*electronics*

*Article*

# PHIR: A Platform Solution of Data-Driven Health Monitoring for Industrial Robots

**Fei Jiang [1], Chengyun Hu [1], Chongwei Liu [2], Rui Wang [3,*], Jianyong Zhu [4], Shiru Chen [5] and Juan Zhang [6]**

[1] Waytous, Beijing 100080, China; fei.jiang@waytous.com (F.J.); chengyun.hu@waytous.com (C.H.)
[2] JD Logistics, Beijing 100176, China; liuchongwei1@jd.com
[3] State Key Laboratory of Software Development Environment (SKLSDE), Beihang University, Beijing 100191, China
[4] Department of Computer, North China Electric Power University, Beijing 102206, China; zhujy@ncepu.edu.cn
[5] Inspur Yunzhou Industrial Internet Co., Ltd., Beijing 100191, China; chenshiru@inspur.com
[6] Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, UK; juan.zhang@northumbria.ac.uk
* Correspondence: ruiking@buaa.edu.cn

**Abstract:** The large-scale application of industrial robots has created a demand for more intelligent and efficient health monitoring, which is more efficiently met by data-driven methods due to the surge in data and the advancement of computing technology. However, applying deep learning methods to industrial robots presents critical challenges such as data collection, application packaging, and the need for customized algorithms. To overcome these difficulties, this paper introduces a Platform of data-driven Health monitoring for IRs (PHIR) that provides a universal framework for manufacturers to utilize deep-learning-based approaches with minimal coding. Real-time data from multiple IRs and sensors is collected through a cloud-edge system and undergoes unified pre-processing to facilitate model training with a large volume of data. To enable code-free development, containerization technology is used to convert algorithms into operators, and users are provided with a process orchestration interface. Furthermore, algorithm research both for sudden fault and long-term aging failure detection is conducted and applied to the platform for industrial robot health monitoring experiments, by which the superiority of the proposed platform, in reality, is proven through positive results.

**Keywords:** health monitoring platform; data-driven; industrial robots

check for updates

**Citation:** Jiang, F.; Hu, C.; Liu, C.; Wang, R.; Zhu, J.; Chen, S; Zhang, J. PHIR: A Platform Solution of Data-Driven Health Monitoring for Industrial Robots. *Electronics* **2024**, *13*, 834. https://doi.org/10.3390/electronics13050834

Academic Editor: Carlo Mastroianni

Received: 12 January 2024
Revised: 16 February 2024
Accepted: 19 February 2024
Published: 21 February 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

Industrial robots (IR) have been widely used in industry and manufacturing for their flexibility, efficiency, and high precision [1]. According to the International Federation of Robotics, annual installations exceeded 500,000 in 2023 [2]. Abnormalities or malfunctions in IRs can lead to unplanned shutdowns of production lines, which will not only affect the yield but also bring about huge economic losses for related companies. Therefore, IR health condition monitoring has become a key to improving the safety and reliability of production systems. However, traditional methods, including technologies based on historical experience, mechanism models, and expert systems, suffer from complex modeling issues, a lack of shared mechanisms, or limited expert knowledge, and thus could not meet the requirement of large-scale robot health monitoring.

With the development of technologies such as the Internet of Things (IoT), big data, cloud computing, and artificial intelligence (AI), data-driven IR health monitoring methods are promising candidates for solving the above problems. To this end, efforts from two directions have been explored: platform construction and algorithm research. For platform construction, Wang et al. [3] provide platforms for IR health monitoring and have experimentally applied some traditional machine learning algorithms. However, they focus more

on platform construction to collect and store robot data, while the model development of IR health monitoring only stops at the introduction of simple machine learning methods. As for algorithms, some studies try to introduce deep learning approaches [4,5] or fuse physical mechanisms with neural networks [6,7] to learn the patterns better. Yet the application of these approaches is virtually unexplored since they are verified on open datasets or a single experimental IR, with a lack of attention to data sources and application deployment. On the other hand, the robot health monitoring task is essentially a multi-variate time series anomaly detection (MTSAD) problem. There has been a recent significant growth in research on this problem, and the above methods have not followed these advances well [8]. To summarize, the platform and algorithm studies described above are isolated and do not comprehensively consider all aspects of the process from deep model training to application deployment.

Specifically, there are the following challenges in applying deep learning or data-driven methods to IR health monitoring and forming services: (1) Deep learning is highly dependent on a large amount of data while industrial robot execution data are often stored in manufacturing execution systems (MES) and separated from sensor data. Data from different IRs and sensors cannot be managed and processed uniformly, and are difficult to directly feed into deep models. (2) Many machine learning/deep learning algorithms and preprocessing methods are complex for most practitioners in the industrial field, so they are not very skilled in developing and using these algorithms. Therefore, this limitation has become a threshold for the migration of the algorithms to the industry. (3) Recent MTSAD algorithms cannot be directly migrated to the IR health monitoring task because IRs perform different actions in factories and have complex and diverse faults, such as sudden motor failures and long-term wear failure of transmission components.

To address the above issues, we propose an innovative PHIR, which is the first relatively successful platform combining IoT systems with health monitoring algorithms in the IR field to our knowledge. First, we propose a cloud–edge–robot architecture to collect high-frequency time series into the cloud. Then, a process orchestration function is developed to enable pluggable integration of various IR-related algorithms, where they can be easily packed into containers and strung as services through a non-code editor. This orchestration function allows developers to focus on the development of the model itself, ignoring upstream and downstream processes such as data cleaning and model deployment. In addition, we have also proposed two deep-learning models for IRs and deployed them on our PHIR. AE-DTW uses autoencoders and modifies the loss function to make it easier to detect sudden faults such as servo motor shutdown and received a best F1 score of 0.84. The transformer-SD improves the traditional transformer with signal decomposition to better learn long-term patterns, which obtained an F1 score of 0.77 on our dataset. The PHIR has been applied in real-life scenarios for more than 2 years, which has been connected to 13 industrial robots and collected more than 100 million time series points. Furthermore, more than 20 components of different algorithms are contained in the PHIR, and its efficiency has been verified by extensive experiments.

The structure of this article is as follows: in Section 2, we introduce the related work; in Section 3, we focus on the system design and the employed IR health condition monitoring algorithms; and in Section 4, we present the results of the experiment.

## 2. Related Work

The maintenance of industrial robots has always been a crucial part of their management and application. In this section, we outline the development process of existing IoT systems and algorithms that are related to IR health monitoring.

Platforms. With the improvement of edge and cloud infrastructure [9–14], a large amount of work [15–17] has been conducted to manage industrial robots using the technology of IoT. In [18], the authors summarize the latest software technologies required to collect and manage all data through IoT devices that are deployed on production lines. Cloud robotics [15] suggests that robots could employ cloud infrastructures such

as databases and data centers to store and process data. The knowledge sharing over the cloud makes it possible to build intelligent robot applications including health condition management [17,19]. However, they are far from industrial sites and face poor real-time performance and the risk of privacy leakage. Therefore, cloud–edge IoT platforms are explored. For example, frameworks for IR health monitoring have been introduced in [3], where some traditional machine learning algorithms have been applied in experiments. However, these platforms only provide framework design but do not develop services for industrial users.

Algorithms. In recent years, with the emergence of new robot control algorithms and systems [20–22], data-driven health monitoring technology has received increasing attention in industry and academia [23,24] for its advantages of simple use, low costs, and suitability for large-scale applications. In this context, industrial equipment data of high frequency and large volume are difficult for humans to annotate. Therefore, related research mainly focuses on unsupervised learning. Ref. [23] assesses the health of IRs by predicting end-effector deviations based on vision. Ref. [24] predicts faults of IRs based on the k-nearest neighbor (KNN) and long short-term memory (LSTM) network and formulates predictive maintenance strategies for IRs based on knowledge graphs (KGs). Although these methods apply AI technology to the field of IR, they do not keep up with the latest advances in deep learning models, such as Transformer. In addition, the IR health monitoring task is essentially an MTSAD problem, while classification-based AD methods [25,26] face the challenge of data imbalance. Therefore, some research takes a process of augmentation and uses the labels [27–30], while some unrealistic anomalies may be introduced in studies. Normality-based anomaly detection methods [31–36] believe that normal data should have one or more characteristics, while data that do not meet them will be judged as abnormal. These methods are widely applied in various fields, rather than being tailored specifically for robots. Therefore, some studies try to use robot features [4,5] or fuse physical mechanisms with neural networks [6,7]. However, these approaches are only verified on open datasets or experimental IRs and have not yet been applied in industrial sites.

## 3. Methodology

For real-time computing and privacy considerations, the system consists of three layers, including the device, edge, and cloud, as shown in Figure 1. The device layer is the monitored objects such as 6-axle robots, actuators, and various sensors that collect environmental or video data from the factory, etc. The edge layer includes various computing devices that connect to the device layer through wired or wireless networks to acquire data and provide a local monitor for the staff on site. Finally, the cloud layer is a centralized computing center that receives data from the edge layer, processes them, trains the model, and delivers the model to the edge layer. Additionally, the cloud provides interfaces for defining their workflows to the users. Next, we will introduce the three primary functions in detail: data collection, workflow orchestration, and our health condition monitoring algorithms.

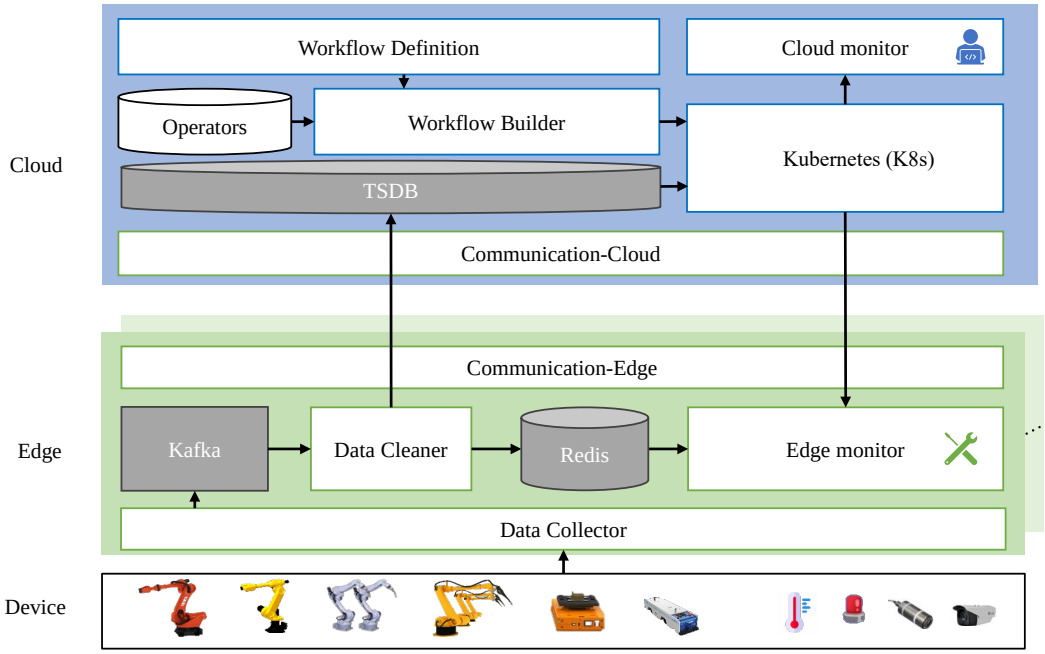



**Figure 1.** The framework of PHIR. The edge layer establishes direct connections to devices, ensuring privacy and real-time data processing. The cloud layer stores data and conducts complex training to ensure the performance of health condition monitoring.

### *3.1. Data Collection*

Data collection for IRs presents several challenges, including the following:

- Heterogeneous Data: Industrial robots generate data from various sources, such as sensors, cameras, and actuators. These data can be in different formats and structures, making it challenging to collect and integrate into a unified system. Thus, robust data processing techniques are required to ensure compatibility and consistency in handling this heterogeneity.
- Different Data Transmission Protocols: Industrial robots often use different data transmission protocols, depending on the devices and systems involved. Coordinating and collecting data from these diverse protocols can be complex and requires specialized integration mechanisms to ensure seamless data transfer.
- Aggregation of High-Frequency Massive Data: Industrial robots generate a significant amount of data at high frequencies, which poses challenges in massive data collecting, storing, and processing in real-time. Efficient aggregation techniques, along with scalable storage and processing infrastructure, are necessary to handle this high-frequency data flow effectively.

To address the challenges of data collection in industrial robots, we have implemented several strategies.

Firstly, we have customized schemas for different data types and stored them in regular databases, temporal databases, and unstructured databases. The system has two groups of entities, where the first group focuses on robot-related devices. This first group includes edge devices, robots, and robotic end effectors. The end effectors are installed on the wrist of a robotic arm to interact with the environment, and their type depends on the application scenario, such as polishing, welding, spraying, handling, and assembly. To handle the static and dynamic data of robots and effectors, their data are modeled as static (attributes) and dynamic (such as robot current and welding effectors) models separately. There is a bi-directional association between edges and robots, and dynamic models are associated with static models. The other group of entities refers to external sensors deployed at the edge to collect environmental data, and video cameras to monitor the site. These entity groups enable effective modeling and management of the various

components in the industrial environment, ensuring that data are collected accurately and efficiently for further analysis and decision making.

Secondly, in the Data Collector component, we have developed a flexible interface for data collection protocols such as OPC UA, MQTT, and TCP sockets. These protocol engines are loosely coupled with the subsequent processing, enabling us to add new protocols easily. Users can customize their own engines, ensuring that the output format conforms to the data schema. The flexibility of our data collection interface ensures that the system can handle diverse data transmission protocols and devices. It also enables quick integration with new protocols, so that the time and effort required to add new data sources to the system can be significantly reduced.

Thirdly, we have developed a Data Cleaner module on the edge layer. This module performs data integrity and correctness verification, as well as temporal alignment processing, which is essential for further analysis. Meanwhile, it compresses data to reduce bandwidth usage between edges and the cloud. These techniques ensure that the system can process and analyze the data quickly and efficiently, enabling optimal performance in industrial settings.

Finally, we have implemented a message queue mechanism for communication between cloud edges. The data are firstly saved in a Redis database, which makes the system more fault-tolerant, and data processing more efficiently and reliably.

### *3.2. Process Orchestration*

Our low code development environment is designed to simplify the implementation of deep learning algorithms in robotics for non-computer professionals. The environment leverages containerization technology, which allows us to encapsulate data pre-processing methods and deep learning algorithms into containers. These containers are portable and can be deployed across different environments, making it easier for users to access and use these components. To manage these containers, we use Kubernetes on the cloud to provide a scalable and efficient framework for deploying, managing, and scaling containerized applications. We can choreograph the processes involved in the implementation of deep learning algorithms to ensure that they are executed efficiently and reliably.

Before developing a complete health monitoring application, we compressed and packaged different algorithms into images, and stored them in the operator base, as Operators described in Figure 1. This also includes algorithms developed specifically for robots, which will be introduced in the next section. To launch an application from the raw algorithms, the user and our system need to follow three main steps: building a workflow, creating Operators (Ops), and deploying an instance, among which the users only do the first step.

Users can define their workflows from the cloud through our Workflow Definition interface, which provides a graphical tool based on Scalable Vector Graphics (SVG) that transforms the algorithms and data streams into nodes and connections between them, as shown in Figure 2. Algorithms and data sources are drowned as some points, and the flows are shown as the lines between them. Users can modify the information of the nodes, such as parameters, image addresses, and node attributes. It provides a user-friendly interface where users can drag-and-drop Ops over a graphical display to build their workflow. Once the workflow is submitted, the Workflow Builder compiles it into a workflow description of a K8s Custom Resource Definition (CRD). This includes the description of the whole data flow and each Ops. The compiled workflow is then ready for deployment.

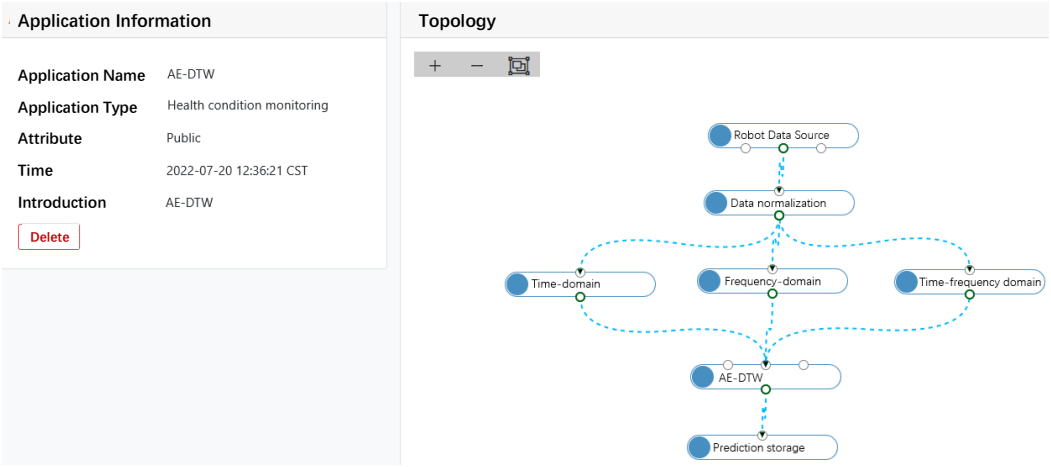



**Figure 2.** The workflow definition interface, where nodes are data sources or processing processes and arrows represent the data flow. Take AE-DTW as an example; data from the data source flow to the normalization component and are then subjected to feature extraction in time, frequency, and time–frequency domains. The features are then fed into the AE-DTW algorithm and the results predicted by the model are stored.

Then the Workflow Builder pulls the algorithm images to the running environment and executes the programs according to the CRD. Now the algorithms come into Ops, which are the basic computational units that perform a custom task on data streams. It is worth noting that we design a user-transparent framework to provide a common environment for embedding the Docker images in Operators, which supports multi-threaded communication between upstream and downstream and saves state for stateful Ops.

To deploy an instance of the workflow, we employ the K8s CustomController, which acts as a vigilant overseer of newly created instances and handles resource management. When a deployment request is received, the CustomController interacts with the K8s API server to allocate pods and services. The Scheduler, a crucial component, employs an intelligent scheduling mechanism to determine the optimal placement of pods, ensuring the smooth and efficient execution of the workflow. Once the Scheduler has determined the best placement, the API server takes charge. It pulls the necessary Docker images and orchestrates their execution within the designated pods. This step ensures that the required resources are provisioned and ready for use. Once the pods are successfully launched and operational, the CustomController receives an update regarding the state of the workflow. This information allows it to maintain an accurate record process of workflow and make any necessary adjustments or updates.

### *3.3. Health Condition Monitoring Algorithms*

The universal anomaly detection algorithm has not been improved due to the characteristics of industrial robots such as complex and diverse abnormal signals, difficulty in capturing abnormal states, and long aging time. In this paper, we extract motion-independent features of IRs. Then we propose two algorithms to detect sudden anomalies (AE-DTW) and learn long-term age patterns (Transformer-SD), which will be introduced below.

#### 3.3.1. AE-DTW

IR time series data are difficult for humans to label due to the high frequency and complex pattern of various abnormal types such as mutation point anomalies, context anomalies, and subsequence anomalies that appear in the collected datasets. Furthermore, data collected over different motions may disturb extracting features related to health. In this paper, we propose an unsupervised anomaly detection model based on autoencoder reconstruction error to capture the probability distribution of industrial robot data under normal conditions in latent space. To eliminate the influence of different motions, we

extract the motion-independent features from time (amplitude fluctuation index), frequency, and time–frequency (signal-to-noise ratio) domains. During the reconstruction process, we introduce Dynamic Time Warping (DTW) in the reconstruction loss, which serves as a health indicator for industrial robots.

Anomaly pattern. The abnormal patterns in the electromechanical signals of the industrial robot body are very complex. Common abnormal patterns in time series data include mutation point anomalies, context anomalies, and subsequence anomalies, as shown in Figure 3. In addition, these abnormal signals may appear in the sensor signals of various components of the industrial robot, which increases the complexity of patterns.

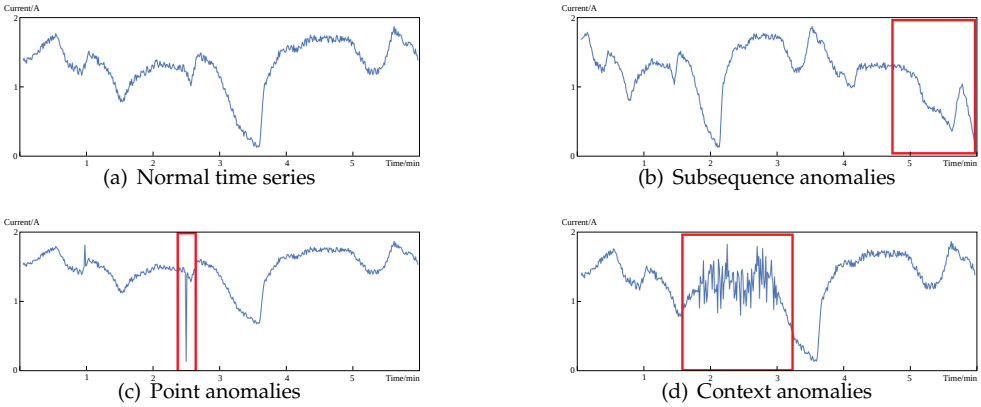



**Figure 3.** Anomaly patterns. The red boxes are anomalies.

For example, IRs generally consist of five parts: reducer, servo motor, connection device, sensor, and controller. Among them, the reducer may suffer from gear wear, gear pitting, and transmission shaft breakage. These faults may appear as mutation point anomalies, context anomalies, and subsequence anomalies in the time series data. Servo motors may experience stator failure, bearing failure, and torque reduction. These failures are manifested in data as distortion of electromechanical signals and signal values exceeding the rated range. The connection device may wear out, causing the movement accuracy of the robot to decrease, which manifests itself as sequence anomalies or range changes in the timing data. Sensors may have faults such as component damage or poor contact, which manifests as point anomalies or context anomalies in the time series data. The controller may have instruction errors or control output errors, which may appear as mutation point anomalies or context anomalies in the time series data.

The model. As shown in Figure 4, the model discards unimportant components in the data and only retains the main characteristics of the signals by fusing the deep representations and the motion-independent features. This paper uses the AutoEncoder model to reduce dimensionality and extract the representations, which is a nonlinear model that can better capture complex patterns with lower computational complexity.

We use the dilated causal convolutional autoencoder, which consists of a residual layer and a nonlinear dimensionality reduction layer. The residual layer generally adopts the structure of convolution $\rightarrow$ normalization $\rightarrow$ convolution $\rightarrow$ normalization, and the input and output of the residual block are connected. The nonlinear dimensionality reduction layer adopts the structure of dilated causal convolution to increase the receptive field while eliminating the grid effect. The output of the last layer of the encoder is the low-dimensional features. The decoder takes the output of the encoder as input, and the output of the decoder is the reconstructed signal $\hat{X} \in \mathbb{R}^{C \times L}$. The decoder is mainly composed of a deconvolution layer and a residual layer. The former is used to increase the dimensionality of the low-dimensional representation. The structure of the residual layer is the same as that of the encoder.

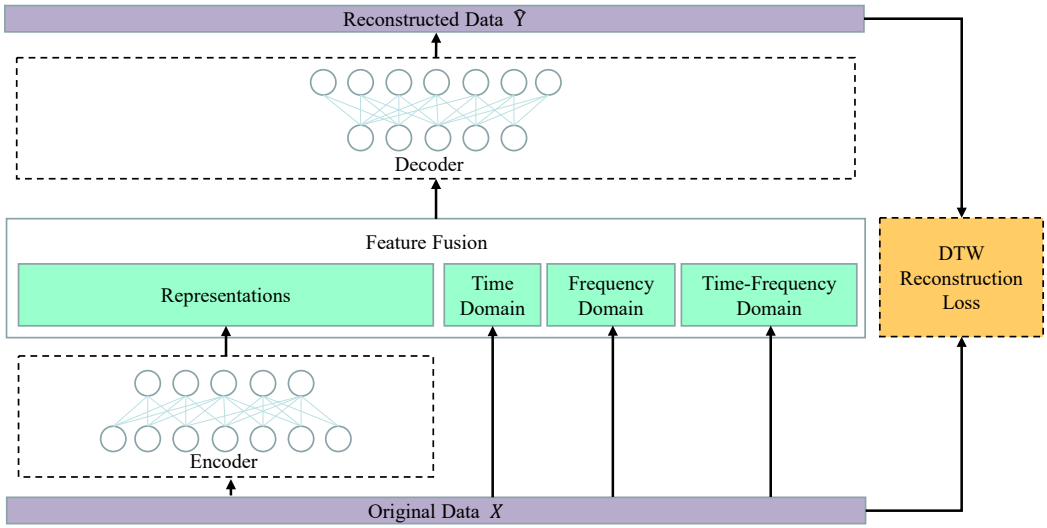



**Figure 4.** The overall architecture of the proposed AE-DTW model, where the purple blocks are original and reconstructed data, the blue blocks are learned features, and the yellow block is the loss.

Finally, the reconstruction error between the input and the predicted signal is used as the health condition index, and whether the IR is abnormal is determined by comparing the offset of the reconstruction error between the monitoring data and the standard data.

The loss function. It is necessary to find a suitable distance metric to make the probability of anomaly points $p(x_t|x_{t-1}\cdots x_{t-\tau}) \rightarrow 0$ and the probability of anomaly sequences $p(x_t\cdots x_{t+h}|x_{t-1}\cdots x_{t-\tau}) \rightarrow 0$. The traditional Euclidean distance assumes a one-to-one correspondence between the prediction sequence and the target sequence, and cannot effectively capture the distortion anomalies of the subsequence. Therefore, this paper uses DTW as the loss function of the reconstruction model, which contains two parts and thus can effectively distinguish the shape difference and distortion difference between the prediction sequence and the target sequence.

Consider that there are $N$ number of industrial robot signals $\mathcal{A} = \{x_i\}_{i\in\{1:N\}}$, for each input signal $x_i = (x_i^1, \cdots, x_i^n) \in \mathbb{R}^{p\times n}$, our model first compresses the input signal into the hidden space and then uses the hidden space state to reconstruct it. Then $\hat{y}_i = (\hat{y}_i^1, \cdots, \hat{y}_i^n) \in \mathbb{R}^{p\times n}$ can be obtained. Define the real input signal as $y_i = (y_i^1, \cdots, y_i^n) \in \mathbb{R}^{p\times n}$, where $y_i = x_i$. Then the DTW loss can be expressed as

$$\mathcal{L}_{DTW} = \alpha\mathcal{L}_{shape}(\hat{y}_i, y_i) + (1-\alpha)\mathcal{L}_{temporal}(\hat{y}_i, y_i), \tag{1}$$

where $\mathcal{L}_{shape}$ measures the shape difference between the predicted and the real IR signal, and $\mathcal{L}_{temporal}$ measures the distortion difference, both of which require the regular matrix between the predicted and real signals. Define the regular path as $A \subset \{0,1\}^{n\times n}$, if $\hat{y}_i^h$ and $y_i^j$ is aligned, then $A_{h,j} = 1$. Otherwise $A_{h,j} = 0$. An effective regular path is recorded as $\mathcal{A}_{n\times n}$. Its starting point and end point are the coordinates of the upper left corner $(0,0)$ and the lower right corner $(n,n)$ of the regular matrix, where points can only move through the directions $\rightarrow, \downarrow, \searrow$ each time. Let $\Delta(\hat{y}_i, y_i) = [\delta(\hat{y}_i^h, y_i^j)]_{h,j}$ represents the regular matrix of the predicted and real IR electromechanical signals, $\delta(\cdot)$ as $\hat{y}_i^h$ and $y_i^j$. Then the dynamic normalized distance is

$$DTW(\hat{y}_i, y_i) = min_{A\in\mathcal{A}_{n\times n}} < A, \Delta(\hat{y}_i, y_i) >, \tag{2}$$

where $< \cdot >$ represents matrix multiplication. We formulate the shape loss based on DTW into

$$\mathcal{L}_{shape}(\hat{y}_i, y_i) = DTW_\gamma(\hat{y}_i, y_i) \quad = -\gamma log\left(\sum_{A \in \mathcal{A}_{n,n}} exp(-\frac{< A, \Delta(\hat{y}_i, y_i) >}{\gamma})\right) \tag{3}$$

according to the findings in [37].

We also include a timing skew index. Specifically, the optimal regular path between the predicted and the real IR signal is $A^*$. The distortion index of the industrial robot signal measures the difference between the optimal regular path $A^*$ and the first diagonal matric and is formulated as

$$TDI(\hat{y}_i, y_i) =< A^*, \Omega > \quad =< argmin_{A \in \mathcal{A}_{n,n}} < A, \Delta(\hat{y}_i, y_i) >, \Omega >, \tag{4}$$

where $\Omega$ is a square matrix of size $n \times n$, and $\Omega(h, j) = \frac{(h-j)^2}{n^2}$, which measures the difference between the optimal regular path $A^*$ and the first diagonal matrix. However, in the above formula, the *argmin* operator is not differentiable, so we consider using $\nabla_\Delta DTW_\gamma(\hat{y}_i, y_i)$ instead of $A^*$, which can be

$$\begin{aligned} A^*_\gamma &= 1/Z \sum_{A \in \mathcal{A}_{n,n}} Aexp^{-\frac{<A, \Delta(\hat{y}_i, y_i)>}{\gamma}} \\ Z &= \sum_{A \in \mathcal{A}_{n,n}} exp^{-\frac{<A, \Delta(\hat{y}_i, y_i)>}{\gamma}} \end{aligned} \tag{5}$$

To sum up, the distortion loss of industrial robot signals based on DTW can be expressed as

$$\mathcal{L}_{temporal}(\hat{y}_i, y_i) =< A^*_\gamma, \Omega > \quad = \frac{1}{Z} \sum_{A \in \mathcal{A}_{n,n}} exp^{-\frac{<A, \Delta(\hat{y}_i, y_i)>}{\gamma}} \tag{6}$$

The DTW-based distortion loss penalizes the degree of distortion. In industrial robot health monitoring, distortion loss can capture cycle offset anomalies caused by bearing failures, sensor failures, and aging and loosening of transmission belts, because such failures appear as timing distortions in timing data. The final loss function is

$$\mathcal{L}_{DTW} = -\gamma log\left(\sum_{A \in \mathcal{A}_{n,n}} exp(-\frac{< A, \alpha\Delta(\hat{y}_i, y_i) + (1-\alpha)\Omega >}{\gamma})\right) \tag{7}$$

#### 3.3.2. Transformer-SD

In practice, industrial robot health monitoring technology often focuses on the abnormal patterns of each component itself, while ignoring the overall aging trend. To capture the long-term aging trend of the nonlinear process for the whole IRs, we propose the Signal Decomposition Transformer model (Transformer-SD) for IRs inspired by Autoformer [38]. The main idea of Transformer-SD is to use the current signal of the servo motor to predict the IR axis position since the transmission ratio decreases as the robot ages.

Although the self-attention-enhanced Transformer model can capture the long-distance dependencies in the signals of industrial robots, two flaws still exist. First, the IR signals contain complex intrinsic patterns. Specifically, it is difficult for Transformer to capture the interrelationships and changes between various components. Secondly, the space complexity increases quadratically with the input length, and directly modeling fine-grained long-term sequences meets the memory bottleneck, which seriously affects the practical application of the Transformer. In recent years, many studies have been trying to solve the efficiency problem of Transformer by replacing the original dense point-by-point attention mechanism with a sparse self-attention. To address the above issues, this paper decomposes

the signal with STL in a component of the deep neural network model and integrates it into the Transformer model based on the sparse self-attention mechanism, making it effectively capture long-term dependencies.

The basic structure of Transformer-SD is shown in Figure 5. Specifically, the length of the input 6-axis IR electromechanical signal is $I$, and the length of the predicted end-effector position sequence is $O$. To improve the ability to capture complex industrial robot electromechanical signal patterns, this paper introduces STL decomposition into the Transformer model and gradually decomposes the original industrial robot electromechanical signal $\mathcal{X}$ into trend terms $\mathcal{X}_t$ and periodic term $\mathcal{X}_s$. In actual implementation, this article uses a sliding average to filter out periodic fluctuations. The remaining part after the sliding average is the periodic term of the signal. Given $\mathbb{X} \in \mathbb{R}^{L \times d}$, trend term $\mathcal{X}_t = AvgPool(Padding(\mathcal{X}))$, periodic term $\mathcal{X}_s = \mathcal{X} - \mathcal{X}_t$, where $\mathcal{X}_t, \mathcal{X}_s \in \mathbb{R}^{L \times d}$. We use $\mathcal{X}_s, \mathcal{X}_t = SeriesDecomp(\mathcal{X})$ to represent the whole process.

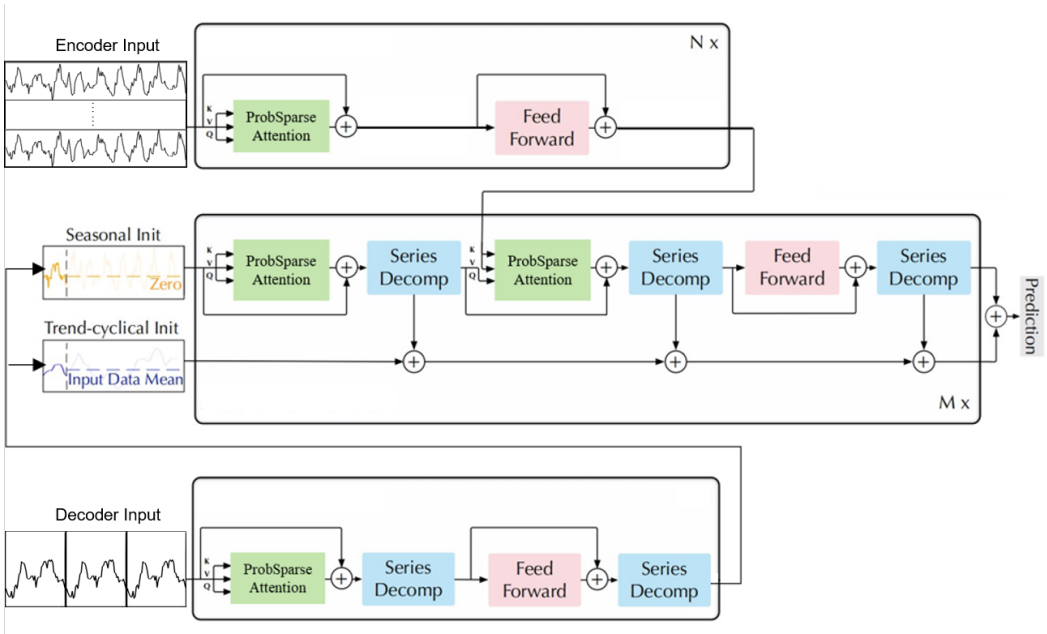



**Figure 5.** The overall architecture of the proposed Transformer-SD model, where green blocks are the self-attention layers, the blue blocks are the series decomposition layers, and the pink blocks are the feed forward layers.

The signal decomposition Transformer model is divided into two parts: the encoder and the decoder. The input of the encoder is the 6-axis IR signal with length $O$ after the predicted time point $\mathcal{X}_{en} \in \mathbb{R}^{O \times d}$, and the input of the decoder is the industrial robot end effector position sequence periodic term $\mathcal{X}_{des} \in \mathbb{R}^{(3I+O) \times d}$ of length $(3I + O)$ and the trend term $\mathcal{X}_{det} \in \mathbb{R}^{(3I+O) \times d}$ of length $(3I + O)$. Each input of the decoder contains two parts. The first part is the industrial robot end-effector position sequence $\mathcal{X}_{de}$ with length $(3I)$ before the prediction time point. Since the Transformer model captures long time series dependencies, this part of the input is three time series with an interval of $\tau$, that is,

$$\mathcal{X}_{de} = [\mathcal{X}_{t-I:t}, \mathcal{X}_{t-I-\tau:t-\tau}, \mathcal{X}_{t-I-2\tau:t-2\tau}] \tag{8}$$

where $t$ represents the prediction time point, and $\mathcal{X}_{de}$ is used to provide the information of the latest period of the output sequence. The second part is a scalar placeholder of length

$O$. The overall process of the model based on long-term dependent prediction errors can be formally expressed as

$$\mathcal{X}_{des}, \mathcal{X}_{det} = SeriesDecomp(\mathcal{X}_{de}) \tag{9}$$

$$\mathcal{X}_{des} = Concat(\mathcal{X}_{des}, \mathcal{X}_0) \tag{10}$$

$$\mathcal{X}_{det} = Concat(\mathcal{X}_{det}, \mathcal{X}_{Mean}), \tag{11}$$

where $\mathcal{X}_{des}, \mathcal{X}_{det} \in \mathbb{R}^{3I \times d}$ represent the periodic and trend term in $\mathcal{X}_{de}$, while $\mathcal{X}_0, \mathcal{X}_{Mean} \in \mathbb{R}^{O \times d}$ represent the 0 and $\nabla$, respectively, and $\mathcal{X}_{de}$ the mean-filled placeholders. The main function of the signal decomposition Transformer encoder is to capture the periodic terms of the 6-axis IR electromechanical signal. The output of the encoder contains the periodic terms, which can be used as mutual information to help the decoder predict more accurately. The encoding process can be formally expressed according to the following formula. Assuming that the encoder has a total of $N$ layers, the $l$th layer encoder can be expressed as $\mathcal{X}_{en}^{l} = Encoder(\mathcal{X}_{en}^{l-1})$, its specific implementation is

$$\mathcal{S}_{en}^{l,1}, _ = SeriesDecomp(SelfAttention(\mathcal{X}_{en}^{l-1})) \tag{12}$$

$$\mathcal{S}_{en}^{l,2}, _ = SeriesDecomp(FeedForward(\mathcal{S}_{en}^{l,1}) + \mathcal{S}_{en}^{l,1}), \tag{13}$$

where the trend item is omitted. $\mathcal{X}_{en}^{l} = \mathcal{S}_{en}^{l,2}, l \in \{1, \cdots, N\}$ represents the $l$th layer coding The output of the detector, $\mathcal{S}_{en}^{l,i}, i \in \{1, 2\}$ represents the periodic term of the 6-axis IR signal obtained after passing through the $l$ layer.

The decoder consists of two parts, namely the accumulation component used to capture trend items and the self-attention component for periodic items. The former separates the periodic terms of the effector position sequence from the original time series in a progressive manner, while the latter can utilize the periodic information in the 6-axis IR signal extracted by the encoder. Assuming that the Transformer-SD decoder consists of $M$ layer decoders, the hidden variable output by the encoder is $\mathcal{X}_{en}^{N}$, and the process of the $l$ layer decoder can be expressed as $\mathcal{X}_{de}^{l} = Decoder(\mathcal{X}_{de}^{l-1}, \mathcal{X}_{en}^{N})$. Specifically,

$$\mathcal{S}_{de}^{l,1}, \mathcal{T}_{de}^{l,1} = SeriesDecomp(SelfAttention(\mathcal{X}_{de}^{l-1}) + \mathcal{X}_{de}^{l-1}) \tag{14}$$

$$\mathcal{S}_{de}^{l,2}, \mathcal{T}_{de}^{l,2} = SeriesDecomp(SelfAttention(\mathcal{S}_{de}^{l,1}, \mathcal{X}_{en}^{N}) + \mathcal{S}_{de}^{l,1}) \tag{15}$$

$$\mathcal{S}_{de}^{l,3}, \mathcal{T}_{de}^{l,3} = SeriesDecomp(feedForward(\mathcal{S}_{de}^{l,2}) + \mathcal{S}_{de}^{l,2}) \tag{16}$$

$$\mathcal{T}_{de}^{l} = \mathcal{T}_{de}^{l-1} + \mathcal{W}_{l,1} * \mathcal{T}_{de}^{l,1} + \mathcal{W}_{l,2} * \mathcal{T}_{de}^{l,2} + \mathcal{W}_{l,3} * \mathcal{T}_{de}^{l,3}, \tag{17}$$

where $\mathcal{X}_{de}^{l} = \mathcal{S}_{de}^{l,3}, l \in \{1, \cdots, M\}$ is the output of the $l$th layer, $\mathcal{S}_{de}^{l,i}, \mathcal{T}_{de}^{l,i}, i \in \{1, 2, 3\}$ represents the periodic term and trend term of the effector position sequence obtained by the $l$ layer of the decoder, $\mathcal{W}_{l,i}, i \in \{1, 2, 3\}$ represents the linear mapping used to extract the trend term of the effector position sequence.

## 4. Experiment

We have deployed the PHIR in reality and collected data from various robots and sensors. In the past 2 years, it has been connected to thirteen six-axis IRs produced by ROKAE and SIASUN, including XB4, XB7, and SR10C. They are located in Beijing and Hangzhou, executing polishing, welding, and assembly processes. On this basis, the data were organized into a dataset and the effect of AE-DTW and Transformer-SD models was further evaluated.

### *4.1. Application Results and Datasets*

A case study is conducted with five IRs and an Automated Guided Vehicle (AGV) in a production line for robot assembly. As Figure 6 shows, to assemble the industrial robots, components of the robot have to pass through approximately five main workstations in

sequence through AGV. Each workstation consists of IRs, positioning clamping fixtures, and gas devices. PHIR is downward connected to the AGV controller and MES with a wired connection and upward connects to the cloud through Communication-Edge services. It is equipped with Kafka and Redis components for stable collection and processing of the 100 Hz high-frequency data from robots to the cloud. Meanwhile, PHIR provides services in terms of managing the data, monitoring the health of IRs, and editing the workflows. The workflows contain 23 Ops including denoising, normalization, dimensionality reduction, health monitoring, and other functions. During the continuous running time of over 1 year, PHIR alarmed for 11 times with an accuracy of 81.8%. The average response time for monitoring the health condition is 5 s, which is much faster than a staff member noticing anomalies. The nature of Kafka and container technology ensures the low latency of data collection and the stability of the system running.




**Figure 6.** Application scenario of IRs in an assembly workshop. The key instruments including AGV, AGV controller, MSE, and our edge device are pointed out with arrows.

In addition, to comprehensively verify our algorithm, we organized the data collected by 13 IRs connected to PHIR. Similarly, we collected the current, speed, and position data of each axis of IR at a frequency of 100 Hz, forming a total of 18 dimensions. The working life of an industrial robot is 3~5 years, so we upsample the data for 10 min every 7 days. According to the type of robot failure, the data are divided into two subsets, where A is the data from robots with sudden failures, and B is from robots with long-term failures due to aging and wear. We have divided all abnormal data into the test set. The specific data can be seen in Table 1.

**Table 1.** Statistics of the two sub-datasets used in experiments.

| No. | Feature | Train | Test | Anomalies |
|---|---|---|---|---|
| A | 18 | 2,160,000 | 2,400,000 | 3.38% |
| B | 18 | 4,080,000 | 4,680,000 | 2.97% |

### *4.2. Metric*

In the experiment, this paper selects the Best F1 Score and AUC value as evaluation indicators. The F1 is a commonly used indicator in anomaly detection problems. It is the harmonic average of *precision* and *recall*. The calculation process is depicted as follows:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{18}$$

where $TP$ (true positives) represents the number of samples predicted to be abnormal. $FP$ (false positives) represents the number of samples predicted to be abnormal but normal.

$FN$ (false negatives) is the number of samples predicted to be normal but abnormal. AUC is the area under the receiver operating characteristic (ROC) curve of the machine learning model. The abscissa of the ROC curve is the false positive rate (FPR), and the ordinate is the true positive rate (TPR). The two calculation methods are described as follows:

$$TPR = \frac{TP}{TP + FN} \tag{19}$$

$$FPR = \frac{FP}{FP + TN} \tag{20}$$

### *4.3. Baselines and Implementation Details*

The proposed approaches are compared against the following sudden and long-term aging failure detection methods.

Sudden Failure Detection Baselines. Four commonly used sudden failure detection baselines are adopted: PCA, t-Stochastic Neighbor Embedding (t-SNE), Autoencoder (AE), and Autoencoder with Convolution (AE-C). PCA and t-SNE are completed with the Scikit-learn library. They can reduce the dimensionality of the 18-dimensional robot signal to 6 dimensions, and then input the signal into the One-class SVM model to detect anomalies. AE has six fully connected layers, and AE-C contains three convolutional layers and three deconvolutional layers. Both use Mean Squared Error (MSE) as the loss function.

Long-term Aging Failure Detection Baselines. Then, four long-term aging failure anomaly detection methods are used to measure the performance of detection, which include the LSTM network, Gated Recurrent Unit (GRU), self-attention Transformer, and Probabilistic Sparse Transformer (Transformer-PS) [39]. LSTM consists of an encoder and a decoder. The encoder contains three layers of LSTM units. The dimension of the hidden state is 64, which is aligned with the structure of the decoder. The Dropout layer deactivates neurons with a probability of 0.5 and is used to connect each layer of LSTM units. Teacher-forcing mode is used when training the model, and the decoder uses the real value as the input of the next time step with a probability of 0.5. The GRU model structure is similar to LSTM, but it uses GRU units internally.

For AE-DTW, the original time series is cut into many equal-length samples by the sliding windows with window_size of 100 and step of 25. Furthermore, since Transformer-SD is prediction-based, we take every 300 timestamps (3s) as input and predict the next 100 ones (1s) instead of using sliding windows. The hyper-parameters of AE-DTW and Transformer-SD are shown in Table 2. The latent space dimension of AE-DTW is 64 dimensions. AE-DTW encoder and decoder have similar structures, with three convolutional layers and deconvolutional layers, respectively. The convolutional layer has a kernel of 3 and a stride of 1, and the deconvolution layer has a kernel of 5 and a stride of 2. The expansion coefficient of the convolution is 2, and the activation function is ReLU. Transformer-SD encoder contains two layers of Transformer modules, and the decoder contains three layers of Transformer modules. Lastly, all the models are built with PyTorch 1.7 and trained on the NVIDIA Tesla V100 GPU of our platform.

**Table 2.** Hyper-parameters of AE-DTW and Transformer-SD.

| Parameter | AE-DTW | Transformer-SD |
|---|---|---|
| optimizer | Adam | Adam |
| window_size | 100 | – |
| time_step | 25 | – |
| input_len | – | 300 |
| output_len | – | 100 |
| learning_rate | 0.001 | 0.001 |
| batch_size | 256 | 256 |
| epochs | 100 | 100 |

### *4.4. Results*

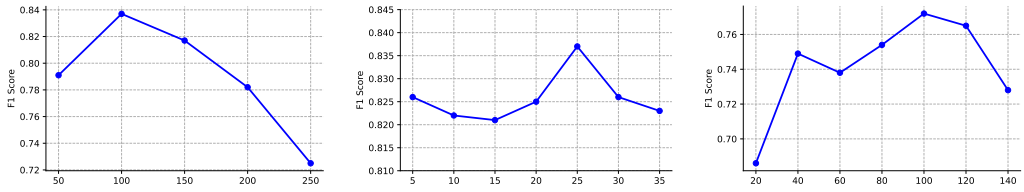
We report the affiliation F1-score and AUC in Table 3. From a dataset perspective, the F1 and UCR of AE-DTW on subset A reach 0.837 and 0.752. On subset B, the F1 and UCR of Transformer-SD reached 0.772 and 0.693, respectively. These show that a single model is insufficient for detecting various IR faults, which drives the requirement for the cooperation of multiple types of models. In terms of the methods of detecting sudden failure, three conclusions can be drawn: First, PCA and t-SNE perform poorly on two subsets, showing that shallow methods are difficult to adapt to complex IR data. Second, the F1 of AE-DTW is higher than that of AE by more than 0.2, indicating that the causal convolution structure could improve the performance of detecting. That is because the causal convolution structure reduces the number of parameters and increases the depth of the network, which better captures the main components of the IR time series. Third, AE-DTW performs better than AE-C, indicating that the DTW loss function is more relevant to the nature of the time series. From the view of the long-term aging failure detection method, LSTM and GRU perform more poorly than Transformer and this indicates the efficiency of the self-attention mechanism. On the other hand, Transformer-SD with AUC over 0.69 performs well, showing that combining Transformer and STL decomposition is an effective way to extract long-term features of time series. Overall, the proposed AE-DTW and Transformer-SD outperform all baselines on both subsets, demonstrating the effectiveness and robustness of sudden and long-term aging failure detection models.

**Table 3.** F1 score and AOC of AE-DTW and Transformer-SD on sub-datasets A and B. The best results are shown in bold and the next best ones are underlined.

| Metric | Subset A | | Subset B | |
|---|---|---|---|---|
| | **Best F1** | **AUC** | **Best F1** | **AUC** |
| PCA | 0.528 | 0.449 | 0.318 | 0.223 |
| t-SNE | 0.540 | 0.451 | 0.511 | 0.472 |
| AE | 0.618 | 0.665 | 0.535 | 0.538 |
| AE-C | <u>0.784</u> | <u>0.721</u> | 0.622 | 0.573 |
| LSTM | 0.610 | 0.439 | 0.621 | 0.525 |
| GRU | 0.649 | 0.527 | 0.611 | 0.510 |
| Transformer | 0.736 | 0.611 | 0.624 | 0.536 |
| Transformer-PS | 0.717 | 0.679 | <u>0.707</u> | 0.623 |
| AE-DTW | **0.837** | **0.752** | 0.630 | <u>0.647</u> |
| Transformer-SD | 0.582 | 0.483 | **0.772** | **0.693** |

### *4.5. Hyper-parameters Analysis*

In this section, a sensitivity analysis on subsets A and B is performed to study three main parameters: window size $Ws$, window step $Ts$, and output length $Ol$. Figure 7a,b show the effect of $Ws$ and $Ts$ on the overall performance of AE-DTW, where the y-axis stands for the F1-score. For subset A, we notice that DTW-AE is sensitive to $Ws$ with a change amplitude exceeding 0.1, and when $Ws = 100$, F1 achieves the best. Thus appropriate window size $Ws$ should be selected according to the depth of the model. Our AE-DTW model achieves the best performance when $Ts = 25$. It is also worth noting that when $Ws = 100$, our model is less sensitive to the $Ts$ value. Figure 7c shows the results of varying the window step output length $Ol$ of Transformer-SD in a range between 20 and 140. The model performs best on the subset B when $Ol = 100$, which suggests that increasing the output time length can improve the performance. However, a larger length can harm the performance as it exceeds the length of time that the model can be predictive based on the input data.




(a) *Ws* (Subset A) (b) *Ts* (Subset A) (c) *Ol* (Subset B)

**Figure 7.** Three sensitivity analysis experiments on subsets A and B, where (**a**,**b**) are the hyper-parameters window size *Ws* and window step *Ts* of AE-DTW, and (**c**) is the output length *Ol* of Transformer-SD.

## 5. Conclusions

We developed an IR health monitoring platform PHIR that combines the IoT technology with data-driven methods, which collects data at high frequency in real-time and provides users with a customized interface for deep algorithm services, providing solutions for deep mining of multi-source data in the industrial field. In addition, we propose two deep models, AE-DTW for sudden faults and Transformer-SD for long-term ones. We deployed the system in a real production environment, collected operating data of various types and models of industrial robots for two years, and established a dataset, which proved the practicability of our system. Experiments based on this dataset demonstrate the effectiveness and advancement of our proposed model. We believe that our work can accelerate the application of deep learning algorithms in the industrial field, facilitate the integration of IT and OT, and contribute to the development of intelligent manufacturing. Due to the data-hungry nature of deep models, that is, the more training data, the better the performance, the future work will focus on the following two aspects. First of all, it is inevitable that more robots will be connected and more data collected, and it is necessary to improve the concurrency capability and high availability of the PHIR system. On the other hand, algorithm optimization should also be carried out to reduce the pressure on data collection and enable a more lightweight model to be deployed.

**Author Contributions:** Conceptualization, F.J. and C.H.; Methodology, F.J.; Validation, C.L.; Writing—original draft, F.J. and S.C.; Writing—review and editing, R.W. and J.Z. (Juan Zhang); Formal analysis, J.Z. (Juan Zhang) and J.Z. (Jianyong Zhu); Investigation, S.C. and J.Z. (Juan Zhang); Supervision, J.Z. (Jianyong Zhu), J.Z. (Juan Zhang) and R.W.; Project administration F.J. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is partially supported by the Fundamental Research Funds for the Central Universities (No. 501QYJC2023121001).

**Data Availability Statement:** The data presented in this study are available on request from the first author or the corresponding author. The data are not publicly available due to commercial restrictions. Due to the nature of this research, participants of this study did not agree for their data to be shared publicly, so supporting data are not available.

**Conflicts of Interest:** Author Fei Jiang and Chengyun Hu were employed by the company Waytous. Author Chongwei Liu was employed by the company JD Logistics. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## References

1. Fei, Y.; Shi, P.; Li, Y.; Liu, Y.; Qu, X. Formation control of multi-agent systems with actuator saturation via neural-based sliding mode estimators. *Knowl.-Based Syst.* **2024**, *284*, 111292. [CrossRef]
2. IFR. Executive Summary: World Robotics 2023 Industrial Robots. 2023. Available online: https://ifr.org/img/worldrobotics/Executive_Summary_WR_Industrial_Robots_2023.pdf (accessed on 20 November 2023).
3. Wang, R.; Mou, X.; Sun, J.; Liu, P.; Guo, X.; Wo, T.; Liu, X. Cloud-edge collaborative industrial robotic intelligent service platform. In Proceedings of the 2020 IEEE International Conference on Joint Cloud Computing, Oxford, UK, 3–6 August 2020; pp. 71–77.

4. Chen, T.; Liu, X.; Xia, B.; Wang, W.; Lai, Y. Unsupervised Anomaly Detection of Industrial Robots Using Sliding-Window Convolutional Variational Autoencoder. *IEEE Access* **2020**, *8*, 47072–47081. [CrossRef]
5. Xiao, H.; Zeng, H.; Jiang, W.; Zhou, Y.; Tu, X. HMM-TCN-based health assessment and state prediction for robot mechanical axis. *Int. J. Intell. Syst.* **2022**, *37*, 10476–10494. [CrossRef]
6. Long, J.; Mou, J.; Zhang, L.; Zhang, S.; Li, C. Attitude data-based deep hybrid learning architecture for intelligent fault diagnosis of multi-joint industrial robots. *J. Manuf. Syst.* **2021**, *61*, 736–745. [CrossRef]
7. Long, J.; Qin, Y.; Yang, Z.; Huang, Y.; Li, C. Discriminative feature learning using a multiscale convolutional capsule network from attitude data for fault diagnosis of industrial robots. *Mech. Syst. Signal Process.* **2023**, *182*, 109569. [CrossRef]
8. Hsu, C.Y.; Liu, W.C. Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing. *J. Intell. Manuf.* **2021**, *32*, 823–836. [CrossRef]
9. Zhu, J.; Yang, R.; Hu, C.; Wo, T.; Xue, S.; Ouyang, J.; Xu, J. Perphon: A ml-based agent for workload co-location via performance prediction and resource inference. In Proceedings of the ACM Symposium on Cloud Computing, Santa Cruz, CA, USA, 20–23 November 2019; p. 478.
10. Yang, R.; Hu, C.; Sun, X.; Garraghan, P.; Wo, T.; Wen, Z.; Peng, H.; Xu, J.; Li, C. Performance-aware speculative resource oversubscription for large-scale clusters. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1499–1517. [CrossRef]
11. Yeung, G.; Borowiec, D.; Yang, R.; Friday, A.; Harper, R.; Garraghan, P. Horus: Interference-Aware and Prediction-Based Scheduling in Deep Learning Systems. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 88–100. [CrossRef]
12. Zhu, J.; Yang, R.; Sun, X.; Wo, T.; Hu, C.; Peng, H.; Xiao, J.; Zomaya, A.Y.; Xu, J. QoS-aware co-scheduling for distributed long-running applications on shared clusters. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 4818–4834. [CrossRef]
13. Wen, Z.; Yang, R.; Qian, B.; Xuan, Y.; Lu, L.; Wang, Z.; Peng, H.; Xu, J.; Zomaya, A.Y.; Ranjan, R. JANUS: Latency-Aware Traffic Scheduling for IoT Data Streaming in Edge Environments. *IEEE Trans. Serv. Comput.* **2023**, *16*, 4302–4316. [CrossRef]
14. Mou, X.; Sun, J.; Zhong, Y.; Wo, T. HyCU: Hybrid Consistent Update for Software Defined Network. In Proceedings of the 2023 IEEE International Conference on Joint Cloud Computing (JCC), Athens, Greece, 17–20 July 2023; pp. 86–92.
15. Kuffner, J. Cloud-enabled humanoid robots. In Proceedings of the Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, Nashville, TN, USA, 6–8 December 2010.
16. Mohanarajah, G.; Hunziker, D.; D'Andrea, R.; Waibel, M. Rapyuta: A cloud robotics platform. *IEEE Trans. Autom. Sci. Eng.* **2014**, *12*, 481–493. [CrossRef]
17. Li, R. Cloud Robotics-Enable Cloud Computing for Robots. Technical Report. 2014. Available online: https://sites.google.com/site (accessed on 21 January 2015).
18. Cerquitelli, T.; Pagliari, D.J.; Calimera, A.; Bottaccioli, L.; Patti, E.; Acquaviva, A.; Poncino, M. Manufacturing as a data-driven practice: Methodologies, technologies, and tools. *Proc. IEEE* **2021**, *109*, 399–422. [CrossRef]
19. Hu, G.; Tay, W.P.; Wen, Y. Cloud robotics: Architecture, challenges and applications. *IEEE Netw.* **2012**, *26*, 21–28. [CrossRef]
20. Chen, L.; Sun, L.; Yang, T.; Fan, L.; Huang, K.; Xuanyuan, Z. Rgb-t slam: A flexible slam framework by combining appearance and thermal information. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 5682–5687.
21. Niestanak, V.D.; Moshaii, A.A.; Moghaddam, M.M. A new underactuated mechanism of hand tendon injury rehabilitation. In Proceedings of the 2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM), Tehran, Iran, 25–27 October 2017; pp. 400–405.
22. Abbasimoshaei, A.; Chinnakkonda Ravi, A.K.; Kern, T.A. Development of a new control system for a rehabilitation robot using electrical impedance tomography and artificial intelligence. *Biomimetics* **2023**, *8*, 420. [CrossRef]
23. Izagirre, U.; Andonegui, I.; Eciolaza, L.; Zurutuza, U. Towards manufacturing robotics accuracy degradation assessment: A vision-based data-driven implementation. *Robot. Comput.-Integr. Manuf.* **2021**, *67*, 102029. [CrossRef]
24. Wang, X.; Liu, M.; Liu, C.; Ling, L.; Zhang, X. Data-driven and Knowledge-based predictive maintenance method for industrial robots for the production stability of intelligent manufacturing. *Expert Syst. Appl.* **2023**, *234*, 121136. [CrossRef]
25. Sharma, S.; Bellinger, C.; Krawczyk, B.; Zaiane, O.; Japkowicz, N. Synthetic oversampling with the majority class: A new perspective on handling extreme imbalance. In Proceedings of the 2018 IEEE international conference on data mining (ICDM), Singapore, 17–20 November 2018; pp. 447–456.
26. Liu, P.; Guo, X.; Wang, R.; Chen, P.; Wo, T.; Liu, X. CSMOTE: Contrastive Synthetic Minority Oversampling for Imbalanced Time Series Classification. In Proceedings of the Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, 8–12 December 2021; Mantoro, T., Lee, M., Ayu, M.A., Wong, K.W., Hidayanto, A.N., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 447–455.
27. Ruff, L.; Vandermeulen, R.A.; Franks, B.J.; Müller, K.R.; Kloft, M. Rethinking assumptions in deep anomaly detection. *arXiv* **2020**, arXiv:2006.00339.
28. Carmona, C.U.; Aubet, F.X.; Flunkert, V.; Gasthaus, J. Neural contextual anomaly detection for time series. *arXiv* **2022**, arXiv:2107.07702.
29. Liu, P.; Guo, X.; Shi, B.; Wang, R.; Wo, T.; Liu, X. Non-salient region erasure for time series augmentation. *Front. Comput. Sci.* **2022**, *16*, 166349. [CrossRef]
30. Liu, P.; Guo, X.; Chen, P.; Shi, B.; Wo, T.; Liu, X. Adaptive Shapelets Preservation for Time Series Augmentation. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; pp. 1–8. [CrossRef]

31. Ruff, L.; Vandermeulen, R.; Goernitz, N.; Deecke, L.; Siddiqui, S.A.; Binder, A.; Müller, E.; Kloft, M. Deep one-class classification. In Proceedings of the International Conference on Machine Learning (PMLR), Stockholm, Sweden, 10–15 July 2018; pp. 4393–4402.
32. Zong, B.; Song, Q.; Min, M.R.; Cheng, W.; Lumezanu, C.; Cho, D.; Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
33. Qiu, C.; Pfrommer, T.; Kloft, M.; Mandt, S.; Rudolph, M. Neural Transformation Learning for Deep Anomaly Detection beyond Images. In Proceedings of the 38th International Conference on Machine Learning, (PMLR), Online, 18–24 July 2021; Volume 139.
34. De Haan, P.; Löwe, S. Contrastive Predictive Coding for Anomaly Detection. *arXiv* **2021**, arXiv:2107.07820.
35. Wang, R.; Liu, C.; Mou, X.; Gao, K.; Guo, X.; Liu, P.; Wo, T.; Liu, X. Deep Contrastive One-Class Time Series Anomaly Detection. In Proceedings of the 2023 SIAM International Conference on Data Mining (SDM), Minneapolis, MN, USA, 27–29 April 2023; pp. 694–702.
36. Mou, X.; Wang, R.; Wang, T.; Sun, J.; Li, B.; Wo, T.; Liu, X. Deep Autoencoding One-Class time Series Anomaly Detection. In Proceedings of the ICASSP 2023—IEEE International Conference on Acoustics, Speech and Signal Processing, Rhodes Island, Greece, 4–10 June 2023; pp. 1–5.
37. Cuturi, M.; Blondel, M. Soft-dtw: A differentiable loss function for time-series. In Proceedings of the International Conference on Machine Learning (PMLR), Sydney, Australia, 6–11 August 2017; pp. 894–903.
38. Wu, H.; Xu, J.; Wang, J.; Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 22419–22430.
39. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada, 20–27 February 2021; Volume 35, pp. 11106–11115.


**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.