a 1-D layer

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

since the training dataset is a time-series data and of a one dimensional nature, unlike the usual application of CNN where the data is typically of two-dimensional shape such as, images. CNN design included 6 one-dimensional convolutional layers of filter equals to 64 and kernel size of one. The kernel size that shows the length of the convolution window/masking window required for the convolution is selected as one. The number of convolution layers is added to guarantee highest possible accuracy, and by try and error it is set to 6 layers of 1-D convolutional layers. The activation function within the created layers is ReLU. The next process following each convolution layer is the pooling layer, in this work the pooling function is selected as the maximum pooling, which indicates selecting the maximum entry in the kernel during the pooling phase. Two fully connected layers are added following the pooling phase, one of size hundred and their activation function is ReLU. The second fully connected layer has three outputs to match the number of classification outputs/faults designated for the training, while its activation function is selected as SoftMax. Finally, the CNN optimizer chosen is Adam optimizer. The LSTM model designed for classification differs from the one used in the previous step, as this model is a classifier while the previous LSTM model is an autoencoder designed to solve multi-regression problems not classification. Only one batch of LSTM neurons is used, this batch has hundred sequential hidden layers or neurons. The layers are fully connected using a dense layer of size one hundred with the activation function ReLU, which is connected to another fully connected dense layer of size three (to match the number of outputs expected from the LSTM model), with the activation function SoftMax. The LSTM classifier parameters are represented by verbose, epochs and batch size which are equal to zero, 10 and 20, respectively.

Here are the classification results when using number of ML and DL classifiers when fed with only faulty data, to perform PS1 fault diagnosis and classification. In table 4, nine ML and DL classifiers are trained separately using five different features at a time, which are selected/extracted using PCA, manually extracted time-domain features, FI, R*k*SE. As well as the entire faulty dataset without any feature selection. The number of features without feature selection is equivalent to the window size of 60. Four features are extracted using PCA, 45 features are selected using FI, compared to 46 features selected using R*k*SE. Finally, four time-domain features extracted from each window, represented by the mean, variance, standard deviation, and signal to noise ration. The number of features selected by each method is the one with the highest fault classification mean accuracy.

Table 4. Classification Accuracy for Different ML and DL approaches using various Feature Engineering Methods.

Classifier	no feature selection	PCA	Time-Domain Features	FI	R*k*SE
LR	0.6911	0.6356	0.2425	0.7019	0.6882
LDA	0.7053	0.6535	0.7859	0.7038	0.7068
KNN	0.8747	0.9128	0.9625	0.8758	0.8816
CART	0.8972	0.9818	0.9951	0.9126	0.9116
NB	0.7089	0.6919	0.4821	0.6930	0.6824
SVM	0.9125	0.8827	0.7298	0.9112	0.9142
RF	0.8189	0.8390	0.9402	0.8196	0.8193
CNN	0.8773	0.8486	0.7575	0.8385	0.8562
LSTM	0.8352	0.9568	0.9684	0.7278	0.7499
Mean Feature Accuracy	0.81	0.82	0.76	0.80	0.80

When observing each row in respect to each feature engineering method, it can clearly show the feature engineering approach giving the highest or lowest 10-fold mean accuracy corresponding to each classification method. The mean feature accuracy row shows the overall accuracy for each

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

feature engineering approach in respect to all ML and DL classifiers combined. It is shown that PCA has the highest mean feature engineering accuracy when applied to the nine classifiers, which proves the consistency of PCA and its validity with different classification techniques. It is also obvious that the time-domain features selected are the one providing highest accuracy to many of the ML and DL classifiers, which are LDA, KNN, CART, RF and LSTM. However, this feature selection technique does not provide consistency in the accuracy results, since LR and NB classifiers for example show exceptionally low performances when applying the four selected time-domain features, comparing to the rest of feature engineering methods. This explains why time-domain features result in lower overall mean feature accuracy comparing to PCA, even though more classifiers have the maximum accuracy when applying time-domain features.

The selection of the suitable feature engineering method is highly dependent on each classifier type and its functionality. The table above has the purpose of investigating the behavioral changes of some of the most common ML and DL classifiers in respect to various commonly used feature engineering methods with time-series datasets. Furthermore, finding the best pair of features and classification approach which provides the most optimal accuracy-complexity trade-off when performing sensor fault detection is the number one aim of these comparisons. As a result, the highest measured sensor fault detection combination is when applying CART using time-domain features, followed by LSTM, KNN and RF using the same extracted features.

4.2. Experiment Two: Component FDD Using the Joint LSTM Autoencoder and Classifier Approach

In this experiment, component faults existing in the hydraulic test rig are detected and diagnosed using a unique approach, in which the detection and diagnosis stages are done separately to ensure more accurate detection of rare occurrences. Figure 5, shows the framework of this experiment.

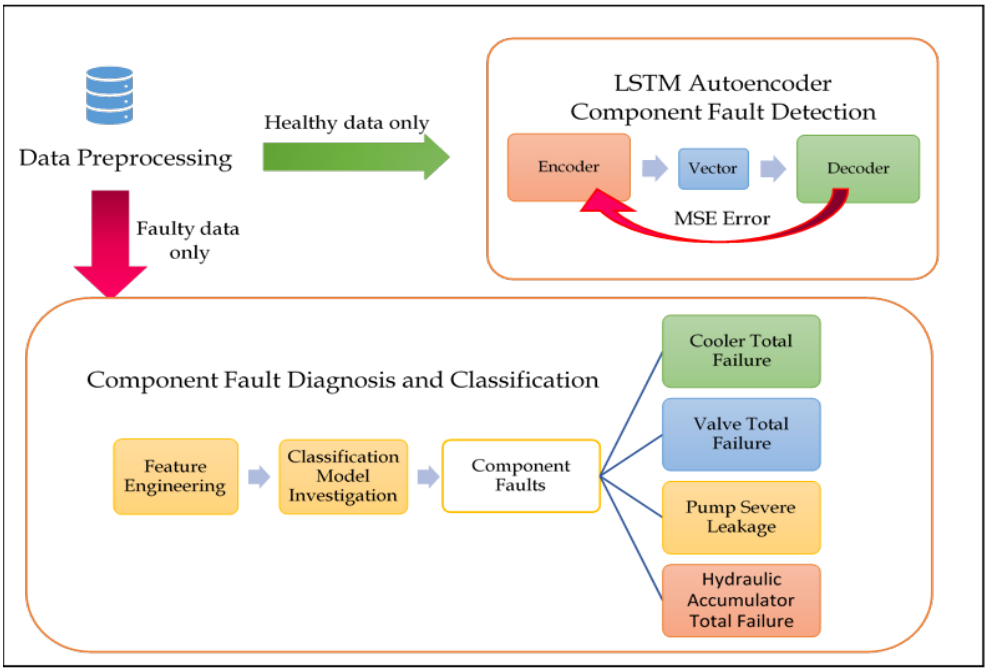

Figure 5 Component FDD Comprehensive Framework

The same steps and parameters created in experiment one (sensor FDD) are repeated in this experiment, excluding the data pre-processing and structuring, and the fault injection schema. The pre-processing step here differs from the previous experiment, since this time it is a multi-variate autoencoding and classification problem, without the application of sliding windows. Moreover, no fault injection is required in this experiment because the component faults studied are already available in the hydraulic test rig dataset used for this experiment. in this section, the data used is the

 doi:10.20944/preprints202012.0058.v1

hydraulic test rig dataset of eleven sensors, which indicates that this experiment is a multi-variate FDD experiment. The healthy data is applied for the detection as the first step of the FDD system represented by the LSTM autoencoder. While the faulty data containing four main component faults; cooler, value and hydraulic accumulator total failures, and pump severe leakage fault, are used in the second stage represented by the fault diagnosis using the supervised ML and DL methods.

In both stages, the data is organized as a 2D matrix of samples and the features expressed by the eleven sensors and their readings at different time points.

4.2.1. Component Fault Detection: LSTM Autoencoder

This section has the same procedure explained in experiment one for fault detection in sensors. The LSTM autoencoder for fault detection stage has the following main steps: (1) design the LSTM autoencoder to fit the problem. (2) Prepare the data into a form acceptable in the LSTM. (3) Train and validate the LSTM autoencoder using healthy data only and calculate the MSE and other error metrics. (4) predict the samples that contain fault and healthy readings. (5) calculate signal difference between the original samples and the predicted samples, using the regular difference and the Pearson's correlation one, to establish accuracy comparisons. (6) Find the best threshold of sequence difference to ensure the best trade-off between precision, recall and f1-score. (7) make component fault detection decisions using the trained, validated LSTM autoencoding model, and their calculated sequence difference compared to the computed threshold.

For training the designed model, 1438 samples of the eleven sensors' reading are used to train and validate the model. The data should be normalized between zero and one, as well as converted to a three-dimensional tensor format (samples, time points for LSTM to remember, number of features) before applying to the LSTM model. The LSTM model designed for component FDD detection is an autoencoder of a sequential hundred hidden LSTM neurons using the activation function ReLU, then a fully connected dense layer of size equal to the number of sensors or features is added. The dense layer contributes to improving the accuracy of the LSTM model, as well as making sure to the LSTM model generates outcomes equal in size with the designated input signal. Finally, the optimizer applied for the LSTM model is Adam. The training parameters of the LSTM autoencoder are epochs= 100 and batch size= 30.

After training and validating the designed model over a hundred epochs, the MSE error of the last epoch is 0.00057 and the MAE is equal to 0.0096. Both error metrics are exceedingly small, which is a high indication of the validity and accuracy of the created model in reconstructing healthy input sequences.

To select the optimal threshold corresponding to the allowed sequence difference between the original sequence and the reconstructed one resulted from the LSTM autoencoder. 4800 samples of size eleven are predicted using the autoencoder, to reconstruct 4800 healthy versions of the prediction samples. The signal difference using between each of the corresponding sequences in the original and reconstructed sequences are computed using (1) The traditional signal subtraction to find the signal difference as a vector, then find the magnitude of this vector. (2) The sequence difference using (1-Pearson's autocorrelation) as a measurement created in this work and proposed to be more accurate measurement for fault detection than the traditional signal subtraction.

To avoid repeating the explanation of each signal difference methods, we will jump right through the results and their comparisons.

A pool of candidate threshold values is created, then the labels of the 4800 prediction samples were obtained based on each threshold in the pool, if the value of the signal difference is higher than the threshold a fault is considered to be detected, thus label 1, else label 0. The precision, recall, and f1-score are computed to each threshold in the pool based on the generated labels and the original labels of the given prediction samples. When applying signal difference using the Pearson's autocorrelation complement, the pool of chosen thresholds between the minimum and maximum observed values are shown in table 5. Furthermore, for each chosen threshold the accuracy, precision, recall and f1-score are computed. Figure 6, illustrates the process of choosing the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

component fault detection threshold based on the precision, recall and f1-score trade-off shown in the table below. As clearly shown in Figure 6, the selected threshold is the intersection between the three curves, which is approximately equal 0.0007 and the accuracy observed for this threshold value is 0.71.

Table 5. The Thresholds of Pearson's Correlation Difference and Their Corresponding Fault Detection Accuracy, Precision, Recall and F1-Score

Threshold	0.0001	0.0003	0.0005	0.0007	0.0009	0.001	0.003	0.005
precision	0.63	0.73	0.77	0.78	0.79	0.81	0.95	0.93
recall	1	0.79	0.74	0.74	0.72	0.68	0.25	0.14
F1-Score	0.77	0.76	0.75	0.76	0.75	0.74	0.39	0.24
Accuracy	0.63	0.69	0.7	0.71	0.7	0.69	0.52	0.46

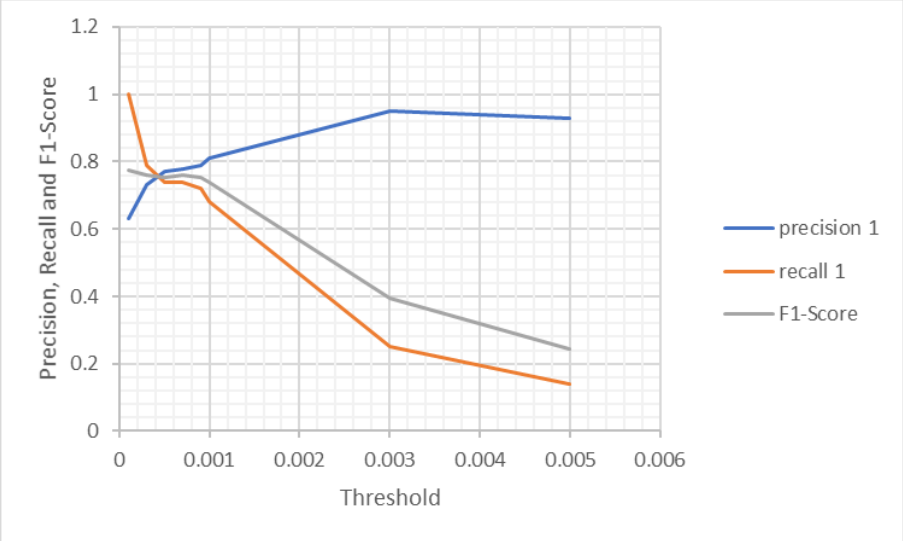

Figure 6 Precision, Recall and f1-Score trade-off for Threshold selection using Pearson's Correlation Difference.

On the other hand, the optimal threshold is also calculated when the optimal signal difference is applied. In Figure 7 the intersection between the three curves is demonstrated. It is shown clearly that the optimal threshold for component fault detection using the sequence subtraction is approximately 0.03, with the fault detection accuracy of 0.69. The optimal accuracy using signal subtraction of 0.69 is less than the measured one using the optimal threshold computed by Pearson's correlation of 0.71. As a result, when comparing the accuracy of the optimal thresholds selected using two different signal deviation measurements, which are the autocorrelation complement and the traditional subtraction. It is clearly shown that the proposed method using Pearson's correlation complement guarantees higher component and sensor faults detection accuracies, comparing to its commonly used traditional subtraction counterpart, based on the measured comparisons in experiment one and two.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

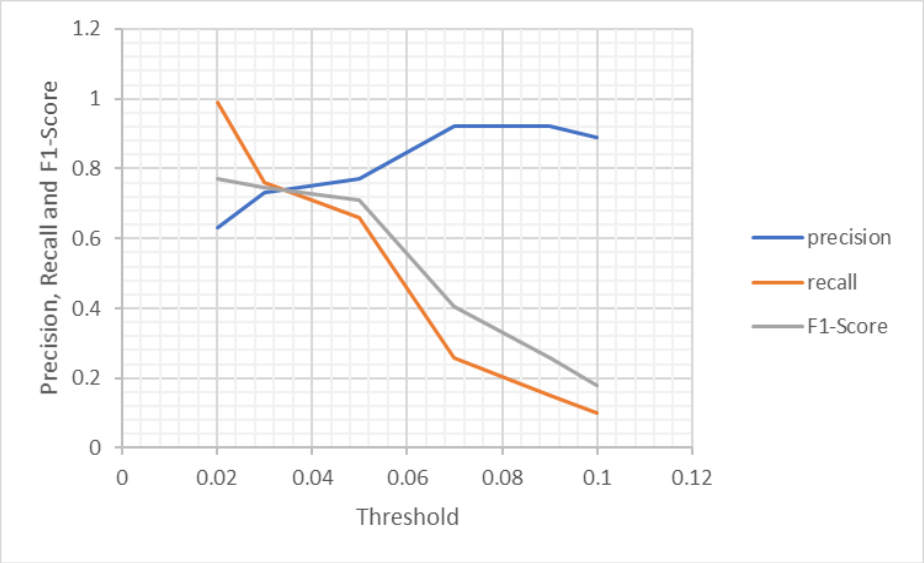

Figure 7 Precision, Recall and f1-Score trade-off for Threshold Selection Using Signal Subtraction Difference.

4.2.2. Component Fault Diagnosis: Classification Schema

In this section, the feature engineering methods compared are FI, PCA and R*k*SE. The time-domain extracted features applied for multi-variate time series sequences without the application of sliding windows, are expected to have lower accuracy values regardless the ML or DL classifier used. Hence, it does not make sense to compute the mean, standard deviation, and variance to a sample of readings extracted from sensors of different nature. However, the time-domain features were extracted and applied to all the classifiers anyways, to prove the point mentioned earlier.

The optimal number for each feature engineering method is computed for each method to ensure the best accuracy and complexity trade off. The overall number of features in each sample is eleven, corresponding to each sensor in the hydraulic test rig the optimal features for FI, PCA, time-domain features and RKSE are four, five, four and nine, respectively.

The accuracies computed for all the ML and DL classifiers are results of dividing the fault data with component faults, into training and testing data, with percentages of 80% to 20% of the faulty data, respectively. Followed by applying 10-fold cross validation technique for each classifier separately.

The parameters and optimizers for each ML method used in this section, are identical to the ones used in the sensor FDD experiment earlier in this paper. Moreover, some minor changes in the CNN and LSTM classifiers' design and parameters has been made comparing to the previous experiment.

CNN design consists of only one 1D CNN layer of filter size 64, and kernel size of one. The layer is sequential which means the input layer is directly connected to the hidden layer(s) that is connected to the output layer. The activation function used is ReLU. Followed by the pooling layer that has pooling size of one and applies maximum pooling as the pooling function. Finally, a fully connected dense layer of size equivalent to the number of expected outputs, with SoftMax activation function is created to make the classification process for the extracted features during the convolutional and pooling layers. The CNN optimizer used is Adam, as a stochastic gradient descent approach to optimize the network. The LSTM classifier applied has only one LSTM batch with two hundred hidden neurons that are sequential in order and nature. Followed by a fully connected dense layer of SoftMax activation function, and naturally Adam is the applied optimizer for the LSTM as well. The verbose, epochs and batch size parameters are applied by testing various values and their effect on the classification accuracy, and they are set to zero, 10 and 20, accordingly.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 December 2020 doi:10.20944/preprints202012.0058.v1

The table below comprehend the component fault classification results when trained by faulty hydraulic test rig data, applying various ML and DL approaches using numerous feature engineering methods.

Table 6. Component Fault Diagnosis Using Various Feature Engineering and Classification Approaches.

METHOD NAME	FI	PCA	R*k*SE	TIME-DOMAIN FEATURES	NO FEATURE SELECTION
LR	0.9962	0.7300	0.7823	0.37599	0.6832
LDA	0.7634	0.7490	0.7528	0.370521	0.7031
KNN	0.9940	0.9229	0.9320	0.831458	0.8677
CART	0.9932	0.9435	0.9912	0.928594	0.6849
NB	0.9924	0.7510	0.7122	0.39526	0.9035
SVM	0.9859	0.9337	0.9310	0.833281	0.8139
RF	0.9930	0.9013	0.9910	0.871042	0.9042
CNN	0.7343	0.8427	0.7343	0.3971	0.7385
LSTM	0.7375	0.8770	0.7375	0.3981	0.73124
MEAN ACCURACY	0.910	0.850	0.840	0.600	0.781

As shown in table 6, feature selection approaches work better than extraction ones such as, PCA and time-domain features when dealing with traditional ML classifiers, to classify multi-variate time series datasets. On the one hand, FI is consistently showing highest accuracy results comparing to the rest of the feature engineering methods when dealing with traditional ML classifiers. Followed by R*k*SE, that is yet slightly lower accuracy than FI for traditional ML approaches, but it shows consistency in all ML classifiers. Moreover, PCA has shown the highest accuracy when applying DL classification algorithms comparing to other feature selection approaches. While FI and R*k*SE are neck to neck when it comes to the classification accuracy using the selected DL approaches. Finally, even though time-domain features were the most accurate ones when applied to sliding-windows for univariate classification as shown in experiment one. As spotted earlier in this experiment, when it comes to extracting time-domain features from multi-variate datasets without applying sliding windows, this feature extraction method is proven weaker than the rest of the approaches, when combined to regardless ML or DL classifiers.

In another side of comparison, KNN, CART, RF and SVM showed great consistency in high accuracy results no matter what feature engineering method is applied, including time-domain features regardless its weakness. CNN and LSTM had lower accuracies comparing to the traditional ML approaches mentioned earlier, and their accuracies drops radically when time-domain features are applied.

To conclude experiment two, it is important to know how to apply the trained models and saved parameters from experiment two at run-time, to make new real-time predictions. The input vector for prediction should have one readings of each sensor of the eleven sensors used for the training process during the offline or training phase. (1) fault detection: fault detection for new samples can be done by; feeding the new sample into the trained and validated LSTM autoencoder model to reconstruct the healthy form of the sequence. Then the reconstructed sequence and the original one is compared by calculating the signal difference using Pearson's autocorrelation. Finally, with the signal difference calculated is above or below the trained threshold, this will determine the existence of faults or not. (2) Fault diagnosis: in case a fault is being detecting using the detection step. The fault should be diagnosed by applying the necessary feature engineering approaches, then feeding the processed new sample to the highest accuracy ML or DL trained

 doi:10.20944/preprints202012.0058.v1

classifier, suitable to the features chosen. In our case, based on the results shown in table 6, it is more accurate to use RF a combined with FI to guarantee better accuracy and complexity trade-off.

4. Discussion

In this section a two-staged FDD approach is proposed. Where the detection and diagnosis are separated into two stages to guarantee detecting rare occurrences and events in hydraulic systems. The detection process is represented by a LSTM autoencoder that learns only from healthy observations, in an attempt to reconstruct the healthy version of the given sequences, sensor window readings or multi-sensors readings. Followed by the comparison between the given sequences and their healthy reconstructed version, to measure the deviation from the healthy state and the given sequences, which is vital to detect the existence of faults and malfunctions when this deviation exceeds a certain, learned threshold. The fault diagnosis is represented by a classification process that can be a ML or DL algorithm, which is trained using only the faulty observations captured by the detection stage using LSTM autoencoder.

The proposed approach is beyond the state-of-the-art methods by the following:

- The proposed method is applied into two entirely different experiments, with different data pre-processing, acquisition and structuring, different DL algorithmic designs, and above all to detect two different fault types: sensor faults and component faults. The methods proposed in the literature only focuses on one fault type, either component faults or sensor faults. However, it is rarely seen that any work shows comprehension in detecting or diagnosing different fault types at once.

- In the detection phase using LSTM autoencoder, some changes are made from the existing related work. The most important addition is using the sequence difference calculated by subtracting the Pearson's autocorrelation from one. The detection results using the complement of Pearson's autocorrelation comparing to the traditional signal difference measure applied in the state-of-the-art research is proved experimentally. In experiment one to detect sensor faults, the detection accuracy using signal difference is observed as 62%, comparing to the detection accuracy when applying our proposed measurement of signal difference, the detection accuracy is observed as 71%. Moreover, in experiment two for component fault detection, the accuracies observed using traditional signal difference, and the proposed one are 69% and 71%, respectively. The results of the detection phase in the two conducted experiments prove the superiority of the proposed signal difference measurement comparing to the traditional subtraction of signals to provide signal difference.

- Investigating various feature engineering approaches and pairing them with numerous ML and DL methods in the diagnosis phase, to determine the most suitable feature engineering method to classifiers of different functionality and design. Furthermore, this pairing gives the opportunity to see how each classifier reacts with different feature engineering methods of different procedure, which would help future work researchers to select the best match pair or avoid the worst pair for both data structures; windows univariate or no window multi-variate. For example, in experiment one when dealing with sensor faults in a sliding window data structure, it was noticeable that the chosen time-domain features shown the highest diagnosis accuracy of almost all the classifiers. i.e. LDA, KNN, CART, RF and LSTM. Although the mean accuracy of all classifiers using PCA was computed the highest of 82%, which is justified by the consistency PCA shows with all the classifiers regardless their functionality. However, time-domain extracted features show extremely low diagnosis accuracies when applied to some classifiers such as LR and NB with the detection accuracies of only 24.25% and 48.21%, respectively. Which explains why the time-domain extraction technique is not the highest mean accuracy even though it provides the highest accuracy to the majority of the supervised methods. On the other hand, in experiment two when component faults were classified using multi-variate sensor's readings without the application of sliding windows, FI

showed the highest diagnosis accuracy for all the ML classifiers, and PCA shows the highest diagnosis accuracy when combined to DL such as, CNN and LSTM.

- In the related work, the diagnosis phase is represented by some chosen type of classifiers combined with a chosen set of features, without any analysis or investigation in respect of other classifiers or features. In this work, after careful experimental observations and calculations, the appropriate features and their suitable classifier is used to represent the diagnosis phase for our algorithm. In experiment one (univariate sliding window structure) the diagnosis phase is chosen using the time-domain extracted features combined with either CART or LSTM classifiers with the diagnosis accuracies of 99.51% and 96.84%. However, in experiment two, when dealing with multi-variate features without the application of sliding windows, FI combined with almost all ML classifiers showed extremely high accuracies exceeding 98%. Thus, RF combined with FI is the best combo used to perform multi-variate diagnosis, especially when FI can be done implicitly during the RF training stage based on FI nature, which can help in reducing time and computational complexities.

In conclusion, the proposed approach is used for the first time in the field of industry 4.0 especially when applied to hydraulic test rigs. Although, the proposed work in this chapter has multiple changes and valuable improvements from the state-of-the art, there is always a place for improvements and further expansions. For future work, it can be a good challenge to improve experiment one, by designing a LSTM autoencoder that can learn multiple sequences of multiple windows that belongs to different sensors at a time applying one LSTM autoencoding model. For example, in experiment one only one sensor at a time (PS1) is used in a sliding window format to train the LSTM autoencoder to predict/reconstruct the healthy window of the given PS1 window of size 60. In the current approach we must train the LSTM autoencoder for each sensor separately to learn how to reconstruct the healthy window of each. However, A multi-variate approach, multi-sequences, multi-sensor deep neural network design would be a sophisticated approach in the future, where maybe a number of LSTM autoencoder batches can be connected together sequentially or in parallel so each can train on different sensor window sequences. Furthermore, the feature engineering methods applied to the diagnosis phase are all in time-domain. Investigating frequency-domain or the combination of time and frequency domains, such as applying Wavelet Coefficient Packer Decomposition (WPD) would add different perspective for the future of FDD in mechanical equipment's.

Author Contributions: Conceptualization, A.M. and M.F.; methodology, A.M.; software, A.M.; validation, M.F.; formal analysis, A.M.; investigation, A.M.; resources, M.F.; data curation, A.M.; writing—original draft preparation, A.M.; writing—review and editing, M.F.; visualization, A.M.; supervision, M.F.; project administration, M.F.; funding acquisition, M.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the DFG research grants LO748/11-1 and OB384/5-1.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Precup, R.-E.; Angelov, P.; Costa, B. S. J.; Sayed-Mouchaweh, M. An Overview on Fault Diagnosis and Nature-Inspired Optimal Control of Industrial Process Applications. *Computers in Industry* **2015**, *74*, 75–94. https://doi.org/10.1016/j.compind.2015.03.001.
2. Wang, P.; Guo, C. Based on the Coal Mine ' s Essential Safety Management System of Safety Accident Cause Analysis; 2013.
3. UCI Machine Learning Repository: Citation Policy https://archive.ics.uci.edu/ml/citation_policy.html (accessed Feb 4, 2020).
4. Park, P.; Marco, P. D.; Shin, H.; Bang, J. Fault Detection and Diagnosis Using Combined Autoencoder and Long Short-Term Memory Network. *Sensors (Basel)* **2019**, *19* (21). https://doi.org/10.3390/s19214612.
5. chen, xiaolu. Tennessee Eastman Simulation Dataset, 2019.

6. Lu, C.; Wang, Z.-Y.; Qin, W.-L.; Ma, J. Fault Diagnosis of Rotary Machinery Components Using a Stacked Denoising Autoencoder-Based Health State Identification. *Signal Processing* **2017**, *130*, 377–388. https://doi.org/10.1016/j.sigpro.2016.07.028.
7. Li, Z.; Li, J.; Wang, Y.; Wang, K. A Deep Learning Approach for Anomaly Detection Based on SAE and LSTM in Mechanical Equipment. *Int J Adv Manuf Technol* **2019**, *103* (1), 499–510. https://doi.org/10.1007/s00170-019-03557-w.
8. Chen, Z.; Deng, S.; Chen, X.; Li, C.; Sanchez, R.-V.; Qin, H. Deep Neural Networks-Based Rolling Bearing Fault Diagnosis. *Microelectronics Reliability* **2017**, *75*, 327–333. https://doi.org/10.1016/j.microrel.2017.03.006.
9. Shao, H.; Jiang, H.; Zhao, H.; Wang, F. A Novel Deep Autoencoder Feature Learning Method for Rotating Machinery Fault Diagnosis. *Mechanical Systems and Signal Processing* **2017**, *95*, 187–204. https://doi.org/10.1016/j.ymssp.2017.03.034.
10. Junbo, T.; Weining, L.; Juneng, A.; Xueqian, W. Fault Diagnosis Method Study in Roller Bearing Based on Wavelet Transform and Stacked Auto-Encoder. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*; 2015; pp 4608–4613. https://doi.org/10.1109/CCDC.2015.7162738.
11. Shao, H.; Jiang, H.; Wang, F.; Zhao, H. An Enhancement Deep Feature Fusion Method for Rotating Machinery Fault Diagnosis. *Knowledge-Based Systems* **2017**, *119*, 200–220. https://doi.org/10.1016/j.knosys.2016.12.012.
12. Verma, N. K.; Gupta, V. K.; Sharma, M.; Sevakula, R. K. Intelligent Condition Based Monitoring of Rotating Machines Using Sparse Auto-Encoders. In *2013 IEEE Conference on Prognostics and Health Management (PHM)*; 2013; pp 1–7. https://doi.org/10.1109/ICPHM.2013.6621447.
13. Chen, Z.; Li, W. Multisensor Feature Fusion for Bearing Fault Diagnosis Using Sparse Autoencoder and Deep Belief Network. *IEEE Transactions on Instrumentation and Measurement* **2017**, *66* (7), 1693–1702. https://doi.org/10.1109/TIM.2017.2669947.
14. Sun, W.; Shao, S.; Zhao, R.; Yan, R.; Zhang, X.; Chen, X. A Sparse Auto-Encoder-Based Deep Neural Network Approach for Induction Motor Faults Classification. *Measurement* **2016**, *89*, 171–178. https://doi.org/10.1016/j.measurement.2016.04.007.
15. Shao, H.; Jiang, H.; Lin, Y.; Li, X. A Novel Method for Intelligent Fault Diagnosis of Rolling Bearings Using Ensemble Deep Auto-Encoders. *Mechanical Systems and Signal Processing* **2018**, *102*, 278–297. https://doi.org/10.1016/j.ymssp.2017.09.026.
16. Huang, Y.; Chen, C.-H.; Huang, C.-J. Motor Fault Detection and Feature Extraction Using RNN-Based Variational Autoencoder. *IEEE Access* **2019**, *7*, 139086–139096. https://doi.org/10.1109/ACCESS.2019.2940769.
17. Welcome to the Case Western Reserve University Bearing Data Center Website | Bearing Data Center https://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website (accessed Oct 19, 2020).
18. NASA | Open Data | NASA Open Data Portal https://nasa.github.io/data-nasa-gov-frontpage/ (accessed Oct 19, 2020).
19. Hui-jie, Z.; Ting, R.; Xin-qing, W.; You, Z.; Hu-sheng, F. Fault Diagnosis of Hydraulic Pump Based on Stacked Autoencoders. *2015 12th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)* **2015**. https://doi.org/10.1109/ICEMI.2015.7494195.
20. Mallak, A.; Fathi, M. Unsupervised Feature Selection Using Recursive K-Means Silhouette Elimination (RkSE): A Two-Scenario Case Study for Fault Classification of High-Dimensional Sensor Data. **2020**. https://doi.org/10.20944/preprints202008.0254.v1.