Proceedings of the 35th European Safety and Reliability & the 33rd Society for Risk Analysis Europe Conference
Edited by Eirik Bjorheim Abrahamsen, Terje Aven, Frederic Bouder, Roger Flage, Marja Ylönen
 Published by Research Publishing, Singapore.
doi: 10.3850/978-981-94-3281-3_ESREL-SRA-E2025-P9570-cd

Robot Fault Detection using Digital Twin and Deep Learning

Haibo Li

Laboratoire Génie Industriel, CentraleSupélec, Université Paris-Saclay, France.
E-mail: haibo.li@centralesupelec.fr

Wenxuan Hu

Sorbonne-Université, France. E-mail: wenxuan.hu@etu.sorbonne-universite.fr

Zhiguo Zeng

Chair of Risk and Resilience of Complex Systems, Laboratoire Genie Industriel, Centralesupelec, Université Paris-Saclay, France. E-mail: zhiguo.zeng@centralesupelec.fr

Reliability and lifetime of robots is critical for modern manufacturing systems, as robots have been widely used in manufacturing, and their failure could lead to substantial financial losses. Predictive Maintenance (PdM) has emerged as an effective strategy, utilizing historical data and prognostic models to anticipate maintenance needs. Digital twins simulate the behavior of real systems and connect to the real system using sensors in real-time, and have shown potentials to booster the performance of predictive maintenance algorithms. In this paper, we present an open-source demonstration platform of using digital twin for robot PdM. A data-driven digital twin is developed first for a robot to predict its temperature during operation. By leveraging time-series data collected in real-time, including motor temperature, voltage, and position, a data-driven algorithm is developed to detect abnormality in motor temperature response. An innovative Recursive Prediction Update (RPU) technique is proposed, which replaces fault-contaminated data with predicted values in real-time, significantly enhancing accuracy of abnormility detection. Results show that the integration of Digital Twins and RPU improves fault detection performance, offering valuable insights for predictive maintenance under limited failure data conditions.

Keywords: Digital twin, predictive maintenance, robotic arm, deep learning, fault diagnosis.

1. Introduction

Robotic systems are integral to automated manufacturing, requiring high reliability and extended lifespans. Unexpected robot malfunctions cause substantial financial losses. For example, costs due to unexpected breakdown of robots are estimated at 100,000 to 200,000 euros per day (Mathur et al., 2001). Maintenance expenses account for 60% to 70% of production system lifecycle costs (Dhillon, 2006), making maintenance optimization critical for operations efficiency and competitiveness of production systems.

To address these challenges, Predictive Maintenance (PdM) has been proposed, leveraging historical data and prognostic models to predict optimal maintenance times. PdM includes diagnostics, which analyze failure causes, and prognostics, which model degradation processes to estimate Remaining Useful Life (RUL) (Aivaliotis et al., 2021). However, the effectiveness of PdM is often constrained by limited historical failure data. Digital twins, a key Industry 4.0 technology (Fuller et al., 2020), offer a promising solution by simulating system behavior and collecting real-time data through condition monitoring. These simulations can train machine learning models for fault diagnosis and RUL prediction (Ellefsen et al., 2019). Reference Aivaliotis et al. (2019) combined degradation models with digital twins of industrial robots to develop PdM models. However, existing studies primarily explore integration potential, lacking detailed insights into advanced machine learning for fault detection and RUL prediction, as well as system-level maintenance planning and scheduling.

In this study, we build an open-source demonstration platform that combined digital twins with AI algorithms to explore fault diagnosis tech-

niques under conditions of limited historical fault data (Court et al., 2024). By considering the temporal dependencies in time series, a data-driven algorithm is developed that utilizes historical data under normal operating conditions, including motor voltage, temperature, and position, as inputs to detect whether the temperature response of a specific motor at a given moment is normal. This study introduces an innovative application of Recursive Prediction Update (RPU), which employs predicted data to iteratively correct input data in real-time, significantly enhancing prediction accuracy. The findings of this study provide valuable insights and a reference for enhancing predictive maintenance under conditions of limited historical fault data.

2. A demonstration platform for robot PdM with digital twins

The structure of the platform is shown in Fig.1, and the structure of the robot is shown in Fig.2. Additional details can also be found on GitHub at the following link: https://github.com/sonic160/digital_twin_robot. The robotic arm is controlled by a Raspberry Pi running Ubuntu 18.04 and equipped with an HD 120° Wide Angle Camera for target recognition and tracking. Featuring six degrees of freedom driven by Smart Serial Bus Servos, the arm is topped with a gripper for grasping objects. Coordinated motor operation enables tasks like goods transfer, with each motor providing feedback on position, voltage, and temperature signals, amounting to 18 features. In addition to the physical robotic entity, we have developed a virtual simulation model of the robot, which connects to the physical entity. Through sensors and data streams, it retrieves the real-time operational status of the physical robot and visualizes it in a virtual environment, providing real-time visualization of the physical system.

Communication between the physical and virtual systems is achieved via ROS, where real-time data is published by the physical robot and subscribed to by the virtual robot. Additionally, a MATLAB-based Condition Monitoring System (CMS) visualizes data streams, stores timestamped data, and tracks the equipment's health status, aiding in anomaly detection and proactive maintenance. This system facilitates data-driven decision-making to prevent equipment failures and optimize performance.

3. Dataset description

3.1. *Experimental Design and Data Collection*

The robot is programmed to move in a random pattern that simulates a real pick-and-place operation. To collect training data under normal operation conditions, the robotic arm is programmed to perform 1,000 pick-and-place tasks. Each consisting of five movements: returning to the initial position, moving to the pick-up location, grasping the goods, transporting them to the placement location, and releasing them. The process repeats for the next task, ensuring efficient and comprehensive data collection.

Pick-up and placement positions are randomly generated within the robot's 3D workspace, using inverse kinematics to determine the required trajectories. This approach enriches the dataset with varied scenarios. Data are collected at a frequency of 10 Hz, resulting in a time series of 104,000 data points, each containing 18 features (position, voltage, and temperature signals from six motors), over a duration of 10,400 s.

3.2. *Data Preprocessing and Failure Injection*

First, the collected time series data were split into training, validation, and test sets in a ratio of 7:1.5:1.5, corresponding to 72,800, 15,600, and 15,600 data points, respectively. Importantly, the temporal order of the data was preserved during the split to maintain its time dependency.

Since the collected data represents normal operating conditions, fault patterns were simulated to study fault detection capabilities. In this study, we simulated the temperature variations of a motor during a fault, specifically the process where the motor stalls for a certain period due to a fault and then resumes normal operation. We selected motor 6, located at the base of the robotic arm, as the target for this analysis. The fault temperature

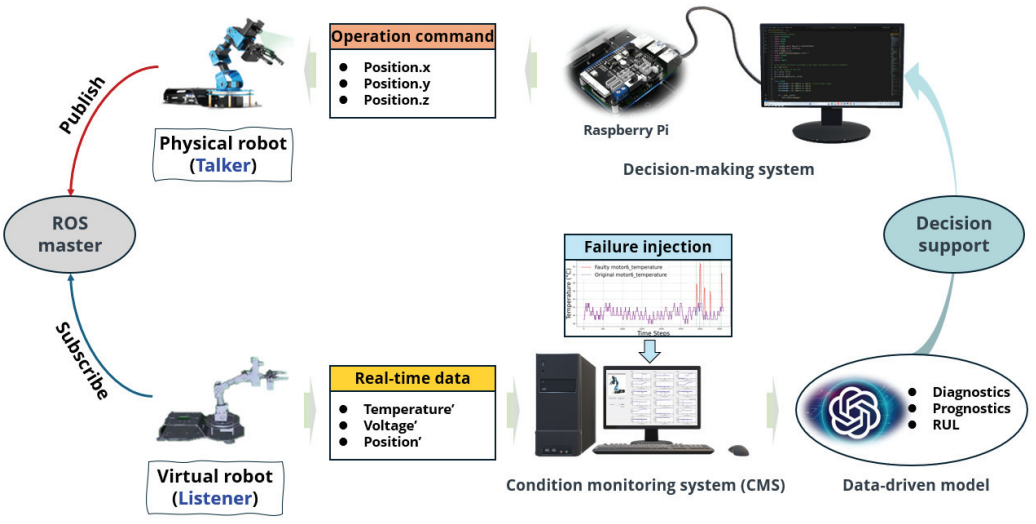

Fig. 1.: Open-source demonstration platform for predictive maintenance of robots using digital twins

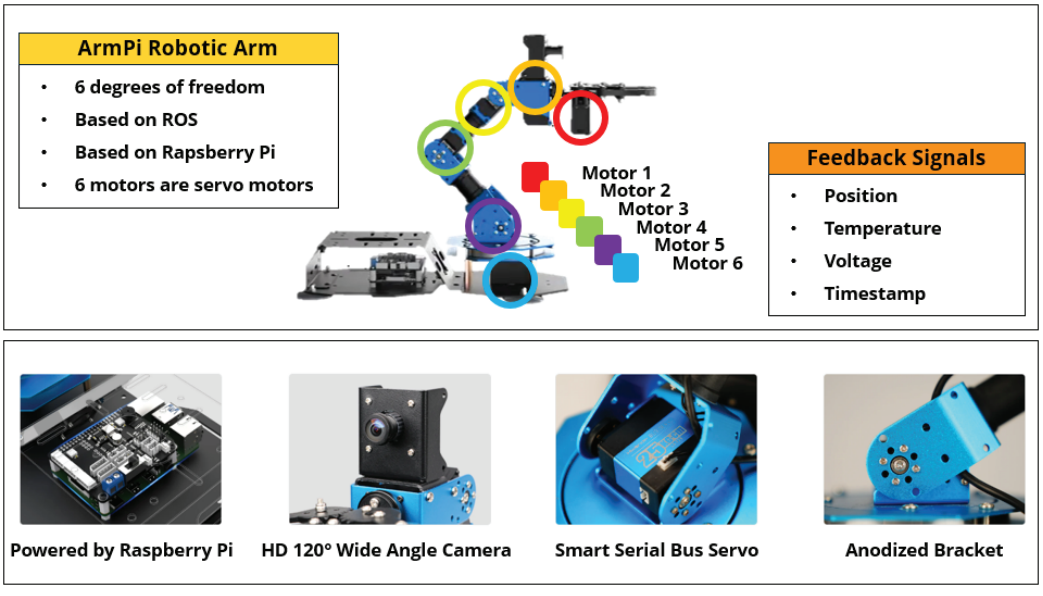

Fig. 2.: ArmPi robotic arm features and components

variation was modeled based on a semi-empirical formula as Eq.1, as shown in Fig.3. A total of 20 faults were simulated and randomly injected into the temperature features of motor 6 in the test set, resulting in 667 fault points, to evaluate the model's fault detection performance.

4. Data-driven digital twin modeling and fault diagnosis

In this section, we first develop a data-driven digital twin model to predict the operational temperature of the robot under normal operation condition. The digital twin model is, then, used to detect failure patterns.

$$T(t) = \begin{cases} T_{\text{env}} + \Delta T_{\max}\left(1 - e^{-\frac{t}{\tau_1}}\right), & 0 \le t \le t_{\text{rise}}, \\ T_{\text{env}} + \left(T_{\text{rise}}(t_{\text{rise}}) - T_{\text{env}}\right) e^{-\frac{t - t_{\text{rise}}}{\tau_2}}, & t > t_{\text{rise}}, \end{cases} \tag{1}$$

where, T_{env}: ambient temperature, $\Delta T_{\max}$: maximum temperature rise, τ_1: time constant for heating, τ_2: time constant for cooling, t_{rise}: end time of the heating phase.

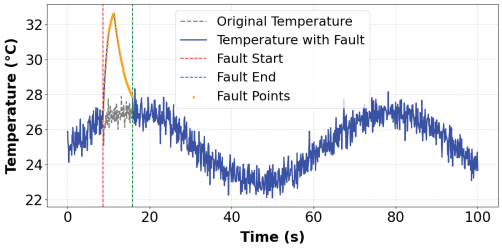

Fig. 3.: Semi-empirical motor temperature curve

4.1. *Digital twin modeling*

Since the collected data are high-frequency time series with significant temporal dependencies, the parameter changes between consecutive points are minimal. Such subtle variations are insufficient for determining whether a single point represents a fault. Therefore, to better capture the temporal dynamics of the data, we adopted a sliding window approach to segment the time series into Input Window and Prediction Window, thereby constructing dataset samples.

Specifically, one typical sample's Input Window is defined as shown in Eq. 2:

$$\{X(t') \mid t' \in [t-39, t-10]\} \tag{2}$$

and one typical sample's Prediction Window is defined as shown in Eq. 3:

$$\{T_6(t') \mid t' \in [t-9, t]\} \tag{3}$$

Here, $X(t')$ represents the feature matrix at current point t', including the following features for six motors:

- $T_i(t')$: The temperature of the i-th motor ($i = 1, 2, \ldots, 6$),
- $P_i(t')$: The joint position of the i-th motor,
- $V_i(t')$: The working voltage of the i-th motor.

With this design, the Input Window captures the past 30 points of multi-motor feature data, while the Prediction Window represents the trends for the next 10 points. Notably, the last point of the Prediction Window, $T_6(t)$, corresponds to the current point t, which is used to determine whether the temperature exceeds the threshold.

During the preprocessing, we applied a sliding window operation to the time series with a step size of 1: $\{X(t') \mid t' \in [t-39, t-10]\} \rightarrow \{X(t') \mid t' \in [t-38, t-9]\} \rightarrow \ldots$, gradually generating the next set of training samples. The same preprocessing was applied to the training, validation, and test sets, resulting in 72,761, 15,561, and 15,561 samples, respectively.

Next, we trained a regression model using the training set data and evaluated its fitting performance on the validation set. During evaluation, the model predicts the Prediction Window $\{T_6(t') \mid t' \in [t-9, t]\}$ based on the Input Window $\{X(t') \mid t' \in [t-39, t-10]\}$. Since we only focus on the last point of each sample, the predicted value of the last point in the Prediction Window, $\hat{T}_6(t)$, was compared to the true value $T_6(t)$, and the residual ϵ is defined as:

$$\epsilon = \|\hat{T}_6(t) - T_6(t)\| \tag{4}$$

4.2. *LSTM modeling*

In this paper, we use an LSTM as the deep learning model used for the regression modeling. LSTM is an effective and commonly used algorithm for processing time series data. It is an improved version of recurrent neural networks, as depicted in Fig. 4. Compared to traditional RNN, it introduces the gate mechanism, allowing it to better handle long-term dependencies and

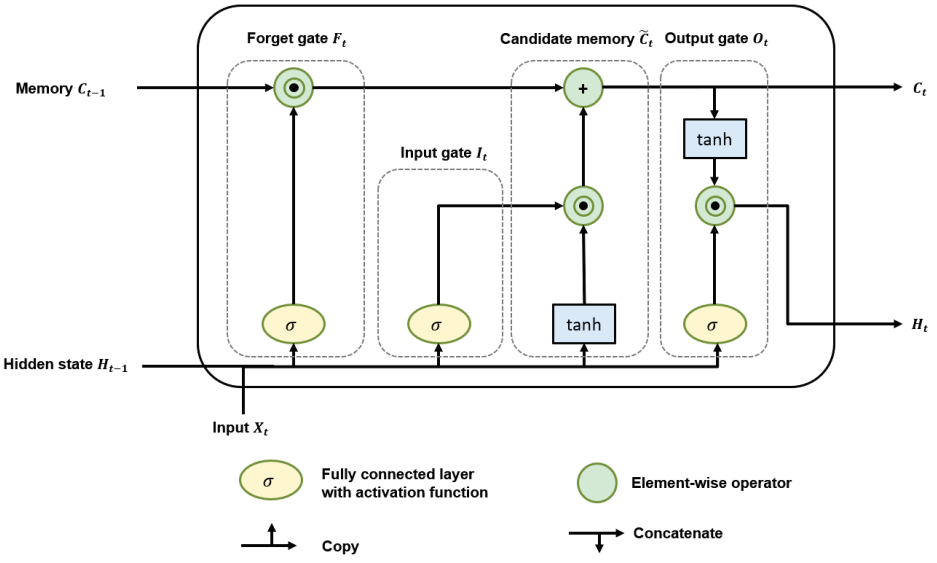

Fig. 4.: Long Short-Term Memory Architecture

prevent issues of vanishing or exploding gradients (Hochreiter, 1997).

Assuming that there are h hidden units, the batch size is n, and the input feature size is d, so the input is $\boldsymbol{X}_t \in \mathbb{R}^{n\times d}$, the previous time step's hidden unit is $\boldsymbol{H}_{t-1} \in \mathbb{R}^{n\times h}$, at this time step t, the input gate is $\boldsymbol{I}_t \in \mathbb{R}^{n\times h}$, the forget gate is $\boldsymbol{F}_t \in \mathbb{R}^{n\times h}$, the output gate is $\boldsymbol{O}_t \in \mathbb{R}^{n\times h}$. The details are as follows:

$$\boldsymbol{I}_t = \sigma\left(\boldsymbol{X}_t\boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hi} + \boldsymbol{b}_i\right) \quad (5)$$

$$\boldsymbol{F}_t = \sigma\left(\boldsymbol{X}_t\boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hf} + \boldsymbol{b}_f\right) \quad (6)$$

$$\boldsymbol{F}_o = \sigma\left(\boldsymbol{X}_t\boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{ho} + \boldsymbol{b}_o\right) \quad (7)$$

where, $\boldsymbol{W}_{xi}$, $\boldsymbol{W}_{xf}$, $\boldsymbol{W}_{xo} \in \mathbb{R}^{d\times h}$ and $\boldsymbol{W}_{hi}$, $\boldsymbol{W}_{hf}$, $\boldsymbol{W}_{ho} \in \mathbb{R}^{h\times h}$ are weight parameters, $\boldsymbol{W}_i$, $\boldsymbol{W}_f$, $\boldsymbol{W}_o \in \mathbb{R}^{1\times h}$ are the bias parameters.

Another component is the candidate memory cell $\tilde{C}_t \in \mathbb{R}^{n\times h}$. Unlike the gates, the candidate memory cell utilizes the tanh activation function.

$$\tilde{\boldsymbol{C}}_t = \tanh\left(\boldsymbol{X}_t\boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hc} + \boldsymbol{b}_c\right) \quad (8)$$

where, $\boldsymbol{W}_{xc} \in \mathbb{R}^{d\times h}$ and $\boldsymbol{W}_{hc} \in \mathbb{R}^{h\times h}$ are the weight parameters, $\boldsymbol{b}_c$ are bias parameters. In this neural network structure, the input gate $\boldsymbol{I}_t$ controls how much new data from $\tilde{\boldsymbol{C}}_t$ is adopted, while the forget gate $\boldsymbol{F}_t$ determines how much data from the memory unit $\boldsymbol{C}_{t-1}$ is retained. This mechanism helps better capture long-range dependencies in sequences and alleviates the issue of gradient explosion:

$$\boldsymbol{C}_t = \boldsymbol{F}_t \bigodot \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \bigodot \tilde{\boldsymbol{C}}_t \quad (9)$$

Finally, we need to define how to compute the hidden state $\boldsymbol{H}_t \in \mathbb{R}^{n\times h}$, which is where the output gate comes into play. In the long short-term memory network, it is simply the tanh transformation of the memory unit, controlled by the output gate. This ensures that the values of $\boldsymbol{H}_t$ always remain within the interval $(-1, 1)$:

$$\boldsymbol{H}_t = \boldsymbol{O}_t \bigodot \tanh(\boldsymbol{C}_t) \quad (10)$$

In this study, the selected hyperparameters of the model are shown in Table 1.

Table 1.: Model hyperparameters and descriptions

Parameter Name	Value	Description
Input_dim / d	(30, 18)	The number of input features for each time step in LSTM.
Hidden_dim / h	256	The number of hidden units in the LSTM layer, determining the hidden state size.
Output_dim	(10, 1)	The number of future time steps to predict.
Num_layers	3	The number of stacked LSTM layers in the model.
Dropout	0.2	The dropout rate applied between LSTM layers to prevent overfitting.
Epochs	100	The maximum number of training epochs.
Learning_rate	0.0001	The learning rate for the Adam optimizer.
Loss_function	MSELoss	The loss function used to compute the error between predictions and true values.
Optimizer	Adam	The optimizer used to update model parameters during training.

4.3. *Fault detection based on digital twins*

The maximum temperature residual observed in the validation set is defined as the Maximum Residual Threshold $\epsilon_{\max}$:

$$\epsilon_{\max} = \max \|\hat{T}_6(t) - T_6(t)\| \tag{11}$$

which represents the upper bound of the prediction error for motor 6's temperature on the validation set. A smaller threshold $\epsilon_{\max}$ indicates a stronger representational capability of the model for the validation data.

Therefore, when the regression model is tested on the test set with injected faults, if the residual of the last point in the Prediction Window satisfies:

$$\|\hat{T}_6(t) - T_6(t)\| > \epsilon_{\max} \tag{12}$$

the current time point $T_6(t)$ can be identified as anomalous. Conversely, it is considered normal data without failure.

Undoubtedly, when testing on the test set, as faults have already been injected into the test set, if the data in the input window is contaminated by the injected fault data, i.e., introducing the temperature rise trend of the injected fault, this will inevitably lead to inaccurate predictions of the prediction window. Specifically, this results in predicted values being overestimated, causing $\hat{T}_6(t)$ to approach $T_6(t)$. Consequently, certain data points may fail to be detected as faults, leading to missed alarms. Therefore, when $\|\hat{T}_6(t) - T_6(t)\| = \epsilon_t > \epsilon_{\max}$, it can be considered that the input window has been contaminated by fault data.

To address this issue, once $\epsilon_t > \epsilon_{\max}$, we replace $\{T_6(t') \mid t' \in [t-14, t-10]\}$ with $\{\hat{T}_6(t') \mid t' \in [t-14, t-10]\}$, which can be regarded as eliminating the contaminated data in the input window. Thus, the input window at the point t becomes: $\{T(t') \mid t' \in [t-39, t-15]\} \cup \{\hat{T}_6(t') \mid t' \in [t-10, t-14]\}$. This online prediction correction process could be called Recursive Prediction Update (RPU).

5. Results

Two widely used regression performance indices, R Square (R^2) and Mean Squared Error (MSE), along with four commonly used classification metrics, Precision, Recall, F1-Score, and Accuracy, are utilized to comprehensively evaluate the performance of the applied algorithm.

The loss during the training process of the LSTM model is shown in Fig. 5. From this loss curve, the model appears to be gradually converging, with consistent performance on both the training and validation sets. There are no obvious signs of overfitting or underfitting. This trend in-

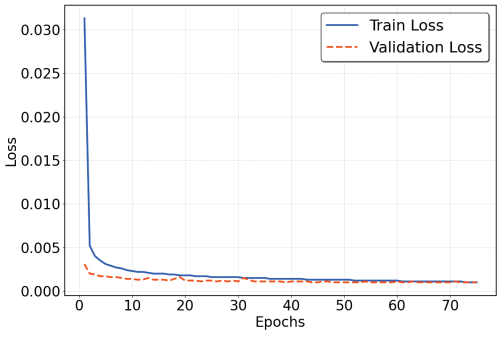

Fig. 5.: Train loss and validation loss for training process

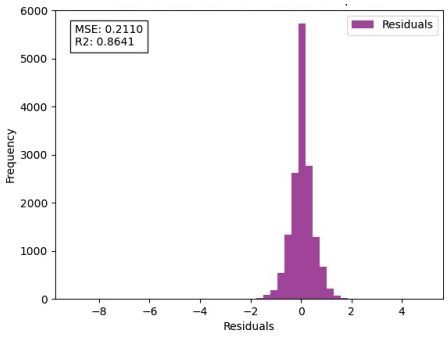

Fig. 6.: The residuals distribution of validation set

dicates that the model training process is stable and effective. The residuals between the predicted and observed values on the validation set are also shown in Fig.6. By observing Fig.6, it is obvious that most of the residuals fall within the range of [-2.5, 2.5]. Based on Eq. 11 and for safety margin considerations, it can be concluded that $\epsilon_{\max} = 3$.

Next, the model is applied to the test set injected with faults, with the data construction process referenced in Section 3.2. A total of 677 fault points were injected into the test dataset, which consists of 15,561 samples, meaning there are a total of 15,561 last points across all samples. During the application of the model for regression predictions on the test set, once the residual of $\hat{T}_6(t)$ and $T_6(t)$ exceeds $\epsilon_{\max}$, the point is labeled as a predicted fault and compared with the true labels of the injected faults.

To mitigate the impact of fault injections on prediction performance, Recursive Prediction Update (RPU) is applied. Specifically, when the residual exceeds $\epsilon_{\max}$, the last five points of motor 6's temperature in the current input window are replaced with the previously predicted values for these points. This replacement reduces the contamination of the input window caused by fault data, thereby enhancing the model's classification performance. The impact of applying or not applying RPU on prediction performance is illustrated in Fig. 7 and Fig. 8, the comparison of classification metrics is shown in Fig. 9.

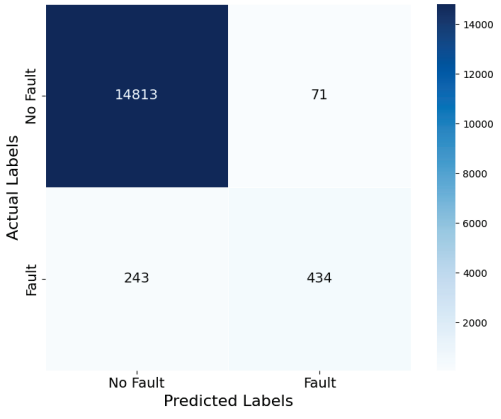

Fig. 7.: The confusion matrix of test set without RPU

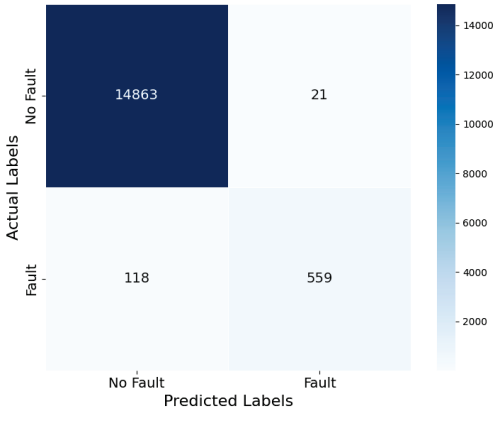

Fig. 8.: The confusion matrix of test set with RPU

The results demonstrate that applying Recur-

sive Prediction Update (RPU) significantly enhances the fault detection performance of the model. With RPU, the model achieves higher accuracy (99.11%), precision (96.38%), recall (82.57%), and F1-score (88.94%) compared to the scenario without RPU. The reduction in false negatives and false positives, as observed in the confusion matrices, underscores the effectiveness of RPU in mitigating the impact of fault data contamination on the input window. These improvements validate the potential of RPU as a robust strategy for improving predictive maintenance systems, ensuring more reliable fault detection in robotic systems.

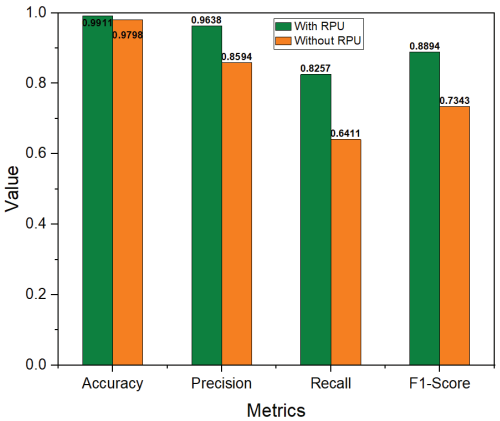

Fig. 9.: Comparison of classification metrics with and without RPU on the test set

6. Conclusions

This study explores a fault detection approach for robotic arm motors through a data-driven digital twin and deep learning. The proposed method utilizes a data-driven model to predict motor temperature anomalies based on multi-sensor input. The introduction of the Recursive Prediction Update (RPU) mechanism further enhances the model's prediction accuracy by mitigating the influence of fault-contaminated input data. Experimental results demonstrate that the proposed approach achieves high classification performance, with clear improvements observed when RPU is applied. This study provides a practical framework for predictive maintenance in scenarios with limited fault data, offering valuable insights for advancing fault diagnosis and maintenance planning in robotic systems.

Acknowledgement

The research of Zhiguo Zeng is partically funded by the French Research Council under contract number ANR-22-CE-10-0004 (Project DFT).

References

Aivaliotis, P., Z. Arkouli, K. Georgoulias, and S. Makris (2021). Degradation curves integration in physics-based models: Towards the predictive maintenance of industrial robots. *Robotics and computer-integrated manufacturing 71*, 102177.

Aivaliotis, P., K. Georgoulias, and G. Chryssolouris (2019). The use of digital twin for predictive maintenance in manufacturing. *International Journal of Computer Integrated Manufacturing 32*(11), 1067–1080.

Court, K. M., X. M. Court, S. Du, and Z. Zeng (2024). Use digital twins to support fault diagnosis from system-level condition-monitoring data.

Dhillon, B. S. (2006). *Maintainability, maintenance, and reliability for engineers*. CRC press.

Ellefsen, A. L., V. Æsøy, S. Ushakov, and H. Zhang (2019). A comprehensive survey of prognostics and health management based on deep learning for autonomous ships. *IEEE Transactions on Reliability 68*(2), 720–740.

Fuller, A., Z. Fan, C. Day, and C. Barlow (2020). Digital twin: Enabling technologies, challenges and open research. *IEEE access 8*, 108952–108971.

Hochreiter, S. (1997). Long short-term memory. *Neural Computation MIT-Press*.

Mathur, A., K. F. Cavanaugh, K. R. Pattipati, P. K. Willett, and T. R. Galie (2001). Reasoning and modeling systems in diagnosis and prognosis. In *Component and Systems Diagnostics, Prognosis, and Health Management*, Volume 4389, pp. 194–203. SPIE.