

Article

Artificial-Intelligence-Based Condition Monitoring of Industrial Collaborative Robots: Detecting Anomalies and Adapting to Trajectory Changes

Samuel Ayankoso ^{1,†,*}, Fengshou Gu ^{1,†}, Hassna Louadah ^{2,†}, Hamidreza Fahham ^{2,†} and Andrew Ball ^{1,†}

¹ Centre for Efficiency and Performance Engineering, University of Huddersfield, Huddersfield HD1 3DH, UK; a.ball@hud.ac.uk (A.B.)

² Institute of Railway Research, University of Huddersfield, Huddersfield HD1 3DH, UK; h.louadah@hud.ac.uk (H.L.); h.faham@hud.ac.uk (H.F.)

* Correspondence: samuel.ayankoso@hud.ac.uk

† These authors contributed equally to this work.

Abstract: The increasing use of collaborative robots in smart manufacturing, owing to their flexibility and safety benefits, underscores a critical need for robust predictive maintenance strategies to prevent unexpected faults/failures of the machine. This paper focuses on fault detection and employs multivariate operational data from a universal robot to detect anomalies or early-stage faults using test data from designed anomalous conditions and artificial-intelligence-based anomaly detection techniques called autoencoders. The performance of three autoencoders, namely, a multi-layer-perceptron-based autoencoder, convolutional-neural-network-based autoencoder, and sparse autoencoder, was compared in detecting anomalies. The results indicate that the autoencoders effectively detected anomalies in the examined complex and noisy datasets with more than 93% overall accuracy and an F1 score exceeding 96% for the considered anomalous cases. Moreover, the integration of trajectory change detection and anomaly detection algorithms (i.e., the dynamic time warping algorithm and sparse autoencoder, respectively) was proposed for the local implementation of online condition monitoring. This integrated approach to anomaly detection and trajectory change provides a practical, adaptive, and economical solution for enhancing the reliability and safety of collaborative robots in smart manufacturing environments.

Keywords: industrial collaborative robot; anomaly detection; trajectory change; autoencoder; dynamic time warping; online monitoring



Citation: Ayankoso, S.; Gu , F.; Louadah, H.; Fahham, H.; Ball, A. Artificial-Intelligence-Based Condition Monitoring of Industrial Collaborative Robots: Detecting Anomalies and Adapting to Trajectory Changes. *Machines* **2024**, *12*, 630. <https://doi.org/10.3390/machines12090630>

Academic Editor: Ahmed Abu-Siada

Received: 30 July 2024

Revised: 2 September 2024

Accepted: 4 September 2024

Published: 7 September 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

There is a tremendous rise and increasing interest in collaborative robots or cobots in the manufacturing industry. Cobots are created for tasks involving either human-robot interaction or proximity between the two (without needing safety cages) [1]. As we transition towards an era of smart manufacturing, cobots are being adopted in big and Small and Midsize Enterprise (SME) industries due to the benefits they have over traditional robots [2]. These benefits include their compact size in terms of weight and payload, ease of movement and installation, versatility for reprogramming to perform various tasks, and safe operation alongside humans [3]. Due to repetitive use and degradation of parts, these robots, like many other machines, are not free from faults, and their breakdown can have negative consequences on production. Therefore, implementing an effective maintenance strategy before the occurrence of faults is crucial for maintaining the reliability and safety of such industrial robots.

There have been so many changes in the way industrial/commercial machines are maintained over the years. Several factors have influenced such transitions including

the need to reduce cost and machine downtime, increase safety, and adopt modern technologies [1]. As a result, machine maintenance has evolved from reactive to preventive maintenance, and from preventive to predictive maintenance, as discussed in [4,5] and illustrated in Figure 1.

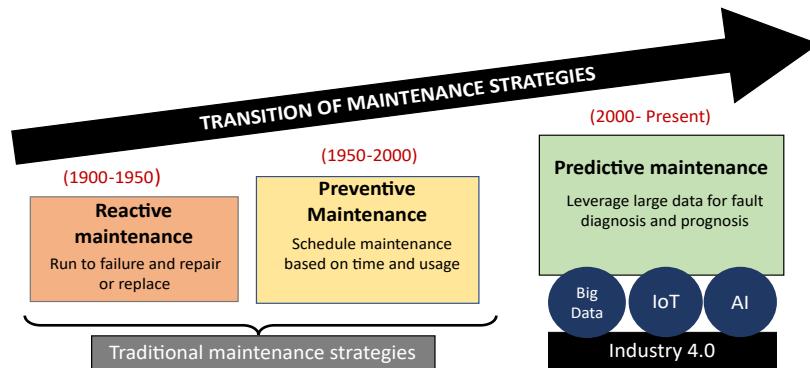


Figure 1. The transition of maintenance strategies.

Predictive maintenance (PM) is a maintenance philosophy targeted at improving the reliability and safety of industrial machines while driving down costs and reducing downtime. Due to the current phase of the industrial revolution (i.e., Industry 4.0), PM is crucial to smart machines, and its core drivers are Big Data, the Internet of Things (IoT), and Artificial Intelligence (AI) [1]. PM is enabled by Prognostics and Health Management (PHM). PHM is an engineering field that addresses the efficient detection, diagnosis, and prediction of faults in industrial machines [6].

PHM strategies are developed through models based on system physics, machine learning (ML), or deep learning (DL) [7]. The choice of any of the three models depends on the system's complexity, data availability, and the need for interpretability. Physics-based approaches offer high accuracy and reliability when dealing with a system whose underlying dynamic behavior and failure mechanisms are well understood. In contrast, ML and DL are data driven; ML requires manual feature engineering with the help of signal processing and expert knowledge to extract relevant patterns, while DL is capable of learning features automatically from raw or unprocessed data [6]. DL benefits from transfer learning, allowing models trained on large datasets to be adapted to new, smaller tasks, as demonstrated in [8]. Although DL often lacks transparency, explainable AI aims to address this issue by making DL model decisions more interpretable, improving trust in critical applications [9].

Different sensors are used in the PHM of industrial robots [10]. These sensors can be placed in the joints, end effector, or robot environment. The common inbuilt sensors are encoders and current transducers for measuring joint positions and motor input currents. The external sensors, on the other hand, include lasers, acoustic sensors, and cameras, but they are difficult to employ for PHM/condition monitoring in an industrial environment [11]. In addition, robot joints often do not come equipped with built-in vibration sensors or acoustic sensors, and they are usually installed in an intrusive way and at an extra cost. For example, a vibration sensor was externally installed in [12], while an acoustic one was installed on the joint surface in [13]. Built-in sensors are the most reliable option for diagnosing mechanical faults in motors and gears, as demonstrated through the use of joint internal current sensors for the six axes of a Hyundai robot in [14]. The data from the same sensors were leveraged in their subsequent works [15,16]. Cobots are designed with a variety of internal sensors that enhance their condition monitoring, as presented in [17] for a universal robot and [18] for a voraus robot.

The two major problems with the PHM of industrial collaborative robots are the lack of or insufficient fault data for training DL models and the inability of the models to generalize and adapt to different tasks and changes in operating conditions. First, it

is difficult to acquire fault data, as the average mean time before the failure of industrial robots is tens of thousands of hours, as reported in [19]. Generally, it is very expensive to induce faults in industrial robots, and, even if carried out, without in-depth knowledge about the system's fault types and degrees, the observed faults may differ from those that occur in real cobots. Furthermore, though the utilization of ML and DL for fault detection and diagnosis has garnered significant attention in the last decade, few related works can be found on the implementation of these data-driven methods. Additionally, most fault detection or diagnosis models are typically trained based on a specific trajectory, which is often not applicable to different operational scenarios or tasks where trajectory change occurs [20,21].

In this study, the detection of artificial anomalies in cobots is investigated. Anomaly detection is crucial for industrial robots to identify abnormal or unusual behaviors early, which can signal potential issues before they escalate into faults or failures. Better put, anomalies represent deviations from normal operating behavior but do not always indicate faults, as they could stem from small variations or changes in operating conditions rather than actual system malfunctions. The multivariate operational data obtainable from a universal robot (UR5e) were employed as a means of detecting anomalies from two groups of designed anomalous conditions. The first group of anomalies were achieved by placing different weights on the robot's shoulder arm while a vibrator with different speeds was used to create the second class of anomalies around the elbow joint. The contributions of this work include:

1. Comparison of three state-of-the-art anomaly detection techniques, which are the multi-layer-perceptron-based autoencoder (MLP-AE), convolutional-neural-network-based autoencoder (CNN-AE), and sparse autoencoder (SAE), in detecting anomalies associated with the overall operation of a cobot;
2. Integration of trajectory change and anomaly detection algorithms for local implementation of online condition monitoring.

Our results show that AEs are advanced unsupervised deep learning techniques, and they are very useful for anomaly detection using complex and noisy datasets (with no manual extraction of health indicators). Additionally, our work provides insights into the potential of combining both anomaly and trajectory change detection models for the practical, adaptive, and cost-efficient online condition monitoring of cobots in smart manufacturing settings. The rest of this paper is structured as follows: The related work is summarized in Section 2, while the anomaly and trajectory change detection models are introduced in Sections 3.1–3.3. In Sections 3.4–3.6, the test setup and theoretical analysis of the experimented anomalies, as well as data visualization and preprocessing, are discussed. The model training and evaluation are covered in Section 4, and Section 5 is about online condition monitoring. The Section 6 is the conclusion and future recommendations of this work.

2. Related Work

Fault detection, diagnosis, and fault prognosis are three important parts of a complete PHM system. In fault detection, the machine's condition is monitored to detect any deviation from its normal operating condition. Fault diagnosis deals with fault type or class and severity detection in machines or components. In contrast, fault prognosis focuses on estimating the remaining useful life (RUL) of such components. Table 1 shows a comparison of the three stages of the PHM process, and more details about each are available in [6,22]. In addition, a complete PHM process is described in Figure 2 which usually starts with data acquisition, and each cycle ends with decision making.

The major challenge in cobot condition monitoring research is the lack/unavailability of fault data because it is costly to induce faults, and different operating conditions of the robot need to be experimentally considered. For example, in [16], a robust fault diagnosis model was developed for one of the servo motors of a six-axis industrial robot under various operating conditions. The operating conditions of the robot included different

speeds (from 10 to 100% with 10% increment), loads (0 g, 500 g, 1000 g, 1500 g, 2000 g, 2500 g, 3000 g, and 3500 g), and motions (simple and complex). During the simple motion, all the robot joints were fixed except the third joint, which was allowed to move $-50 +50 +50 -50$. The motion of a welding task was adopted for the complex motion, in which all the joints of the robot moved independently. Moreover, while designing the experiment, two states were considered; the first was when the robot was operating normally, and the second state was when a bearing fault (inner race fault artificially created by grinding) was added to the third joint servo motor. The fault diagnosis of the harmonic reducer of an industrial robot was considered in [23] using time-domain vibration signals. A one-dimensional convolutional neural network with a matrix kernels framework was proposed and trained offline. The test stand consisted of three (3) fault-free industrial robots, six faulty harmonic reducers, and PV-571 orthogonal direction (XYZ) vibration sensors. The fault dataset was acquired by replacing the two joints of each of the fault-free industrial robots. The drawback of the two works ([16,23]) is the fact that it is expensive to obtain data relating to the joint's motor or gear fault of the robot.

Table 1. Comparison of the differences between fault detection, fault diagnosis, and fault prognosis in the context of condition-based and predictive maintenance for industrial collaborative robots.

Aspect	Fault Detection	Fault Diagnosis	Fault Prognosis
Objective	Identify the occurrence of fault or anomaly	Determine faulty conditions and their degree	Predict when failure or major fault would occur based on current condition
Data required	Healthy state sensor data	Healthy and fault states sensor data	Historical sensor data
Method	Residual models and threshold-based algorithms	Classification models	Time-series predictive models
Output	Alarm abnormal conditions	Show the class, type, or degree of fault that has occurred	The remaining useful life of machines/parts before failure
Examples	Abnormal trajectories, overheating, excessive vibrations	Identify failed motor or gear, sensor failure, and communication issue	Predict when to replace a cobot arm based on wear and usage patterns

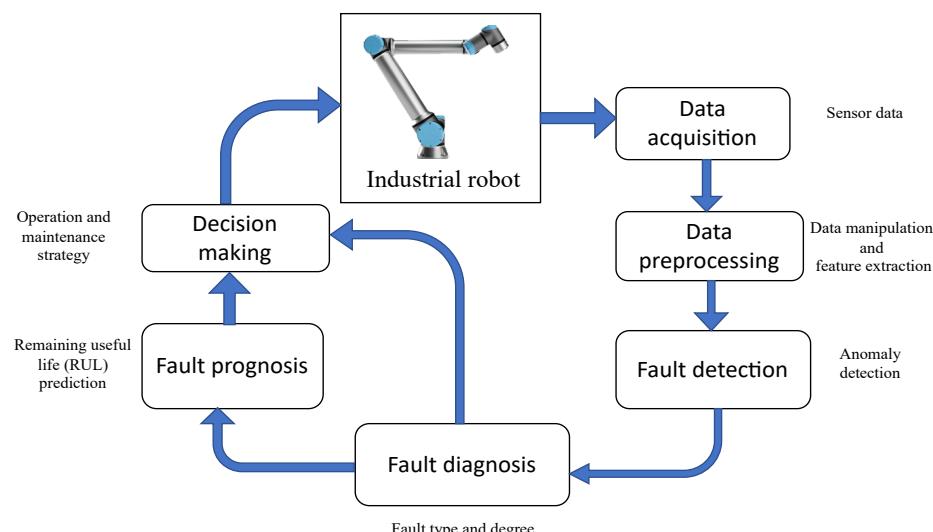


Figure 2. Cobot PHM process.

Given the challenge mentioned earlier, a potential solution is to create high-fidelity models capable of simulating faults, such as the framework outlined in [24] and the multibody model developed in [25]. Another solution is to focus on fault detection, as it has proven to be a more practical and cost-effective approach. The development of a fault detection model does not necessarily require fault data, and there is reduced expertise needed in its development compared to in fault diagnosis and prognosis.

There are a few existing studies on fault detection for collaborative robots to explore. First, a comparison of different anomaly detection methods for an industrial collaborative robot was performed in [17]. Fifteen anomaly detection methods were examined in the pick-and-place task using a cylindrical object and experimenting with different anomalies such as pushing joint 3 or 5 during the task, changing the moved object weight (from 0.2 kg to 0.5 kg, 1 kg, and 2.5 kg), object drop, applying a twisting force to the robot's Tool Center Point (TCP), and others. Some of the methods compared were the K-nearest neighbor technique, Principal Component Analysis (PCA), deep AE, variational AE, LSTM-based AE, and others. Their analysis showed that many methods gave high detection accuracy; however, the model accuracy was reduced when subjected to new program branches, non-deterministic trajectories, and deliberate contact with the environment. This means the developed detection models will underperform when there is a trajectory change, payload, or contact with the environment.

A framework that can inform an operator of anomalous trajectories was developed for a seven-axis collaborative robot. A two-level framework was proposed in [20]. The first level is where fault detection is performed through an algorithm, while the second level is a diagnostic check by a human operator to isolate the fault. The proposed algorithm for fault detection has three steps: (1) trajectory clustering; (2) health state identification; and (3) functional failure clustering. Three anomalous conditions were artificially induced to ascertain the effectiveness of the proposed framework, which were end effector overload, excessive friction in a joint, and random variations in the test data. The gap in the authors' work is that four trajectories and three failures were investigated, and they do not cover all the possible cases that the robot can be subjected to during operation. This suggests the fault detection system will be unreliable when an unknown trajectory is used.

In [26], the authors proposed a model-based method for the condition monitoring of cobots. Rubber bands of varying elasticity were used to simulate different degrees of force on one of the robot joints (specifically, the third joint). The physical model of the UR5 collaborative robot was used to obtain signals, such as position and current, reflecting the robot's operational behavior under diverse test conditions. These signals were compared with actual signals from the robot during operation. Abnormal joint conditions were effectively observed when calculating the residual values between the target and actual signals. The findings indicate that the residual of the current signal exhibits an increasing trend as the magnitude of abnormal forces increases across different trajectories, distinguishing normal and abnormal states.

In our previous work [27], anomalous conditions were artificially introduced to a cobot by attaching different weights on its third joint. Two unsupervised learning models, namely, PCA and an SAE, were developed to identify such anomalies using multivariate data obtained from a universal robot (UR5e). These models were employed to reconstruct the normal or baseline data. The evaluation of the PCA model involved examining the Hotelling T^2 and Q-statistic metrics based on the reconstruction error, while the Mahalanobis distance was utilized for the SAE model to identify abnormal conditions. The outcomes demonstrate that both methods are effective at sensitively detecting the investigated anomaly. However, PCA exhibits greater computational efficiency. Due to the potential of AEs, further research is needed to explore SAE and other variations of AEs, while also considering another group of anomalies.

An adaptive anomaly detection scheme was proposed in [21] by the authors to provide a robust detection system that works when there is a change in the data pattern. The scheme has two stages: the first is the concept drift (CD) detection stage, and the second stage is

anomaly detection (AD). The CD model was trained with multivariable data such as current, position, speed, and torque data from the robot, while the AD model was trained using torque data. During the online deployment of the scheme, the CD and AD models were retrained when a concept drift was detected; otherwise, AD was performed. A probabilistic encoder was used for the CD detector, while a CNN encoder structure was developed for the anomaly detector. The scheme was validated using normal and fault data from an Omron TM5-900 cobot. A major issue identified about the introduced concept was the computation requirement of the two models. The authors plan to combine the two ideas (CD and AD) in a single model in their future work. In addition, the fault in the robot motor or gear system was simulated by manipulating the torque data (10% increase).

Research Challenges and Proposed Solutions

A summary of existing challenges related to cobot anomaly detection includes (1) model performance degradation under a change in operating condition (i.e., speed or trajectory change) and environmental changes (e.g., external push), (2) limited coverage of fault scenarios during experiments targeted at fault diagnosis, (3) the computational complexity of anomaly detection models and very few works on the online deployment of detection schemes.

To solve the challenges highlighted above, two classes of algorithms are proposed in this study. The first class is for anomaly detection, while the second is for trajectory change detection. The sensitivity of three AE-based anomaly detection algorithms to two different anomalies is compared. Moreover, we introduce, analyze, and implement a flowchart to integrate the anomaly and trajectory change detection models into the real-time monitoring of cobots.

This work shows that slight variations (anomalies) in robot operating data can be detected by combining the Mahalanobis distance with few-layer autoencoder models. Another novel aspect of this study is the implementation of the trajectory adaptive condition monitoring framework, which improves the reliability of the anomaly detection models.

3. Methodology

The Section 3 explains the basics of AI-based anomaly detection and introduces three different algorithms for detecting anomalies. It also presents an algorithm for detecting changes in the robot's trajectory. Additionally, this section covers the test setup, the effects of the designed anomalies, and how the data were visualized and preprocessed.

3.1. AI-Based Anomaly Detection

The framework for anomaly detection is shown in Figure 3, which expressly depicts the training and testing phases of an AI-based detection model called an autoencoder (AE). This unsupervised learning AI method comprises three components, an encoder, latent space, and a decoder, in that order, as illustrated in Figure 4.

An AE is a type of neural network that learns to compress input data into a smaller representation (encoding) and then reconstructs the original input from this representation (decoding) through training. During training, the normal data alone are used, and they serve as both input and a target. Subsequently, the trained model undergoes testing using normal and anomalous data, with a higher reconstruction error in this scenario indicating abnormalities within the data [28,29].

Despite being unsupervised, the weights are updated iteratively using optimization algorithms like Adam optimization or stochastic gradient descent, minimizing the difference between the input and the reconstructed output through backpropagation [30]. After presenting the general architecture of an AE, the three specific types of AEs utilized in this study will be examined.

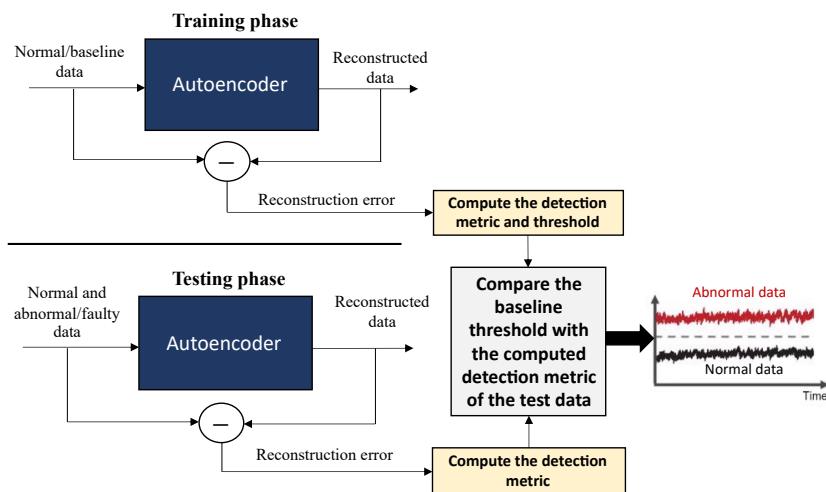


Figure 3. Anomaly detection framework.

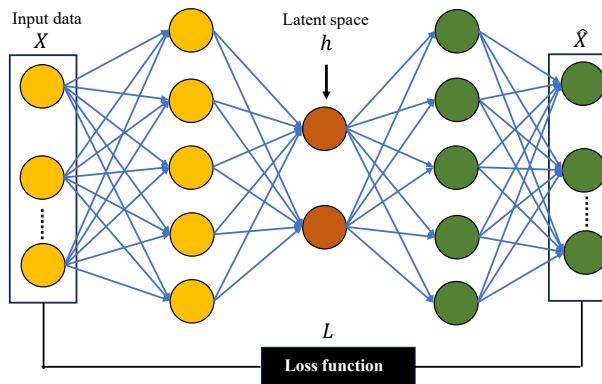


Figure 4. Autoencoder architecture.

3.1.1. MLP-Based Autoencoder

The MLP-AE is a type of AE architecture that utilizes multi-layer perceptrons (MLPs) or an FNN as its building blocks—that is, the encoder and decoder parts. An MLP consists of multiple layers of neurons where each neuron in one layer is connected to every neuron in the subsequent layer, forming a dense, fully connected network [31,32]. Mathematically, the operation within a single neuron is expressed as follows:

$$f_i = f_{act} \left(\sum_{i=1}^n w_i x_i + b \right), \quad (1)$$

where f_i is the neuron output upon activation, $f_{act}(\cdot)$ represents the activation function (common examples include sigmoid, tanh, ReLU, and softmax functions [33]), w_i is the weight associated with the i -th input, b is the bias, and n is for the number of inputs (x). MLP-AEs are flexible and can be adapted to various tasks and datasets by adjusting the number of layers, the number of neurons in each layer, and the activation functions used.

3.1.2. CNN-Based Autoencoder

This is a type of AE architecture that incorporates convolutional and max pooling or upsampling layers in its encoder and decoder structure. The convolution operation involves applying a filter, also known as a kernel, to an input image or time-series data to highlight patterns and features present in the input [29,31,34]. The convolution operation is expressed mathematically as follows:

$$f_{ij}^l = f_{act} \left(\sum_i^{i+m-1} k_j^l * x_i^{l-1} + b_j^l \right), \quad (2)$$

where f_{ij}^l and x_i^l are the output and input of the convolution, $*$ represents the convolution operator, i is the sliding index, which is determined by the input (x_i^l) size, k_j^l is the kernel or filter of the j -th neuron in layer l , m is the filter size, and b_j^l is the scalar bias of layer l .

A max pooling operation is a downsampling operation used in the encoder part and after the convolutional layers to decrease the spatial dimensions of the input data, while an upsampling operation is used in the decoder part to increase the spatial dimensions of feature maps. The CNN-AE architecture is particularly well suited for tasks involving spatial or sequential data such as images, time series, or sequences.

3.1.3. Sparse Autoencoder

This is a type of AE with the ability to enforce sparsity in the learned representations through an improvement of the standard loss function such as the mean squared error (MSE) [35]. Technically, overfitting is prevented by adding a regularization term to the loss function, and it also helps in a more efficient use of computational resources. The modified loss function has $L2$ sparsity regularization terms, hence the reason for the name SAE [36].

$$L = \underbrace{\frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K (x_{kn} - \hat{x}_{kn})^2}_{mse} + \underbrace{\gamma * \Omega_{weights}}_{L2 \text{ regularization}} + \underbrace{\mu * \Omega_{sparsity}}_{sparsity \text{ regularization}}, \quad (3)$$

where L is the improved loss function, γ is the coefficient of the $L2$ regularization, and μ is the coefficient of the sparsity regularization.

3.2. Anomaly Scores and Detection Metrics

Anomaly scores are numerical values used to quantify the degree of abnormality. There are various ways of computing anomaly scores, one of which is the maximum mean error of the AE model. Given the consideration of multivariate data, the Mahalanobis distance (MD) serves as a key metric for computing the distance between multivariable time-series data [37]. It is preferred over the Euclidean distance due to its ability to incorporate data correlations.

$$MD = \sqrt{(x - \mu)C^{-1}(x - \mu)^T}, \quad (4)$$

where MD is the Mahalanobis distance of data x to reference data y . μ and C represent the mean and covariance of the reference or baseline data.

The threshold of the model can be calculated through the 3-limits of the baseline MD result.

$$MD_\alpha = \mu_{MD} + 3\sigma_{MD}, \quad (5)$$

where μ_{MD} is the mean of MD , and σ_{MD} is the standard deviation of MD .

The confidence limit of the model is $MD \leq MD_\alpha$.

After computing the anomaly scores and threshold, there are other metrics to evaluate the performance of anomaly detection algorithms. These include accuracy, precision, recall, F1 score, false-positive rate, false-negative rate, receiver operating characteristic area under the curve, etc.

3.3. Trajectory Change Detection Models

Robot trajectory can change based on the task; however, this change leads to variations in sensor readings, motor responses, and overall system behavior. Hence, fault/anomaly detection and diagnosis models trained on previous trajectory patterns may struggle to adapt and accurately detect faults or anomalies when faced with new trajectories. The speed

at which the robot operates and the payload carried by the robot also affect the overall dynamics; however, the scope of this work is limited to drift caused by a change in trajectory. There are a few algorithms that can be used to detect a change in robot trajectories. These include dynamic time warping and kernel density estimation. In this work, dynamic time warping (DTW) is used because it is robust to time variation and non-linear temporal behavioral differences between two trajectories. DTW is a powerful detection algorithm which calculates the similarity between robot trajectories and identifies any significant deviations or changes [38]. The steps of the DTW algorithm are as follows:

1. Define two trajectories' data (Q_1 and Q_2), representing the robot's movements in the X, Y, Z coordinates over time:

$$Q_1 = \begin{bmatrix} q_{x,1} & q_{y,1} & q_{z,1} \\ q_{x,2} & q_{y,2} & q_{z,2} \\ \vdots & \ddots & \vdots \\ q_{x,n} & q_{y,n} & q_{z,n} \end{bmatrix}, \quad (6)$$

$$Q_2 = \begin{bmatrix} q_{x,1} & q_{y,1} & q_{z,1} \\ q_{x,2} & q_{y,2} & q_{z,2} \\ \vdots & \ddots & \vdots \\ q_{x,m} & q_{y,m} & q_{z,m} \end{bmatrix}, \quad (7)$$

where n and m are the number of data points in each trajectory.

2. Create a matrix CD of size $n \times m$, in which the elements in the i th row and the j th row correspond to the distance between the i th elements in Q_1 and the j th elements in Q_2 . This involves computing and storing the Euclidean distance between points X, Y, and Z in Q_1 and Q_2 within a distance matrix.
3. Create the cumulative distance matrix. First, the top-left corner of the matrix is initialized as $CD[1, 1]$ because the DTW algorithm needs to align the start of the two sequences:

$$CD[1, 1] = D[1, 1]. \quad (8)$$

Then, the other elements in the matrix are calculated by as follows:

$$CD[i, 1] = CD(i - 1, j) + D[i, 1], \quad (9)$$

$$CD[1, j] = CD(1, j - 1) + D[1, j], \quad (10)$$

$$CD[i, j] = D(i, j) + \min(CD[i - 1, j], CD[i, j - 1], CD[i - 1, j - 1]). \quad (11)$$

The final DTW distance is $CD[n, m]$, which is the lower-right element of the matrix CD . The normalized final distance, CD_{final} , is

$$CD_{final} = \frac{CD[n, m]}{(n + m)}, \quad (12)$$

The smaller the value of CD_{final} , the higher the similarity between Q_1 and Q_2 . A trajectory change happens if the threshold is $CD_{th} < CD_{final}$.

3.4. Test Setup

The industrial robot utilized for this study was a UR5e collaborative robot manufactured by Universal Robots [39]. To enable seamless communication and data exchange between the universal robot and external apps, a Real-Time Data Exchange (RTDE) framework was employed [40]. RTDE utilizes appropriate networking protocols and APIs such as an Ethernet connection to provide a reliable and high-bandwidth communication channel between the external app program (e.g., a Python script) and the UR5e robot controller, thus facilitating efficient data exchange, reducing latency, and enabling fast and accurate

control of the robot's movements. An overview of the setup for data exchange in the UR5e robot is depicted in Figure 5.

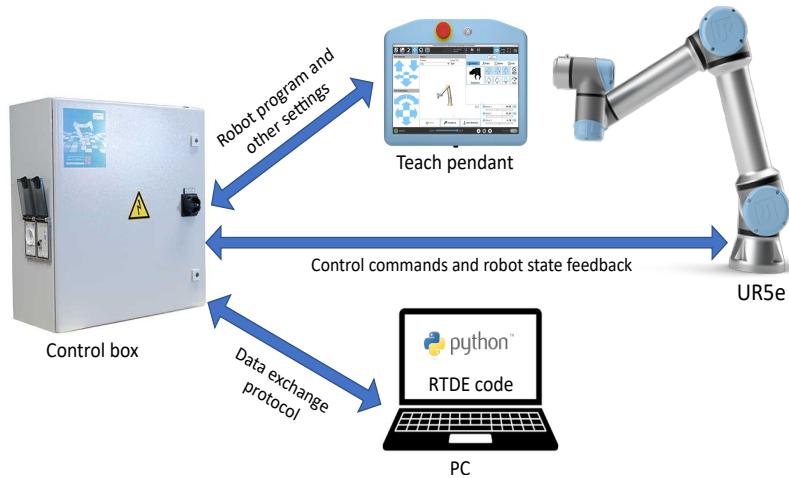


Figure 5. Overview of the data exchange setup for UR5e robot.

By leveraging the capabilities of RTDE, the data-gathering process was customized and adapted according to the requirements of our application. This allowed us to collect a wide range of variables, including joint temperatures, current, voltages, velocities, positions, moments, and TCP information, in real time. During the experiments conducted, the collection frequency was set to every 0.04 s within the RTDE program to ensure the data were acquired at a high resolution. Ideally, this frequency would correspond to a sampling rate of 25 Hz. However, due to the concurrent execution of other processes within the program, the actual data collection frequency was approximately 22 Hz. Despite this slight deviation from the desired frequency, the collected data provide valuable insights into the robot's performance and behavior. Two sets of experiments were carried out, and the same trajectory was followed during each experiment. The two groups of experiments were conducted as follows.

3.4.1. Anomaly 1 Experiments

Five tests were performed for the group 1 experiments, as summarized in Table 2. The first test of experimental set 1 was the baseline—in this case, no external weight was added to the shoulder arm of the robot—whereas, in the remaining four tests in this group, a 3D-printed part with holes (for attaching different numbers and sizes of bolts) was fitted to the shoulder arm of the robot as shown in Figure 6a. The weight of the 3D-printed part was 150, and this was increased by changing the number and size of bolts on the 3D-printed part. The weights used for the four cases were 150 g, 330 g, 450 g, and 750 g, respectively.

Table 2. Group 1 experiments for the first class of anomalies.

Experiment	Weight	Label
1	0 g	
2	150 g	
3	330 g	
4	450 g	abnormal
5	750 g	

3.4.2. Anomaly 2 Experiments

In the second group of experiments, summarized in Table 3, normal and abnormal operational data were collected when a vibrator was placed on the elbow joint of the robot. The vibrator was powered by a lithium battery and controlled using an ESP32

microcontroller, as illustrated in Figure 6b. The first test in experimental set 2 was the baseline, in which the vibrator was not turned on. Two more tests were performed in experimental set 2 where the vibrator was set to half and full speed, respectively (based on the PWM cycle in the control code).

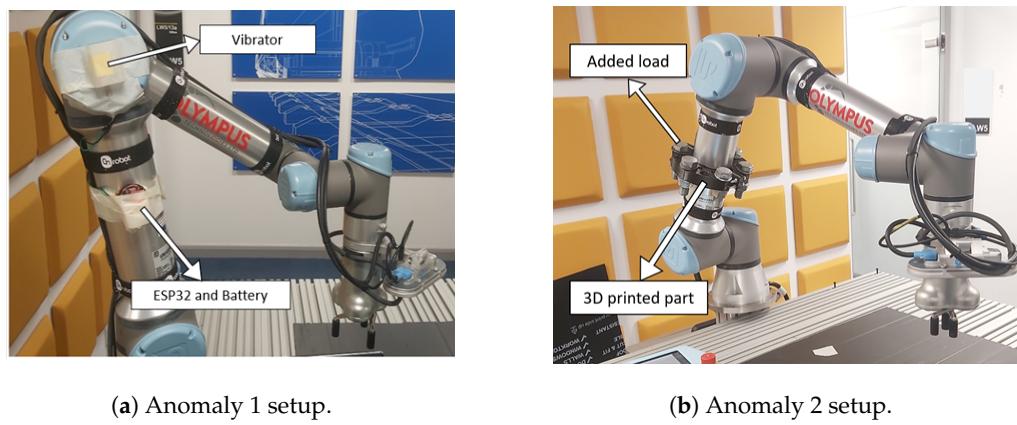


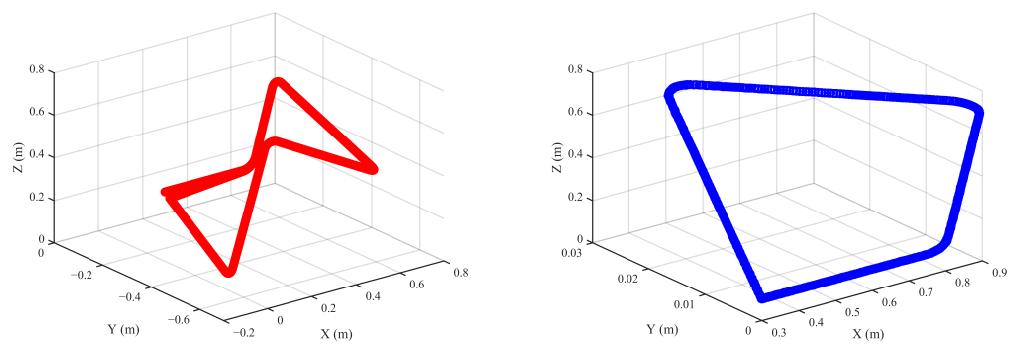
Figure 6. Universal robot (UR5e) with two different anomalies.

Table 3. Group 2 experiments for the second class of anomalies.

Experiment	Speed	Label
1	0 speed	normal
2	50% speed	abnormal
3	00% speed	

In the two groups/sets of experiments conducted, the robot moved through a path consisting of 6 points, as observable in the 3D trajectory plot in Figure 7a. As shown in Figure 7b, another trajectory named TCP translational motion 2 was designed using the teach pendant programming interface and used for the acquisition of the test dataset employed in validating the trajectory change detection model. The two trajectories were designed to be arbitrary rather than tailored to a specific task like the robot picking up and placing an object. However, a key consideration in designing the trajectory paths using the teach pendant was to ensure that all six joints were involved in the movement. In both trajectories (TCP translational motions 1 and 2), none of the joints remained static, allowing for comprehensive observation of the impact of the investigated anomalies across all the joints. The maximum speed and acceleration of the motion were 100 mm/s and 400 mm/s², respectively. In total, 93 variables were acquired from the robot. Furthermore, the total number of data points in each experiment was 4000, and each experiment lasted for about 185–186 s/≈ 3 min. Out of the 93 variables, 30 were used in training the models studied in this work. The signals/variables eliminated were either the target signals or they were variables with very small residuals when the plots of each signal based on different anomalies were produced. Additionally, the joint temperatures were not considered in the anomaly detection model development because we noticed that these variables increased based on the time and the order of the experiment.

Among the selected variables, four of them, namely, motor input current, joint angular position, TCP position, and TCP force/torque, are plotted in Figures 8 and 9 to show the effect of the normal and abnormal conditions on those robot variables. By visual inspection of both figures, no significant difference can be seen between the data variables obtained under normal and abnormal conditions. The reason for this can be linked to the closed-loop feedback control system of the robot, which quickly corrected deviations. Notwithstanding, the signals will be statistically analyzed in Section 3.6 to show the variations caused to each signal by the two sets of anomalies.



(a) TCP translational motion 1.

(b) TCP translational motion 2.

Figure 7. Combined view of TCP translational motions.

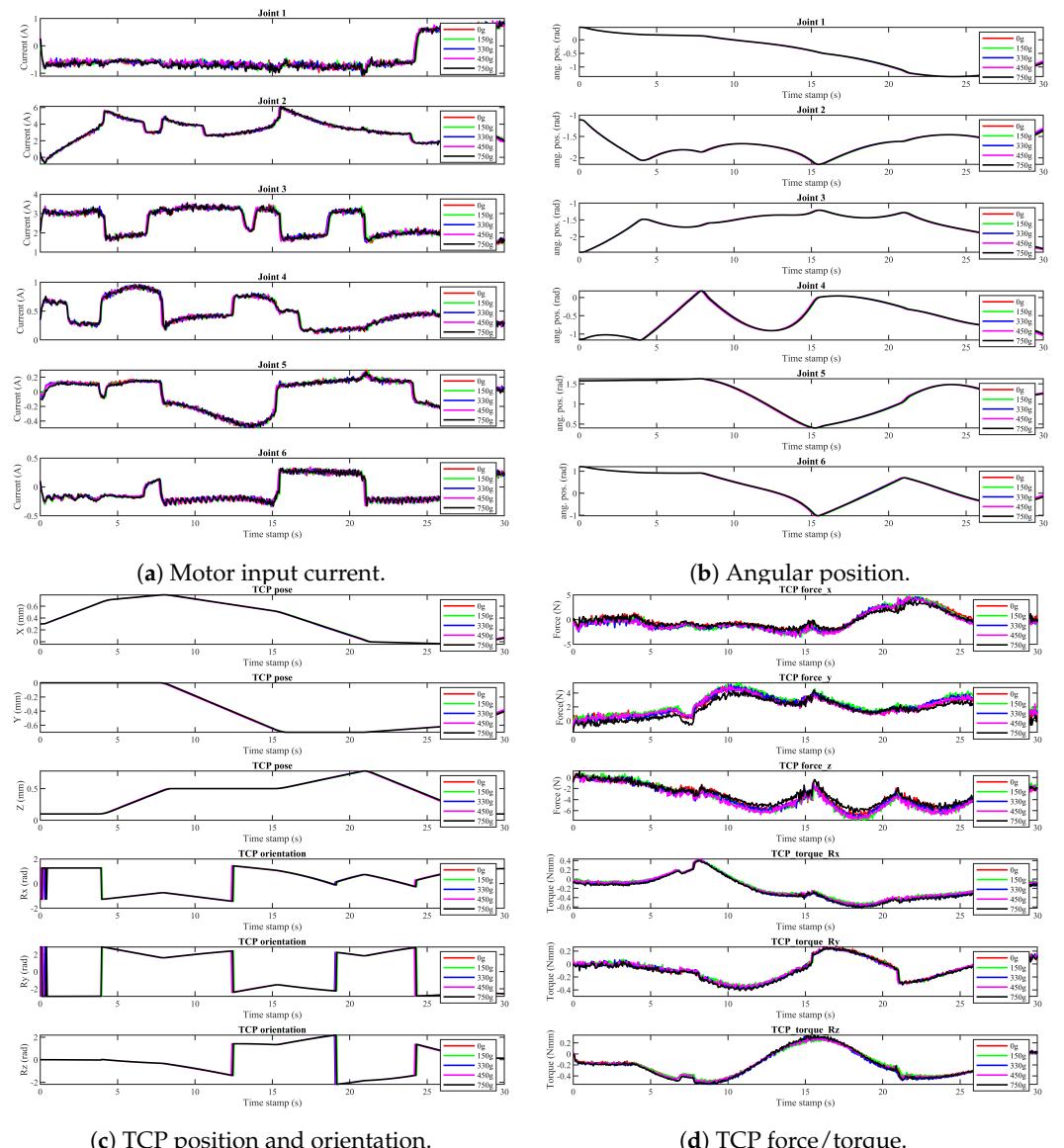


Figure 8. Plots of some cobot variables for the normal condition and first anomalies.

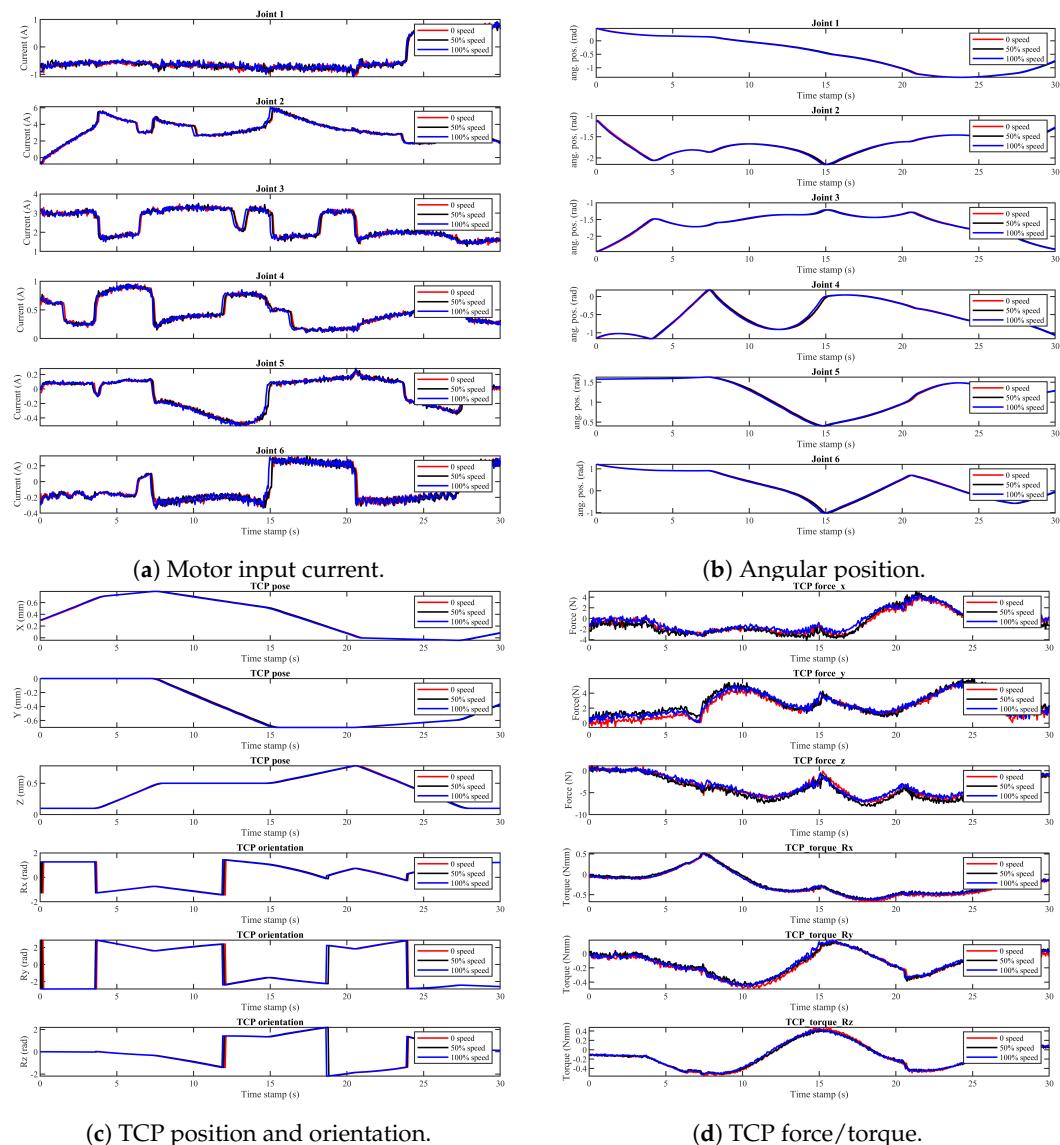


Figure 9. Plots of some cobot variables for the normal condition and second anomalies.

3.5. Theoretical Analysis of the Impact of Designed Anomalies

The theoretical explanation of the effects induced by the two anomalies is presented in this section. Generally, the motion of any industrial robot/manipulator solely depends on the kinematic model, dynamic model, and control algorithm, and these three parts are interconnected. As a result, we can calculate some elements in the dynamic equation from the kinematic knowledge, and the robot can be controlled to the desired trajectory using the dynamic information. The dynamic model of a robot is usually expressed as

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau, \quad (13)$$

where $M(q) \in \mathbb{R}^{n \times n}$ is the inertia matrix; $C(q, \dot{q}) \in \mathbb{R}^{n \times n}$ is the matrix of Coriolis and Centrifugal terms; $G(q) \in \mathbb{R}^{n \times 1}$ denotes the gravitational force vector; q, \dot{q} , and $\ddot{q} \in \mathbb{R}^{n \times n}$ are the joint's position, velocity, and acceleration, respectively; and τ is the vector of the total joint torques (which is dependent on the motor torque, friction, and other external torques).

When friction(τ_f) is accounted for, the equation becomes

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau - \tau_f. \quad (14)$$

Anomaly 1 involved applying varying weights to the link near joint 3. This increased the torque (τ) and input current required to move the joint. Joint faults, often caused by component degradation and increased friction, are a significant concern. The added weight increases the inertia (M) of the link; thus, more force is required to accelerate or decelerate it. Moreover, as the torque increases, the frictional torque is increased (τ_f), additional strain is put on the motor and gearbox, and the robot's accuracy and precision in reaching desired positions are affected based on the control algorithm used.

The second group of anomalies involved introducing a vibrator with two distinct vibrating speeds to joint 3. In practice, the simulated anomalies created with the vibrator could represent mechanical faults in the joint's motor or gears, which are common fault sources in robotics. The vibrator induces vibrations in the joint, resulting in displacement from its intended position, which affects the accuracy of the joint's positioning. Consequently, the joint's torque and current are changed (increased or decreased) to compensate for the anomalous condition. If the vibration intensity is significant, it may induce oscillations or resonance effects in the mechanical parts of the joint. Additionally, the mechanical stress induced by the vibration could accelerate wear on components such as bearings and gears.

3.6. Data Visualization and Preprocessing

For easy visualization of the variation in our dataset, the difference between the mean of each abnormal signal and that of the corresponding normal signal was computed with and without normalization for each group of experiments. The mathematical equation for the normalization of each of the robot variables ($y_{k,\text{norm}}$) is

$$y_{k,\text{norm}} = \frac{y_k - \text{mean}(y_k)}{\text{std}(y_k)}, \quad (15)$$

where k is an indicator of the data variable. Figure 10a shows the mean error plot of each signal for the different weights (i.e., weights 1–4 in the test cases) with respect to the baseline data, while Figure 10b shows the mean error plots after the signals were normalized. In the first plot, the TCP forces show high magnitude and variation compared to the other variables. A similar figure is shown in Figure 11a,b for the second anomaly experimental dataset. By examining the figure, it is clear that the mean errors of the TCP force variables are more dominant, and more variables appeared after normalization in Figure 11b. Furthermore, the data were also smoothed to remove noise from the measured signals. The smoothening function takes the median of data variables over a w -length sliding window through the following equation:

$$y_s[i] = \frac{1}{s} \sum_{i-\frac{s}{2}}^{i+\frac{s}{2}} y[i+j], \quad (16)$$

where $y_s[i]$ is the smooth value at point i , $y[i+j]$ represents the original data points within the window centered at position i , and s is the span of the moving average window. The sliding window length used in this work was 50 based on the result of comparing different window lengths in [27].

In the two figures (Figures 10 and 11), the variables on the x-axis are *jointi pos*, *jointi vel*, *jointi cur*, *TCP pos*, and *TCP force*. *Jointi pos*, *jointi vel*, and *jointi cur* represent the i-th joint position, velocity, and current, respectively, while *TCP pos* and *TCP force* represent the Tool Center Point position and force/torque in x , y , and z positions and Rx , Ry , and Rz orientation.

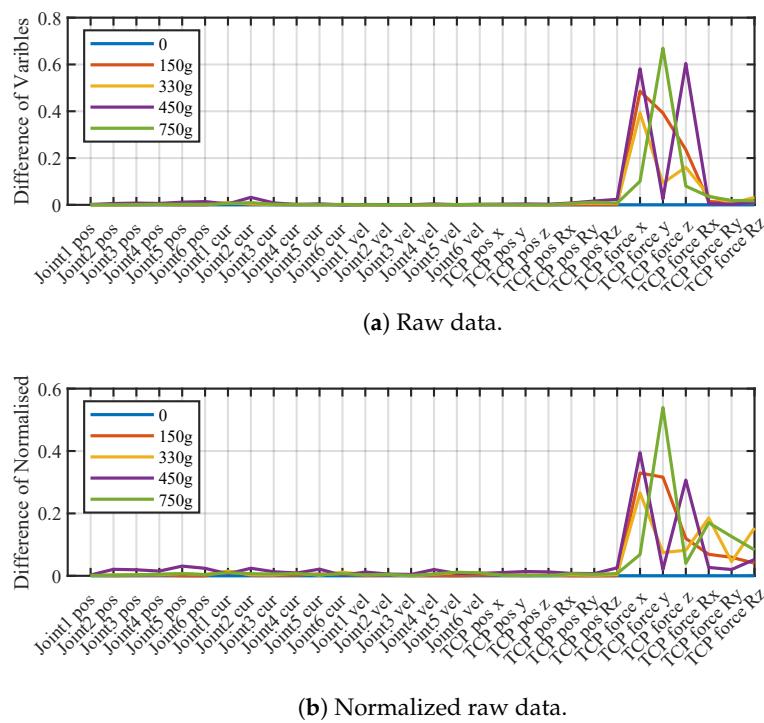


Figure 10. Anomaly 1 experimental dataset visualization.

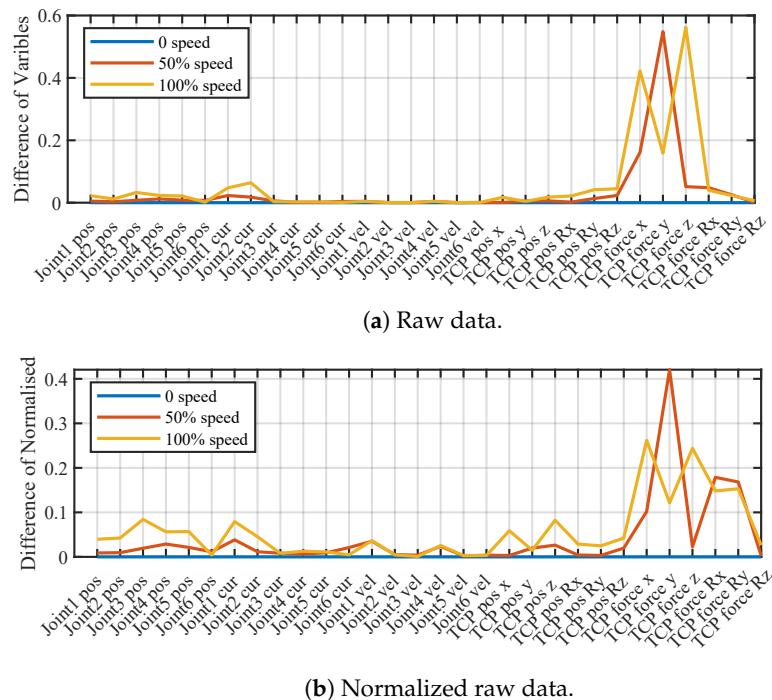


Figure 11. Anomaly 2 experimental dataset visualization.

4. Model Training and Results

In this section, the training process and results of the anomaly and trajectory change detection models are discussed.

4.1. Anomaly Detection Models

The architecture of the three AEs discussed in Section 3.1 was created, and the models were trained using the normal data from the two groups of experiments that were previously normalized and smoothed. This means six models were trained simultaneously:

three different AE models for group 1 normal data and another three unique AE models for group 2. The MLP-AE architecture consisted of four and three layers in the encoder and decoder parts, respectively, as shown in Figure 12a, while the CNN-AE architecture was made up of four layers in both the encoder and decoder parts, as illustrated Figure 12b. The SAE architecture was only composed of one layer in both the encoder and the decoder parts, as shown in Figure 12c. The training epochs set for the MLP-AE, CNN-AE, and SAE were 50, 10, and 50, respectively, when the Adam optimizer was used. The time it took to train each model is summarized in Table 4. It was observed that the MLP-AE took the least amount of time, followed by the SAE, while the CNN-AE took a relatively longer time.

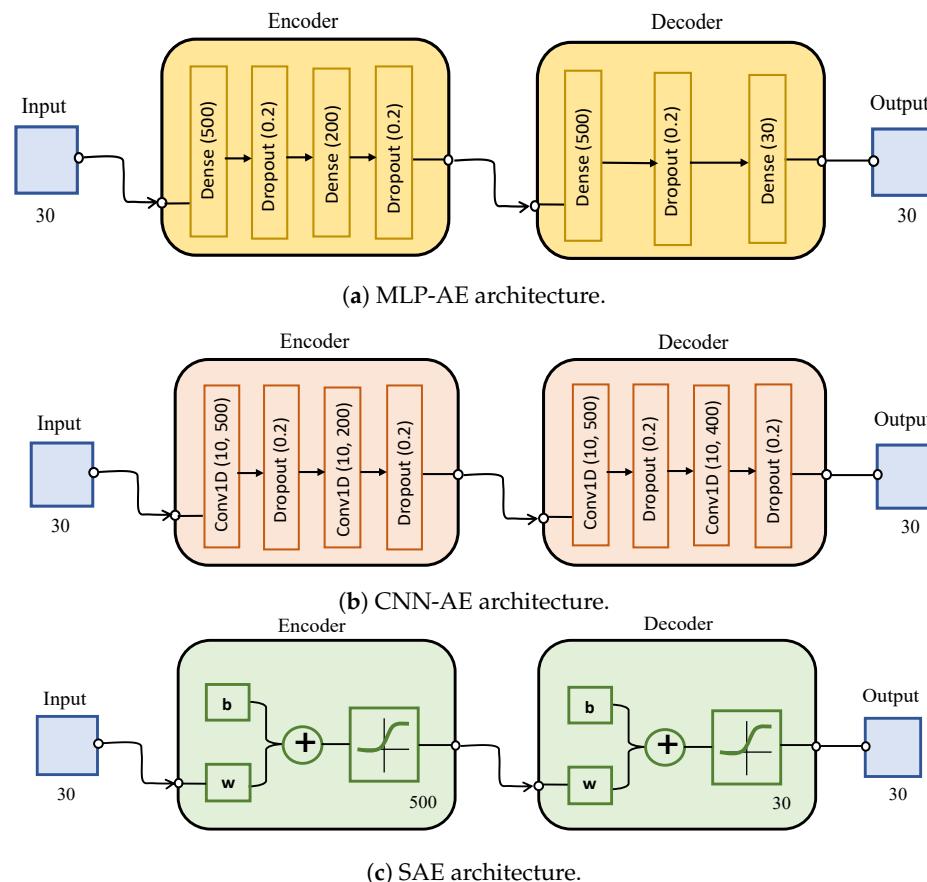


Figure 12. Autoencoders' architecture.

Table 4. Training time of the anomaly detection models.

Model	Training Time (s)	
	Group 1 Experiment	Group 2 Experiment
MLP-AE	6.10	6.23
CNN-AE	60.78	60.59
SAE	11.64	12.85

In Figures 13 and 14, the *MD* between the baseline residual of the SAE model and its mean is shown. The detection threshold was calculated using the three-sigma value of *MD*, as depicted by the broken red lines in both plots. In the first subplot of Figure 13, the *MD* values (the solid black line) are below the threshold except for the starting and end points, where non-zero *MD* values can be observed. In subplots 2 to 5 of Figure 13, the distance between the residual of the different weights (150 g, 330 g, 450 g, and 750 g) and the mean of the baseline residual is plotted, and it can be seen that the *MD* values are mostly above the threshold except for a few points. This means that the SAE model could detect the

anomaly caused by the weights (weights 1–4). Similarly, the plot obtained for anomaly 2 (Figure 14) has the same interpretation as that of anomaly 1 (Figure 13). It can be observed that the MD values (solid blue lines) are mostly below the threshold for the baseline case, while the MD values are mostly above the threshold for the anomalous conditions where the vibrator speed was set to 50 and 100%, respectively.

Furthermore, the accuracy and F1 score of each experimental set with respect to the three models (MLP-AE, CNN-AE, and SAE) were evaluated for the anomalous cases, and the results are shown in Tables 5 and 6. Accuracy measures how well the model correctly predicts both normal and anomalous instances out of the total instances, while the F1 score reflects the balance between precision and recall. Across all experiments, the three models achieved an accuracy greater than 93% for the anomalous cases, with the lowest F1 score exceeding 96%. Moreover, in both tables, it can be observed that the CNN-AE had the overall best accuracy. However, SAE was used for the local condition monitoring implementation in Section 5 because it had the simplest architecture. Since anomaly detection is similar to binary classification, with only two data classes, it tends to result in higher accuracy. To prevent overfitting, we used smaller-layer models, applied early stopping, and included a regularization term in the SAE.

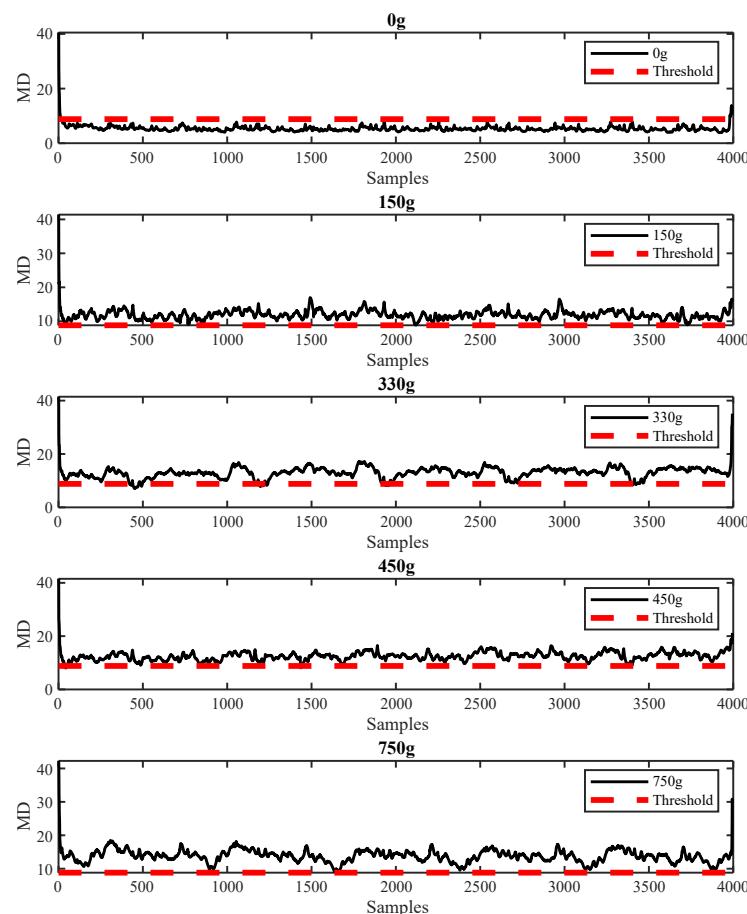


Figure 13. MD plot of the baseline and different weights of the arm load ($w = 50$).

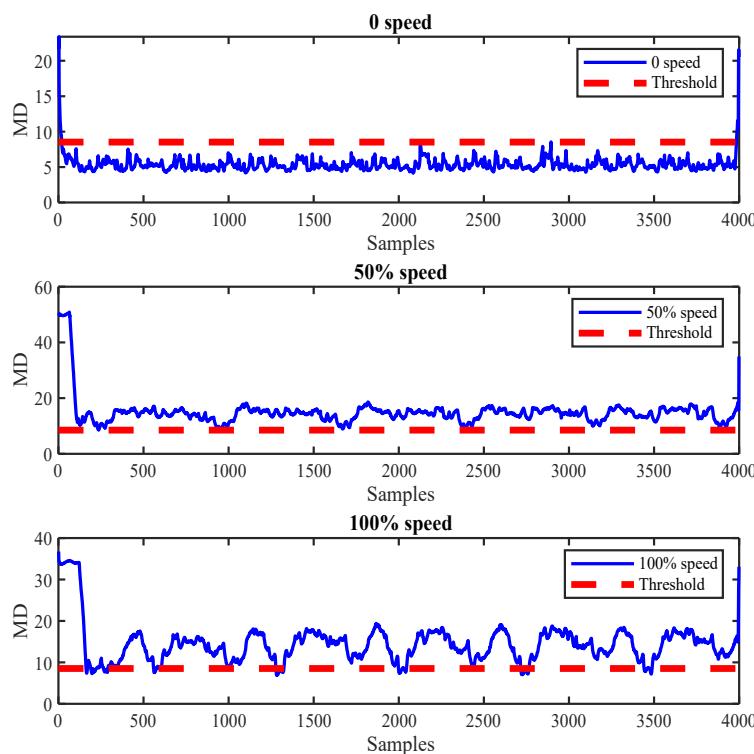


Figure 14. MD plot of the baseline and different speeds of the vibrator ($w = 50$).

Table 5. Performance metrics of the anomaly detection models with group 1 anomalies.

MLP-AE		CNN-AE		SAE	
S/No.	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy
150 g	100%	100%	100%	100%	96.78%
330 g	96.13%	98.02%	100%	100%	97.48%
450 g	99.45%	99.72%	100%	100%	99.60%
750 g	100%	100%	100%	100%	99.80%

Table 6. Performance metrics of the anomaly detection models with group 2 anomalies.

MLP-AE		CNN-AE		SAE	
S/No.	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy
50% speed	99.90%	99.95%	100%	100%	99.90%
100% speed	95.82%	97.87%	98.50%	99.24%	93.50%

4.2. Trajectory Change Detection Model

The cobot TCP position in the X, Y, and Z directions was used in the trajectory change detection model development. The DTW algorithm was employed in estimating the cumulative distance between trajectory 1 data for the baseline case and the weight 4 anomalous condition. The distance between the two trajectory data was found to be 8.5609×10^{-4} . Based on the distance, a threshold was then set as 0.005 (based on the observed range of variation) to detect a change in trajectory. Afterward, we used the same algorithm to determine the cumulative distance between TCP translational motion 1 (i.e., the trajectory displayed in Figure 7a) and the new motion shown in Figure 7b. We then compared the cumulative distance to the threshold set, and the result obtained shows that the distance between the test trajectory and the previous one was higher than the set threshold, thus indicating that a trajectory change was detected.

5. Online Monitoring

This section presents a framework for the online condition monitoring of cobots, and the framework was also implemented.

5.1. Monitoring Framework

One of the key benefits of using cobots, as opposed to traditional industrial robots, lies in their flexibility. This flexibility allows cobots to handle multiple tasks simultaneously, both in small- and large-scale industries. Hence, we have designed a robust and trajectory change adaptive framework to accommodate this versatility, as illustrated in Figure 15.

The framework assumptions are the following:

1. The data read by the cobot are for one cycle of a particular collaborative task.
2. The robot has primary (trajectory 1) and secondary collaborative tasks (i.e., other trajectories, 2, 3, 4, etc.).
3. It is assumed that the detection model used for each trajectory is pre-trained, and a new model is selected from the saved secondary tasks' models when a trajectory change occurs.
4. The online monitoring framework will alert the maintenance operator either when an anomaly is detected or when there is a case of a completely new trajectory without a corresponding pre-trained model.

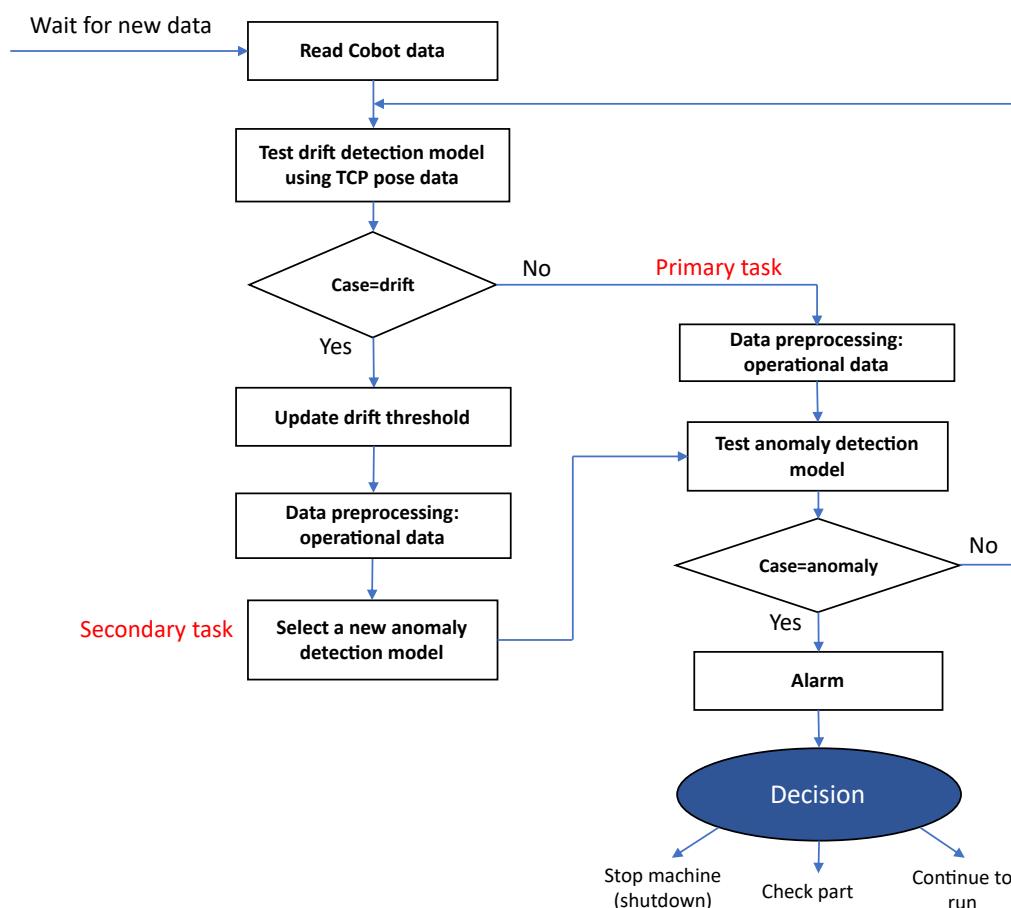


Figure 15. A flowchart for robust and drift-resilient cobot online condition monitoring framework.

5.2. Online Monitoring Implementation

In this section, we will discuss the implementation of the online monitoring framework. The implementation was carried out locally with a personal computer, and the MATLAB app designer was used to design a graphical user interface (GUI) for the online condition

monitoring of the UR5e cobot. The online monitoring process includes real-time data acquisition, preprocessing, and the detection of anomalies as the robot performed different operations. The code begins by specifying the folder where the robot data are stored. Then, the data variables of interest are read, and some program-related variables are also initialized, including a variable where the list of processed data files can be tracked and another variable for setting flags for the loop control. The user interface shown in Figure 16 is updated with messages indicating that the robot monitoring has begun and the type of task or trajectory by the robot is being checked (see this update in Figure 17a). Afterward, a while loop runs continuously to monitor the robot. This loop is designed to keep running while the program is active. Within the loop, the code performs all or some of the following steps for each data file:

- Checks if the file has been processed before.
- If it is a new data file, it reads the data from the Excel file and stores them.
- Smooths and normalizes the data.
- Calculates the DTW distance between the new trajectory data and a baseline trajectory.
- Determines if a trajectory change is detected and updates the UI.
- Employs the SAE for anomaly detection and updates the UI.

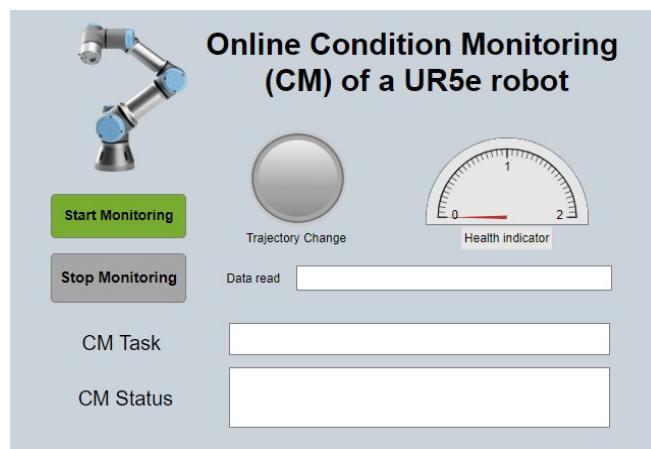


Figure 16. Online monitoring GUI.

Depending on whether a trajectory change is detected, the code updates the user interface and performs anomaly detection:

- If no change is detected, the accuracy of the detection model is checked, and the UI is updated accordingly.
- If a change is detected, the anomaly detection model is used to check the accuracy of a different trajectory, and the UI is also updated.

In Figure 17b, an example case of an anomaly and no trajectory change is displayed in the GUI. The code keeps track of processed files by adding them to the list of processed files. Additionally, a timer is used in the code to determine if new data have been detected within a specified timeout period. If no new data are detected within the timeout, the loop stops, and a message is displayed in the user interface (see Figure 17c). As shown in Figure 17d, the program can also stop when the 'Stop Monitoring' push button is pressed. In summary, the code is designed to continuously monitor data from an industrial robot, compare its trajectory with a baseline, and detect anomalies in the robot's behavior.



Figure 17. Online monitoring GUI for different test scenarios.

6. Conclusions

This research presents the condition monitoring of cobots using their multivariable operational data. It was found that acquiring fault data for developing fault diagnosis and prognosis models which can capture various faults and estimate the remaining useful life (RUL) of robots and their components is costly. As a result, fault/anomaly detection is a more practical and cost-effective approach. The literature review also revealed that the performance of anomaly detection models is affected by changes in trajectory. Therefore, both anomaly detection and trajectory change detection models were explored in this study.

The performance of three AEs (MLP-AE, CNN-AE, and SAE) was compared in terms of their anomaly detection capabilities. Additionally, DTW was employed for trajectory change detection. The results showed that the AEs were highly effective, achieving an overall accuracy of over 93% and an F1 score exceeding 96% for different anomalous conditions. The DTW algorithm also demonstrated satisfactory performance in distinguishing between two different trajectories.

Furthermore, a framework was presented for the online deployment of two digital models. These models included a trajectory change model and an anomaly detection model, which were trained offline using data collected during the cobot's primary task (trajectory 1). The assumptions crucial for the framework's operation were examined. Finally, the online monitoring framework was implemented locally and displayed the operation through a user interface. The algorithms and framework have the potential to be used for cobots and other industrial robots performing predefined manufacturing tasks in an industrial setting. In our future work, we aim to enhance the online monitoring algorithm by incorporating a model that is trained online once a trajectory change is detected, without the need to use pre-trained anomaly detection models. This will foster the online training and update of the trajectory change and anomaly detection algorithms to create a more robust condition monitoring framework. Additionally, the anomaly detection models developed in this study are for a particular operating condition (i.e., defined trajectory, tool speed, and acceleration). The model cannot be generalized for other trajectories. Hence, a residual-based model leveraging the robot mathematical model will be explored as part of our future research.

Author Contributions: S.A.: methodology, experiments, data analysis, coding, visualization, writing—original draft, writing—review and editing. F.G.: methodology, experiments, data analysis, coding, validation, visualization, writing—review and editing, resources, supervision. H.L.: methodology, experiments, visualization, writing—review and editing. H.F.: methodology, experiments, writing—review and editing, resources. A.B.: supervision, resources. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data will be made available on request.

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

References

1. Aliev, K.; Antonelli, D. Proposal of a Monitoring System for Collaborative Robots to Predict Outages and to Assess Reliability Factors Exploiting Machine Learning. *Appl. Sci.* **2021**, *11*, 1621. [[CrossRef](#)]
2. Knudsen, M.; Kaivo-oja, J. Collaborative Robots: Frontiers of Current Literature. *J. Intell. Syst. Theory Appl.* **2020**, *3*, 13–20. [[CrossRef](#)]
3. Lee, J.; Ni, J.; Singh, J.; Jiang, B.; Azamfar, M.; Feng, J. Intelligent Maintenance Systems and Predictive Manufacturing. *J. Manuf. Sci. Eng.* **2020**, *142*, 110805. [[CrossRef](#)]
4. Achouch, M.; Dimitrova, M.; Ziane, K.; Sattarpanah Karganroudi, S.; Dhouib, R.; Ibrahim, H.; Adda, M. On Predictive Maintenance in Industry 4.0: Overview, Models, and Challenges. *Appl. Sci.* **2022**, *12*, 8081. [[CrossRef](#)]
5. Montero Jiménez, J.J.; Vingerhoeds, R.; Grabot, B.; Schwartz, S. An ontology model for maintenance strategy selection and assessment. *J. Intell. Manuf.* **2023**, *34*, 1369–1387. [[CrossRef](#)]
6. Biggio, L.; Kastanis, I. Prognostics and Health Management of Industrial Assets: Current Progress and Road Ahead. *Front. Artif. Intell.* **2020**, *3*, 578613. [[CrossRef](#)]
7. Kumar, P.; Khalid, S.; Kim, H. Prognostics and Health Management of Rotating Machinery of Industrial Robot with Deep Learning Applications—A Review. *Mathematics* **2023**, *11*, 3008. [[CrossRef](#)]
8. Raouf, I.; Kumar, P.; Lee, H.; Kim, H.S. Transfer Learning-Based Intelligent Fault Detection Approach for the Industrial Robotic System. *Mathematics* **2023**, *11*, 945. [[CrossRef](#)]
9. Ucar, A.; Karakose, M.; Kirimça, N. Artificial Intelligence for Predictive Maintenance Applications: Key Components, Trustworthiness, and Future Trends. *Appl. Sci.* **2024**, *14*, 898. [[CrossRef](#)]
10. Sabry, A.H.; Ungku Amirulddin, U.A.B. A review on fault detection and diagnosis of industrial robots and multi-axis machines. *Results Eng.* **2024**, *23*, 102397. [[CrossRef](#)]
11. Galan-Uribe, E.; Amezquita-Sanchez, J.P.; Morales-Velazquez, L. Supervised Machine-Learning Methodology for Industrial Robot Positional Health Using Artificial Neural Networks, Discrete Wavelet Transform, and Nonlinear Indicators. *Sensors* **2023**, *23*, 3213. [[CrossRef](#)] [[PubMed](#)]
12. Han, H.; Lin, Y.; Gu, L.; Xu, Y.; Gu, F. Vibration Analysis Based Condition Monitoring for Industrial Robots. In *Proceedings of the IncoME-V & CEPE Net-2020*; Zhen, D., Wang, D., Wang, T., Wang, H., Huang, B., Sinha, J.K., Ball, A.D., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 186–195. [[CrossRef](#)]
13. Yun, H.; Kim, H.; Jeong, Y.H.; Jun, M.B.G. Autoencoder-based anomaly detection of industrial robot arm using stethoscope based internal sound sensor. *J. Intell. Manuf.* **2023**, *34*, 1427–1444. [[CrossRef](#)]
14. Rohan, A.; Raouf, I.; Kim, H.S. Rotate Vector (RV) Reducer Fault Detection and Diagnosis System: Towards Component Level Prognostics and Health Management (PHM). *Sensors* **2020**, *20*, 6845. [[CrossRef](#)]
15. Raouf, I.; Lee, H.; Kim, H.S. Mechanical fault detection based on machine learning for robotic RV reducer using electrical current signature analysis: A data-driven approach. *J. Comput. Des. Eng.* **2022**, *9*, 417–433. [[CrossRef](#)]
16. Lee, H.; Raouf, I.; Song, J.; Kim, H.S.; Lee, S. Prognostics and Health Management of the Robotic Servo-Motor under Variable Operating Conditions. *Mathematics* **2023**, *11*, 398. [[CrossRef](#)]
17. Graabæk, S.G.; Ancker, E.V.; Christensen, A.L.; Fugl, A.R. An Experimental Comparison of Anomaly Detection Methods for Collaborative Robot Manipulators. *IEEE Access* **2023**, *11*, 65834–65848. [[CrossRef](#)]
18. Brockmann, J.T.; Rudolph, M.; Rosenhahn, B.; Wandt, B. The voraus-AD Dataset for Anomaly Detection in Robot Applications. *IEEE Trans. Robot.* **2023**, *40*, 438–451. [[CrossRef](#)]
19. Majid, M.A.A.; Fudzin, F. Study on Robots Failures in Automotive Painting Line. *ARPJ. Eng. Appl. Sci.* **2017**, *12*, 62–67.
20. Polenghi, A.; Cattaneo, L.; Macchi, M. A framework for fault detection and diagnostics of articulated collaborative robots based on hybrid series modelling of Artificial Intelligence algorithms. *J. Intell. Manuf.* **2023**, *35*, 1929–1947. [[CrossRef](#)]
21. Kermenov, R.; Nabissi, G.; Longhi, S.; Bonci, A. Anomaly Detection and Concept Drift Adaptation for Dynamic Systems: A General Method with Practical Implementation Using an Industrial Collaborative Robot. *Sensors* **2023**, *23*, 3260. [[CrossRef](#)]
22. Qiu, S.; Cui, X.; Ping, Z.; Shan, N.; Li, Z.; Bao, X.; Xu, X. Deep Learning Techniques in Intelligent Fault Diagnosis and Prognosis for Industrial Systems: A Review. *Sensors* **2023**, *23*, 1305. [[CrossRef](#)]
23. Zhou, X.; Zhou, H.; He, Y.; Huang, S.; Zhu, Z.; Chen, J. Harmonic reducer in-situ fault diagnosis for industrial robots based on deep learning. *Sci. China Technol. Sci.* **2022**, *65*, 2116–2126. [[CrossRef](#)]
24. Samuel, A.; Eric, K.; Hassna, L.; Hamidreza, F.; Fengshou, G.; Andrew, B. A Hybrid Digital Twin Scheme for the Condition Monitoring of Industrial Collaborative Robots. *Procedia Comput. Sci.* **2024**, *232*, 1099–1108. [[CrossRef](#)]
25. Raviola, A.; Guida, R.; Bertolino, A.C.; De Martin, A.; Mauro, S.; Sorli, M. A Comprehensive Multibody Model of a Collaborative Robot to Support Model-Based Health Management. *Robotics* **2023**, *12*, 71. [[CrossRef](#)]

26. Han, H.; Huang, C.; Song, Y.; Li, D.; Hamidreza, F.; Fengshou, G.; Andrew, B. Residual-based fault detection of abnormal joint running state of industrial collaborative robot. In *Proceedings of the UNIfied Conference of DAMAS, IncoME and TEPEN Conferences (UNIfied 2023)*; Springer: Cham, Switzerland, 2024; Volume 1. [[CrossRef](#)]
27. Samuel, A.; Xiaoxia, L.; Hassna, L.; Hamidreza, F.; Fengshou, G.; Andrew, B. Sensitivity of PCA and Autoencoder-Based Anomaly Detection for Industrial Collaborative Robots. In *Proceedings of the UNIfied Conference of DAMAS, IncoME and TEPEN Conferences (UNIfied 2023)*; Springer: Cham, Switzerland, 2024; Volume 1. [[CrossRef](#)]
28. Tziolas, T.; Papageorgiou, K.; Theodosiou, T.; Papageorgiou, E.; Mastos, T.; Papadopoulos, A. Autoencoders for Anomaly Detection in an Industrial Multivariate Time Series Dataset. *Eng. Proc.* **2022**, *18*, 23. [[CrossRef](#)]
29. Chen, Z.; Yeo, C.K.; Lee, B.S.; Lau, C.T. Autoencoder-based network anomaly detection. In Proceedings of the 2018 Wireless Telecommunications Symposium (WTS), Phoenix, AZ, USA, 17–20 April 2018; pp. 1–5. [[CrossRef](#)]
30. Soydaner, D. A Comparison of Optimization Algorithms for Deep Learning. *Int. J. Pattern Recognit. Artif. Intell.* **2020**, *34*, 2052013. [[CrossRef](#)]
31. Ayankoso, S.; Olejnik, P. Time-Series Machine Learning Techniques for Modeling and Identification of Mechatronic Systems with Friction: A Review and Real Application. *Electronics* **2023**, *12*, 3669. [[CrossRef](#)]
32. Popescu, M.C.; Balas, V.E.; Perescu-Popescu, L.; Mastorakis, N. Multilayer Perceptron and Neural Networks. *World Sci. Eng. Acad. Soc. (WSEAS) Trans. Circuits Syst.* **2009**, *8*, 579–588.
33. Sharma, S.; Sharma, S.; Scholar, U.G.; Athaiya, A. Activation Functions in Neural Networks. *Int. J. Eng. Appl. Sci. Technol.* **2020**, *4*, 7. [[CrossRef](#)]
34. Indolia, S.; Goswami, A.; Mishra, S.; Asopa, P. Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. *Procedia Comput. Sci.* **2018**, *132*, 679–688. [[CrossRef](#)]
35. Adcock, B.; Dexter, N. The gap between theory and practice in function approximation with deep neural networks. *SIAM J. Math. Data Sci.* **2021**, *3*, 624–655 [[CrossRef](#)]
36. Train an Autoencoder—MATLAB. Available online: <https://uk.mathworks.com/help/deeplearning/ref/trainautoencoder.html> (accessed on 20 April 2023).
37. Leys, C.; Klein, O.; Dominicy, Y.; Ley, C. Detecting multivariate outliers: Use a robust variant of the Mahalanobis distance. *J. Exp. Soc. Psychol.* **2018**, *74*, 150–156. [[CrossRef](#)]
38. Kloska, M.; Grmanova, G.; Rozinajova, V. Expert enhanced dynamic time warping based anomaly detection. *Expert Syst. Appl.* **2023**, *225*, 120030. [[CrossRef](#)]
39. Universal Robot UR5 Technical Specification. 2016. Available online: <https://www.rarukautomation.com/wp-content/uploads/2022/11/universal-robot-ur5-technical-spec.pdf> (accessed on 4 May 2023).
40. Real-Time Data Exchange (RTDE) Guide. 2024. Available online: <https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/> (accessed on 24 April 2023).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.