Fault diagnosis of robot joints using neural networks or the other AI approaches

By

Yun, Jinchao

MSc Robotics Dissertation

School of Engineering Mathematics and Technology
UNIVERSITY OF BRISTOL
&
School of Engineering
UNIVERSITY OF THE WEST OF ENGLAND

A MSc dissertation submitted to the University of Bristol and the University of the West of England in accordance with the requirements of the degree of MASTER OF SCIENCE IN ROBOTICS in the Faculty of Engineering.

August 28, 2025

Declaration of own work

I declare that the work in this MSc dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

Jinchao Yun. 23 July 2025

Ethics statement

This project did not require ethical review as determined by my supervisor Prof. Quanmin Zhu.

Jinchao Yun. 23 July 2025

Fault diagnosis of robot joints using neural networks or the other AI approaches

Jinchao Yun

Abstract— **Industrial manipulators operating in flexible production lines are prone to small joint deviations caused by wear, backlash, and dynamic disturbances, which can accumulate into significant positioning errors and productivity loss. This research develops a lightweight neural-network-based framework for accurate and real-time fault diagnosis of robot joints, with the UR10 robot serving as the case study. A complete methodology was implemented, including kinematic modeling, trajectory planning, systematic fault injection at varying magnitudes (0.1°, 0.5°, 1°), multifrequency sampling (50, 200, 800 Hz), feature fusion of end-effector position and orientation, and dataset construction. A multiclass backpropagation neural network (BPNN) was designed and trained on these data, then embedded into a MATLAB/Simulink simulation to realize a closed-loop online diagnostic system. Experimental results demonstrate that large static faults (1°) are detected with accuracies above 99%, while moderate sensitivity is achieved for subtle 0.1° faults, where accuracy increases from 58.0% at 50 Hz to 80.2% at 800 Hz. Under dynamic sinusoidal disturbances, the model maintains robust performance for low-amplitude oscillations but degrades under stronger, higher-frequency perturbations. Real-time simulations confirm that the diagnostic pipeline provides consistent and interpretable fault identification across all joints, supporting operator-facing monitoring. Overall, the study quantifies the trade-off between sampling frequency, diagnostic accuracy, and computational cost, and establishes that a fused-pose BPNN offers a practical, low-cost, and sensor-free solution for predictive maintenance and fault-tolerant control of industrial robots.**

I. INTRODUCTION

Industrial robots are core components of modern automated manufacturing and have transformed flexible production lines through their high efficiency, versatility and precision. Yet the continuous and often harsh operating conditions in which manipulators work inevitably lead to mechanical wear, backlash and thermal effects in the joints. Such degradation manifests as small angular offsets or gain errors that accumulate and compromise end–effector accuracy, reduce productivity and can cause costly downtime. Early fault diagnosis is therefore crucial for maintaining safety and efficiency.

Conventional fault diagnosis approaches rely on dedicated sensors, accelerometers or vision systems, to measure deviations between commanded and actual trajectories. Although straightforward to implement, these methods incur high hardware cost and offer limited insight into the exact fault source. Model–based and observer–based schemes, such as adaptive and sliding–mode observers, seek to estimate fault signals from state deviations but require accurate dynamic models and exhibit poor robustness to parameter uncertainties. These limitations motivate the exploration of data–driven techniques capable of learning complex nonlinear relationships without explicit system models.

In recent years, neural networks have emerged as powerful tools for robot fault diagnosis. Zhang and Zhu fused end–effector position and attitude information and trained a back-propagation (BP) network to map pose deviations to joint fault categories, achieving high accuracy even for subtle 0.1° offsets and highlighting the importance of feature fusion[1]. Hu et al. reported similar success on the UR10 manipulator, achieving over 99% simulated accuracy with a relatively simple BP architecture[2]. Convolutional neural networks (CNNs) extend this approach by automatically learning hierarchical features from raw sensor data. Pan, Qu and Peng demonstrated that a deep CNN could simultaneously diagnose sensor and actuator faults including gain, offset and malfunctions, and outperformed support vector machines and shallow networks in both accuracy and training time[3]. A deep residual network (DRNN) architecture further enhanced robustness by employing small convolution kernels and noise–injection training, leading to superior performance compared with traditional CNNs and long–term memory networks[4].

Beyond feed–forward architectures, researchers have investigated deep belief networks (DBN) combined with signal processing to extract richer temporal features. Jiao and Zheng applied wavelet transforms to denoise joint–bearing vibrations and fed normalised energy–entropy vectors into a DBN, achieving 97.96% accuracy and demonstrating real–time potential[5]. Hybrid models that integrate spatial and temporal processing are particularly appealing for predictive maintenance. Eang and Lee proposed a CNN–RNN observer that ingests temperature and speed measurements from motor drives and provides early fault detection with reduced computational complexity[6]. In the context of collaborative robots, Choi et al. converted vibration time series into images using recurrence plots, Gramian angular fields and spectrograms; they used generative adversarial networks to augment scarce fault data and found that DenseNet–based classifiers achieved the highest recognition accuracy[7]. Other studies employ wavelet scattering networks and wide neural networks to diagnose faults in Delta–type parallel robots, noting that feature selection via ANOVA enables detection of untrained faults[8], while cloud–based CNN frameworks transform sliding Fourier spectra into RGB images to enable remote fault diagnosis via industrial IoT[9]. A comprehensive review of manipulator faults underscores that machine–learning–driven methods and digital twins are becoming mainstream as they exploit multi-modal signals: voltage, cur-

rent, speed, torque and vibration to achieve higher diagnostic reliability[10].

Despite this progress, several challenges remain. Many studies focus on single features (e.g., end effector position), limiting sensitivity when faults are subtle. Others employ deep architectures that are computationally intensive and unsuitable for real-time applications. There is also limited systematic evaluation of how different feature combinations and sampling rates affect diagnosis accuracy. Furthermore, most work considers either sensor or actuator faults in isolation and lacks unified frameworks for concurrent fault detection.

Beyond single robot diagnosis, recent research has explored collaborative and data-limited scenarios. Qin and Wang developed a federated fault diagnosis method that encodes the local time-frequency data of each robot using the Gramian Angular Field and aggregates pre-trained local models across robots, their mutual diagnosis model, based on the consistency of group behavior, accurately identifies individual and system-wide faults in multi-robot systems[11]. Nandakumar et al. reviewed anomaly detection methods in autonomous robotic missions and proposed a unified classification of anomalies into spatial, temporal and spatiotemporal categories, emphasising that a common definition is essential for reliable detection across heterogeneous platforms[12]. For swarm robotics, O'Keeffe argued that predictive maintenance, anticipating degradation and acting before failures manifest, outperforms reactive approaches and requires autonomous fault detection, diagnosis and recovery to ensure long-term autonomy[13].

Efforts to reduce dependence on large labelled datasets have spurred the use of generative and semi–supervised models. Mitrevski and Plöger modified the sensor-based fault detection and diagnosis (SFDD) algorithm by using restricted Boltzmann machines as generative models to model sliding-window correlations, their framework achieved 88.6% precision and 75.6% recall on a mobile logistics robot and highlighted that model selection strongly influences diagnostic performance[14]. To overcome data scarcity and the sim-to-real gap, Mc Court et al. proposed a digital failure twin that simulates realistic failures and allows component-level fault diagnosis from system-level end-effector data, while noting that performance is sensitive to digital twin fidelity[15]. Chen et al. introduced a domain-adversarial neural network (DANN) framework that transfers knowledge from simulated to real data, improving a baseline CNN's accuracy from 70.00% to 80.22% on real robot tests[16]. Kumar's TemporalTwinNet embeds bidirectional LSTM layers into the digital twin pipeline to capture temporal dependencies and narrows the sim-to-real gap to 9.44%, demonstrating the importance of time-series modelling[17]. Liu et al. applied digital twin techniques to mechanical reducers and proposed a fault distribution GAN to map virtual and physical signals, the resulting system achieved accuracies up to 99.5%, surpassing CycleGAN and other baselines[18].

Alternative network paradigms are also emerging. Zuo et al. developed a multi-scale residual attention spiking neural network (MRA-SNN) for bearing fault diagnosis, which directly encodes vibration signals into spiking sequences and uses attention mechanisms to enhance representation. Experiments on benchmark datasets show improved accuracy and energy efficiency compared with conventional ANN-based methods[19]. Deng et al. introduced a semi–supervised Informer model that leverages abundant unlabeled robot monitoring data together with a small amount of labeled samples, by harnessing long-term dependencies, the Informer achieves accurate diagnosis of single and compound faults despite limited labeled data[20]. These advances indicate a shift towards federated, digital-twin and low-power approaches that address data scarcity, energy constraints and multi-robot cooperation.

To bridge these gaps, this research develops a lightweight yet effective neural-network-based framework for robot joint fault diagnosis. The following section outlines the aims of the research and the specific objectives that guide its implementation.

Research Aims

1) **Characterise fault propagation:** investigate how small joint deviations (0.1°-1°) propagate to end-effector pose errors by combining position and orientation measurements, and quantify the diagnostic benefit of fusing these features.
2) **Benchmark neural architectures:** assess the accuracy, robustness and computational requirements of a backpropagation (BP) neural network and compare its performance across different sampling frequencies and fault magnitudes. Where appropriate, contrast BP results with more advanced architectures reported in the literature, such as residual CNNs[4], federated learning models[11], and digital twin frameworks[16].
3) **Inform future research:** provide insights into feature fusion strategies, dataset design and real–time implementation that can inform the development of scalable fault diagnosis systems and predictive maintenance for industrial robots.

Project Objectives and Implementation Methods

1) **Kinematic modelling and trajectory planning:** derive the UR10 robot's forward kinematics using the Denavit-Hartenberg convention and generate baseline joint trajectories via linear interpolation between start and end configurations.
2) **Dataset construction:** systematically inject static and dynamic joint faults of varying magnitudes (0.1°, 0.5°, 1°) into each joint, sample end-effector pose data at multiple frequencies (50, 200, 800 Hz), and apply min–max normalisation to construct balanced training, validation and test sets.
3) **Neural network design and training:** implement a multiclass BP neural network with fused position-orientation inputs, tune hyperparameters such as learning rate and network depth, and explore the use of

simulated data generated by a digital twin to augment the training set.

4) **Real-time diagnosis and robustness evaluation:** embed the trained network into a MATLAB/Simulink environment to create a real-time diagnostic prototype, evaluate classification accuracy, precision, recall, cross-entropy loss and area under the ROC curve for both static and dynamic fault scenarios, and analyze the impact of sampling frequency on diagnostic performance.
5) **Comparative analysis:** compare the BP network results with alternative strategies such as generative-model-based fault detection[14] and semi-supervised learning[20] reported in the literature, highlighting advantages and limitations.
6) **Trade-off assessment:** quantify the trade-offs between dataset size, sampling frequency, computational cost and diagnostic accuracy, and draw recommendations for real-world deployment.

By achieving reliable and precise fault diagnosis with relatively simple neural network structures, this research contributes a practical solution that can be integrated into predictive maintenance and real-time monitoring frameworks for industrial robots in flexible manufacturing environments.

II. RESEARCH METHODOLOGY

A. Implementation

The proposed methodology for robot joint fault diagnosis is implemented in several stages: kinematic modelling of the manipulator, trajectory planning and fault injection, feature extraction and dataset construction, and the design and training of a neural network classifier. All implementations are conducted in MATLAB R2024b on a Windows 11 platform to ensure reproducibility.

1) Kinematic Modelling of UR10 Robot: The UR10 is a six-degree-of-freedom (6-DOF) industrial manipulator developed by Universal Robots. As shown in Figure 1. It consists of six revolute joints, offering a payload of 10 kg and a reach of 1300 mm, making it widely applicable in flexible production environments. The first three joints primarily determine the position of the end-effector, while the last three joints control the orientation of the wrist. Specifically:

- Joint 1 controls the base rotation in the horizontal plane.
- Joint 2 adjusts the vertical elevation of the arm.
- Joint 3 enables extension and retraction of the arm.
- Joint 4 bends or straightens the wrist.
- Joint 5 rotates the wrist laterally.
- Joint 6 rotates the end-effector about its own axis.

Forward kinematics is derived using the Denavit–Hartenberg (DH) convention. The homogeneous transformation from link $i-1$ to link i is given by:

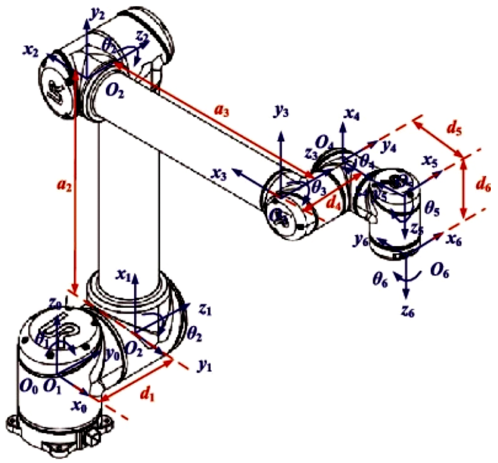


Fig. 1: Kinematic Modelling of UR10 robotic arm.

$$
\begin{aligned}
{}^{i-1}T_i &= \mathrm{Rot}_{z_{i-1}}(\theta_i)\cdot \mathrm{Trans}_{z_{i-1}}(d_i)\cdot \mathrm{Trans}_{x_i}(a_i)\cdot \mathrm{Rot}_{x_i}(\alpha_i) \\
&= \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{1}
$$

The overall transformation from the robot base to the end-effector is obtained by multiplying all link transformations:

$$
{}^0T_6 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 \cdot {}^4T_5 \cdot {}^5T_6 \tag{2}
$$

which can be expressed as:

$$
{}^0T_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix} \tag{3}
$$

Here, (p_x, p_y, p_z) denote the position of the end-effector, and the 3×3 rotation matrix $\mathbf{R}$ represent the orientation of the end-effector relative to the base coordinate system.

The corresponding DH parameters for the UR10 robot are summarised in Table I.

TABLE I: Denavit–Hartenberg Parameters of UR10

Joint i	a_i **(mm)**	α_i (°)	d_i **(mm)**	θ_i (°)
1	0	90	127.3	θ_1
2	−612.0	0	0	θ_2
3	−572.3	0	0	θ_3
4	0	90	163.9	θ_4
5	0	−90	115.7	θ_5
6	0	0	92.2	θ_6

From the transformation matrix, the position vector of the end-effector is obtained as:

$$
\mathbf{p} = [p_x, p_y, p_z]^T \tag{4}
$$

The orientation is extracted from the rotation matrix using Euler angles in Z–Y–X convention (yaw–pitch–roll):

$$\text{Roll } (\phi) = \arctan 2(r_{32},\ r_{33}) \tag{5}$$

$$\text{Pitch } (\theta) = \arctan 2\left(-r_{31},\ \sqrt{r_{11}^2 + r_{21}^2}\right) \tag{6}$$

$$\text{Yaw } (\psi) = \arctan 2(r_{21},\ r_{11}) \tag{7}$$

2) Trajectory Planning and Fault Injection: A baseline trajectory is generated by linear interpolation in joint space between predefined start and end configurations, as shown in Table II.

TABLE II: Start and End Joint Configurations of UR10

Joint	Start (°)	End (°)
1	0	30
2	−90	−120
3	90	130
4	0	−30
5	0	40
6	0	30

Each trajectory is executed for a fixed duration $T = 5$s. To generate sufficient data, each trajectory is uniformly sampled at a sampling rate $f \in \{50, 200, 800\}$Hz, leading to $N = T \cdot f = \{250, 1000, 4000\}$ samples per trajectory, as shown in Table III. This enables evaluation of the influence of dataset size on network training and diagnostic accuracy.

TABLE III: Sampling configurations for online simulation (with $T = 5$ s).

Sampling frequency (Hz)	Sampling interval (ms)	Samples per trajectory	Number of training sets
50	20	250	1750
200	5	1000	7000
800	1.25	4000	28000

To simulate faults, constant angular deviations are injected into individual joints:

$$\theta_i^{fault}(t) = \theta_i(t) + \Delta\theta \tag{8}$$

where $\Delta\theta \in \{0.1°, 0.5°, 1°\}$. Each joint is perturbed independently, producing six faulty trajectories and one fault-free trajectory, leading to seven classes of motion data.

3) Feature Extraction and Dataset Construction: For each sampled trajectory point, the end-effector **position** $[x, y, z]$ and **orientation** [roll, pitch, yaw] are extracted, forming a six-dimensional feature vector:

$$\mathbf{f} = [x, y, z, \text{roll}, \text{pitch}, \text{yaw}] \tag{9}$$

To avoid scale differences among features, min–max normalisation is applied:

$$f_{norm} = \frac{f - f_{\min}}{f_{\max} - f_{\min}} \tag{10}$$

The choice of dual feature fusion (combining both position and orientation) is motivated by the limitations of single-feature approaches. Position-only features cannot effectively capture faults in joints that mainly affect orientation, while orientation-only features show high correlation across different joints, leading to classification ambiguity. By fusing both sets of features, the neural network receives richer information, enabling more robust fault discrimination, especially for small fault magnitudes.

4) Neural Network Design and Training: A multi-class Backpropagation Neural Network (BPNN) is constructed to classify robot joint faults based on the extracted features. Its architecture is presented in Table IV.

TABLE IV: Neural Network Architecture

Layer	Neurons	Activation Function	Description
Input	6	–	Position (x,y,z) + Orientation (roll,pitch,yaw)
Hidden 1	13	Sigmoid	Non-linear mapping
Hidden 2	13	Sigmoid	Feature representation
Output	7	Softmax	6 faults + 1 normal

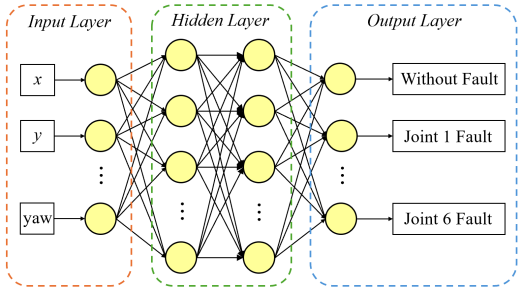


Fig. 2: Structure of the BP Neural Network.

The schematic structure of the BP neural network is shown in Figure 2, and its details are as follows.

Input Layer: The input layer receives the six-dimensional feature vector $[x, y, z, \text{roll}, \text{pitch}, \text{yaw}]$.

Hidden Layers: Two hidden layers are included, each containing 13 neurons. The sigmoid activation function is used:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{11}$$

This maps real-valued inputs into the range (0,1), introducing non-linearity and enabling the network to capture the complex relationship between input features and joint faults.

Output Layer: The output layer has seven neurons, corresponding to six joint faults and one fault-free case. The softmax activation function converts outputs into a probability distribution:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{12}$$

where $C = 7$ is the number of classes. This ensures that $\sum_{i=1}^{C} \hat{y}_i = 1$.

Loss Function: The model is trained by minimizing the cross-entropy loss:

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{13}$$

Cross-entropy is effective for classification because it penalises incorrect predictions with high confidence, forcing the model to increase the probability assigned to the correct class.

During training, both accuracy and loss curves are monitored to ensure convergence and prevent overfitting. The trained model is exported for integration into the Simulink real-time diagnostic system.

B. Experiment Methodology

To validate the effectiveness and robustness of the proposed joint fault diagnosis framework, a set of systematic experiments were conducted. This section focuses on how the experimental procedures were designed to test and evaluate the diagnostic performance of the system under different fault scenarios and noise conditions.

1) Simulink-Based Real-Time Fault Diagnosis Model: To validate the real-time performance of the proposed diagnostic system, an end-to-end simulation model was implemented in MATLAB/Simulink R2024b. The system replicates the closed-loop process of robotic joint command generation, fault injection, kinematic transformation, feature extraction, classification, and visualization.

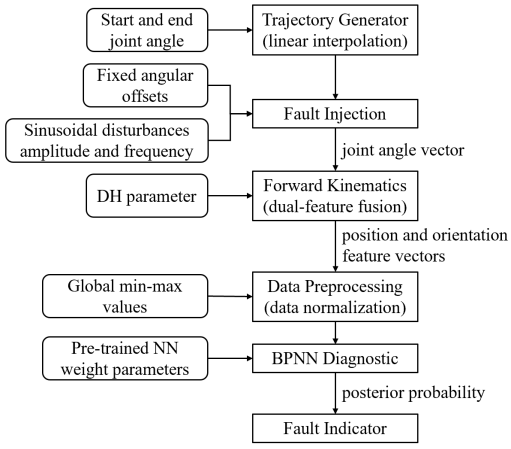


Fig. 3: Structure of the Simulink-based real-time fault diagnosis system, showing six subsystems and signal flows.

The structure of the model is shown in Figure 3, and consists of the following six core subsystems:

1) **Trajectory Generator**: This block produces time-varying joint commands based on joint space interpolation between known start and end poses as Table II. Each trajectory spans 5 seconds, with data sampled at 50/200/800 Hz to ensure compatibility with the neural network's training distribution.
2) **Fault Injection Module**: To simulate joint fault behavior, this module injects perturbations into normal joint signals. Fixed angular offsets (e.g. $\Delta\theta = 0.5°$) are used to simulate constant mechanical misalignment. Dynamic faults (e.g. sinusoidal disturbances) are used to simulate backlash and vibration.
   Faults can be injected into any joint or multiple joints simultaneously. Fault types and onset times are configurable via block parameters or signal triggers.
3) **Forward Kinematics Subsystem**: This subsystem computes the real-time end-effector pose based on the faulted joint angles using the same DH parameter model defined in Table I. The computation outputs the homogeneous transformation matrix ${}^0T_6$, from which the end-effector position and orientation are extracted. The orientation is converted into Euler angles using the Z–Y–X decomposition as defined in Equations 5–7. The result is a 6D pose vector $[x, y, z, \text{roll}, \text{pitch}, \text{yaw}]$ which is passed to the diagnostic model.
4) **Data Preprocessing Block**: This block collects the outputs of the forward kinematics subsystem and assembles them into a structured 6D feature vector. To ensure strict consistency between the training data and the online simulation, min–max scaling is applied in real time using the same global normalization parameters (f_{min}, f_{max}) computed from the entire training dataset. This guarantees that the neural network always receives input features within the same distributional range as during training, thereby preventing performance degradation caused by trajectory-specific scaling differences.
   The normalized pose vector is then formatted to match the input layer requirements of the neural network and sampled at each solver step.
5) **Neural Network Diagnostic Block**: The trained BP neural network is stored as a model function file for real-time fault diagnosis model, which generates a Simulink compatible subsystem from the exported MATLAB network. This allows the neural network to be directly embedded in the real-time simulation model.
   The 6D input feature vector is processed through the network, and the output is a 7-element probability vector indicating the predicted fault class (six joints + normal). Sigmoid activations are applied in the hidden layers to introduce nonlinearity, and a softmax activation is used in the output layer to convert raw scores into class probabilities.
6) **Fault Indicator System**: The final diagnostic decision is sent to a visual output system that includes a set of LED-style indicators to visualize which joint is currently predicted to be faulty or a numeric display to show the predicted class index.

Each simulation loop processes a new pose sample at every step, allowing the network to make continuous real-time decisions about system state.

2) Random Error Injection for Robustness Evaluation: While previous studies evaluate neural network performance using static fault conditions (i.e., fixed angular offsets), real-

world systems often exhibit time-varying disturbances due to backlash, vibration, or compliance. To assess the robustness of the trained BPNN under such scenarios, a new type of fault input is designed using a combined model:

$$\theta_i^{fault}(t) = \theta_i(t) + \Delta\theta + A\sin(\omega t) \tag{14}$$

where $\Delta\theta$ is the fixed offset angle (0.5° in this study), A is the amplitude of the sinusoidal disturbance (e.g., 0.2°), and ω is the angular frequency of the disturbance (1-5 Hz range)

This model simulates faults with both static and dynamic components. Datasets are generated using the same kinematic model, and the trained BPNN (without retraining) is tested on these new samples to evaluate generalization and robustness.

3) Evaluation Metrics: To comprehensively assess the model's diagnostic performance, several standard classification metrics were employed:

Accuracy The proportion of correctly classified instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

where TP = true positives, TN = true negatives, FP = false positives, FN = false negatives.

Precision Measures the correctness among predicted positives:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{16}$$

Recall Measures the ability to detect actual positives:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{17}$$

Cross-Entropy Loss Used as the training objective for the BP neural network. Given predicted probabilities $\hat{y}_i$ and one-hot encoded targets y_i, the binary cross-entropy loss is:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{18}$$

where C is the number of classes.

Confusion Matrix A $C \times C$ matrix showing predicted vs. actual classes. It visualizes misclassifications, e.g., Joint 2 fault being misclassified as Joint 3.

ROC Curve and AUC Receiver Operating Characteristic (ROC) curve plots the true positive rate vs. false positive rate. Area Under Curve (AUC) close to 1 indicates excellent separability.

4) Experimental Scenarios: Three experimental modes were configured:

1) **Single-joint static fault**: A constant offset is applied to a single joint (e.g., Joint 3: +0.5°).
2) **Dynamic sinusoidal fault**: A time-varying disturbance added to simulate transient performance drift.

III. RESULTS

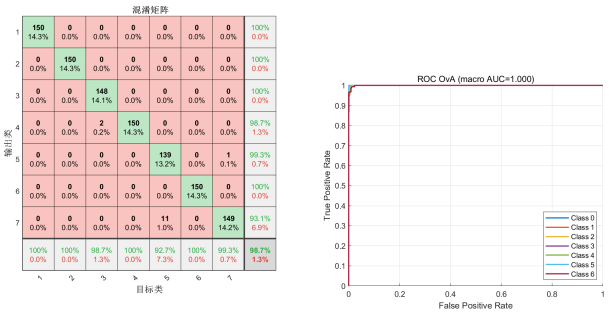
A. *Effect of Sampling Frequency on Diagnostic Accuracy*

Three sampling frequencies (50 Hz, 200 Hz, 800 Hz) were evaluated with joint angle errors of 1 °, 0.5 ° and 0.1 °. Table V reports the quantitative outcomes in Accuracy, Cross-Entropy, Precision, Recall, and AUC.

TABLE V: Performance versus sampling frequency

Joint Error	Frequency (Hz)	Accuracy (%)	Cross-Entropy	Precision (%)	Recall (%)	AUC (%)
1°	50	99.14	0.02105	99.18	99.14	99.98
	200	99.61	0.00940	99.62	99.61	99.99
	800	99.95	0.01308	99.95	99.95	99.99
0.5°	50	95.05	0.05589	95.41	95.05	99.64
	200	97.48	0.03784	97.51	97.48	99.94
	800	98.48	0.03202	98.56	98.48	99.99
0.1°	50	58.02	0.17338	55.61	58.02	87.72
	200	67.33	0.13902	67.24	67.33	92.23
	800	80.24	0.14164	81.15	80.24	96.20

The result shows that for large faults (e.g., 1°), the diagnostic performance is uniformly high (over 99 % accuracy across all frequencies), raising the frequency yields only marginal gains while the training time grows markedly at 800 Hz. For small faults (0.5° and especially 0.1°), there is a significant decline in diagnostic performance. Moderate increases in frequency improve separability, accuracy at 0.1° rises from 58.02% at 50 Hz to 80.24% at 800 Hz, indicating that denser sampling improves the separability of subtle deviations. But it should be noted that high frequencies enlarge the dataset size, which increases training time and reduces suitability for scenarios with strict real-time requirements.



(a) Confusion matrix (b) ROC curve

Fig. 4: Confusion matrix and ROC curve at 0.5° joint error and 200 Hz sampling rate.

To further illustrate the diagnostic capability, the confusion matrix and ROC curve at 0.5° joint error with a sampling frequency of 200 Hz are presented in Figures 4a and 4b, respectively.

The confusion matrix reveals excellent classification overall, but with notable overlap between Class 4 (Joint 3, elbow extension) and Class 7 (Joint 6, wrist rotation). This confusion arises because both joints influence the end-effector orientation: J3 contributes to extension but, along the tested path, its translational effect is largely compensated by upstream joints, leaving an orientation signature similar to

that of J6, which rotates the wrist about its axis. As a result, the network occasionally misclassifies one as the other.

Figure 4b presents the ROC curves for all seven classes. The curves are tightly clustered along the top-left boundary, and the macro-average AUC reached 100 %, indicating an excellent balance between true positive and false positive rates across all classes.

In summary, the experiments confirm that sampling frequency has a significant impact on diagnostic performance, especially for small-magnitude joint errors. A frequency of 200 Hz already provides highly reliable results, while 800 Hz offers the best accuracy and robustness by investing more training time.

B. Robustness under Dynamic Disturbances

To further assess the robustness of the diagnostic framework, experiments were conducted by fixing the sampling frequency at 200 Hz and injecting a constant joint error of 0.5° as the baseline condition. On top of this baseline, sinusoidal disturbances of varying amplitude ($A = 0.1°$ and $0.3°$) and frequency ($f = 1$, 3, and 10 Hz) were superimposed. Table VI presents the results in terms of accuracy, cross-entropy, precision, recall, and AUC.

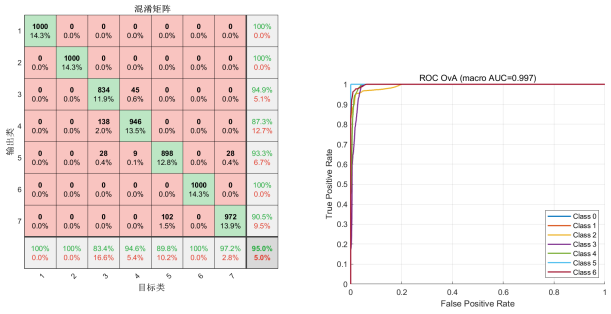
TABLE VI: Diagnostic performance under dynamic disturbances (200 Hz sampling, 0.5° baseline error).

Condition	Amplitude (°)	Accuracy (%)	Cross-Entropy	Precision (%)	Recall (%)	AUC (%)
Baseline (no disturbance)	0	97.96	0.3099	98.07	97.96	99.83
$f = 1$ Hz	0.1	94.57	0.3292	94.59	94.57	99.65
	0.3	77.23	0.5970	79.06	77.23	96.71
$f = 3$ Hz	0.1	95.00	0.3137	95.13	95.00	99.67
	0.3	71.81	0.7644	75.52	71.81	95.28
$f = 10$ Hz	0.1	90.47	0.4377	90.45	90.47	99.18
	0.3	67.86	0.8037	69.56	67.86	94.37

The baseline case with only the 0.5° constant error achieved an accuracy of 97.96% and an AUC of 99.83%, confirming the high reliability of the model without dynamic perturbations. When a low-amplitude disturbance ($A = 0.1°$) was introduced, the network maintained strong performance across all tested frequencies: accuracies remained above 90%, and AUC values stayed higher than 99%. For instance, at $f = 3$ Hz and $A = 0.1°$, the model achieved 95.00% accuracy and 99.67% AUC, indicating that small oscillatory noise has only a minor effect on detection performance.

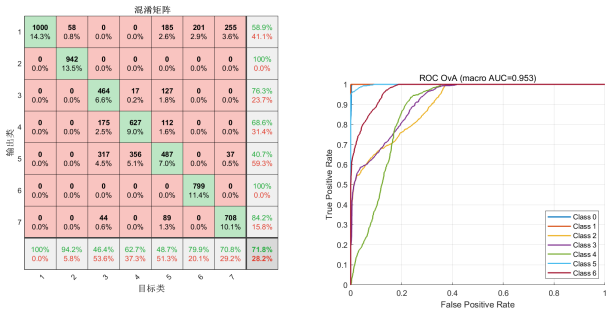
However, larger disturbances ($A = 0.3°$) caused more pronounced degradation, particularly as frequency increased. At $f = 1$ Hz, the accuracy dropped to 77.23% with an AUC of 96.71%. At higher frequencies, the effect became stronger: for example, at $f = 10$ Hz with $A = 0.3°$, the accuracy declined to 67.86% and the AUC decreased to 94.37%. This pattern shows that stronger and faster oscillations introduce disturbance components that overlap significantly with the fault signatures, making it more difficult for the classifier to distinguish between different joint faults.

A focused comparison was conducted at a fixed disturbance frequency of $f = 3$ Hz for two amplitudes ($A = 0.1°$ and $A = 0.3°$), under the same baseline setting (200 Hz sampling and a constant $0.5°$ joint error). As shown in Figure 5a and 5b. With a low-amplitude disturbance of $A = 0.1°$, the classifier retained strong separability: the one-vs-all ROC yielded a macro-average AUC of 99.7%, and the confusion matrix indicated high recalls across classes (e.g., 100% for Classes 1, 2, and 6), with only minor leakage among Classes 3–5. The overall accuracy remained high at 95.0%, confirming that small oscillations introduce only a limited reduction in discriminative performance.



(a) Confusion matrix (b) ROC curve

Fig. 5: Confusion matrix and ROC curve at A = 0.1°, f = 3 Hz sinusoidal disturbance.



(a) Confusion matrix (b) ROC curve

Fig. 6: Confusion matrix and ROC curve at A = 0.3°, f = 3 Hz sinusoidal disturbance.

As shown in Figure 6a & 6b. When the amplitude increased to $A = 0.3°$, separability degraded markedly. The macro-average AUC dropped to 95.3%, and misclassifications concentrated among Classes 3–5, where the recalls declined to 46.4% (Class 3), 62.7% (Class 4), and 48.7% (Class 5). Although Class 1 preserved a 100% recall, its precision decreased to 58.9% due to samples from other classes being incorrectly assigned to Class 1. The overall accuracy fell to 71.8%. These results indicate that larger oscillations amplify the overlap of residual features—particularly for mid-arm joints exhibiting coupled effects on reach and orientation—thereby reducing class separability.

In summary, the results demonstrate that the proposed model is robust to small-amplitude dynamic disturbances and retains high discriminative ability in these cases. Nevertheless, when both amplitude and frequency are large, performance degradation becomes evident. These findings

highlight the importance of considering disturbance intensity and frequency in practical applications, where industrial robots may be subject to external vibrations and dynamic noise.

C. Real-Time Fault Detection in Simulink

A real-time simulation model was implemented in Simulink to validate the on-line diagnostic workflow during operation (Figure 7). The model exposes all experiment controls at run time, including the target joint ID for fault injection, the magnitude of the constant offset, an enable flag for dynamic perturbations, and the amplitude–frequency pair (A, f) of a sinusoidal disturbance. Given user-specified start/end joint configurations and a motion duration, the *Trajectory Generator* produces a joint-space profile that is passed to the *Fault Injection* module, where the commanded joint angles are modified according to the selected scenario. The perturbed trajectory is mapped to the end-effector pose by the *Forward Kinematics* block, from which a six-dimensional feature vector (position and attitude) is extracted.

To ensure consistency with off-line training, the *Data Preprocessing* block applies min–max normalization using pre-computed bounds. The normalized features are streamed to the *Neural Net Diagnostic* block, which loads the pre-trained BP network parameters and outputs posterior probabilities over seven classes (one no-fault plus six joint-fault classes). Three scopes provide live observability: (i) joint-angle trajectories for each run, (ii) the time history of the 6D normalized features, and (iii) the seven posterior probabilities. A display block reports the predicted class ID in real time (0 = no fault; 1-6 = Joint1-Joint6 fault).

For online visualization, Scope-3 plots the seven posterior probabilities over the 5 seconds trajectory. We illustrate typical behavior with Joint 3 (class label 4 in the signals). Under a 0.5° static fault only, the posterior for class 4 remains dominant and stable at about 90 % throughout, with a brief dip to roughly 70 % near 0.3 s; all competing classes stay close to zero (Figure 8). This indicates a confident and temporally consistent identification when no dynamic disturbance is present.

Superimposing a low-amplitude sinusoid (A = 0.1°, f = 1 Hz) introduces a phase-locked modulation of the posteriors (Figure 9): the class-4 probability oscillates but never drops below 65 % and continuously exceeds the other classes, so no false detections occur. With a larger disturbance (A = 0.3°, f = 1 Hz), the posteriors exhibit pronounced swings (Figure 10). At several instants (e.g., 0.75 s and 1.75 s) class 4 is temporarily overtaken by other classes, yielding short misclassification windows before rapidly reverting to the correct label. This time-domain pattern matches the offline results for large-amplitude sinusoidal disturbance, where increased oscillation reduces feature separability among kinematically coupled joints.

Overall, the real-time traces confirm that the proposed Simulink pipeline provides accurate and practically robust fault identification across joints. The system maintains high confidence under static faults and modest dynamic perturbations, and it tolerates disturbances arising from mechanical backlash, structural compliance, or vibration. Even when stronger disturbances cause transient probability inversions, the classifier recovers promptly, and the scopes offer immediate, interpretable feedback suitable for on-line monitoring and diagnosis.

IV. DISCUSSION AND CONCLUSIONS

A. Discussion

This study focused on the UR10 robotic arm and aimed at locating joint faults using a lightweight BPNN based on the end-effector pose features under real-time constraints. It achieved an integrated implementation from the construction of fault data sets, feature extraction (position + Euler angles) and normalization, to the training of BPNN and the online diagnostic closed-loop in Simulink. The experimental programme covered systematic static offsets (1°, 0.5°, 0.1°) at a sampling rate of 50/200/800 Hz, complemented by sinusoidal dynamic disturbances superimposed on a 0.5° baseline. Quantitative results show that the approach achieved uniformly high accuracy for large static faults (over 99% for 1° across all frequencies), strong performance for 0.5° (95.05–98.48%), and meaningful sensitivity at 0.1° with accuracy rising from 58.02% at 50 Hz to 80.24% at 800 Hz. This demonstrates a trend where the ability to distinguish minor deviations increases as the sample density increases. From the combined evaluation of accuracy, cross-entropy, precision, recall rate, and AUC, etc., it can be seen that the trade-off between separability and data/compute cost, as 200 Hz delivering a practical balance and 800 Hz offering additional gains for small-magnitude faults at higher training cost. This trend is consistent with the performance of the confusion matrix and ROC curve (for example the macro average AUC is close to 100% at 0.5° at 200 Hz), indicating that the classification boundary has good separability and calibration.

The dynamic disturbance experiment further revealed the stability of the system under small, low-frequency vibrations and the degradation mechanism under large, high-frequency vibrations. With a 0.5° baseline, low-amplitude disturbances ($A = 0.1^\circ$) maintain at least 90% accuracy across 1/3/10 Hz with AUC values above 99%, whereas higher amplitude ($A = 0.3^\circ$) causes accuracy to fall to 77.23% at 1 Hz, 71.81% at 3 Hz, and 67.86% at 10 Hz. This pattern is consistent with disturbance components encroaching on the fault manifolds in feature space as both amplitude and frequency increase, thereby reducing class separability. The confusion matrix and ROC curves at 3 Hz illustrate this contrast explicitly. These observations are important for practical deployment where vibration, compliance, or backlash may introduce time-varying effects.

The Simulink real-time model validates the end-to-end workflow: the data path (trajectory – fault injection – forward kinematics – normalisation – BPNN – posterior probabilities) and the scopes provide live, interpretable posterior probabili-

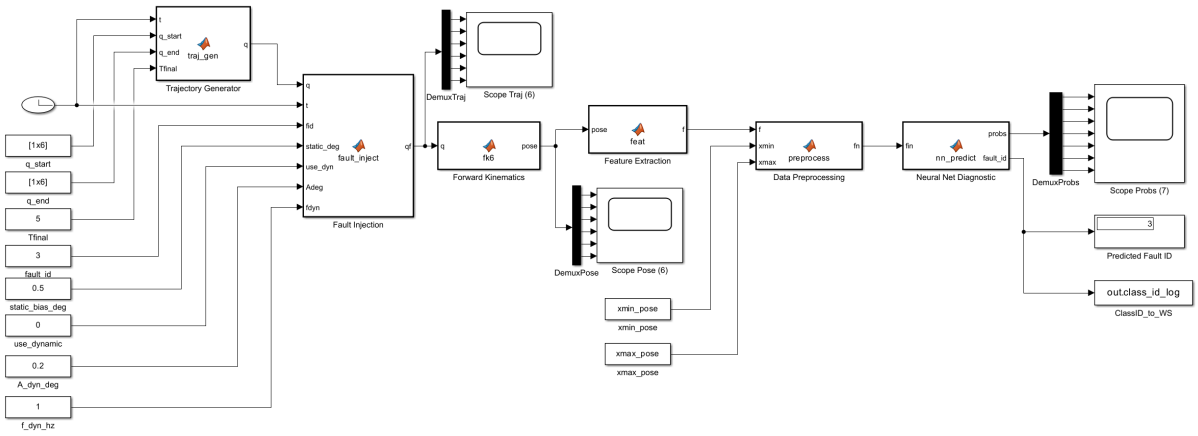


Fig. 7: Simulink real-time robot joint fault diagnostic model.

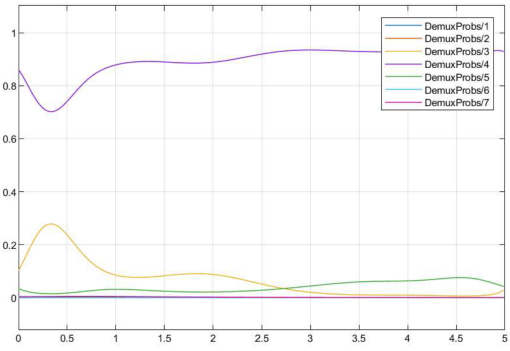


Fig. 8: Seven posterior probabilities at joint 3 with 0.5° static fault only

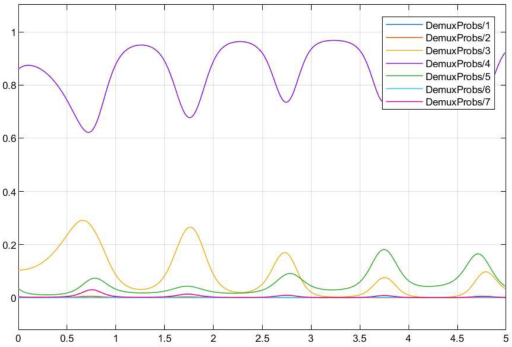


Fig. 9: Seven posterior probabilities at joint 3 with 0.5° static fault and low-amplitude sinusoid (A = 0.1°, f = 1 Hz) disturbance

ties and diagnostics that align with offline behaviour, offering a clear route to online monitoring and operator feedback.

B. Limitations

Evaluation is predominantly simulation-based using forward kinematics; effects from sensor noise, calibration drift, latency and quantisation in a physical stack were not comprehensively exercised. Results rely on a single start–end path, limiting workspace and attitude diversity and contributing

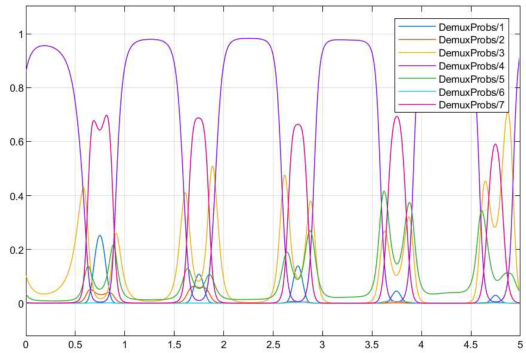


Fig. 10: Seven posterior probabilities at joint 3 with 0.5° static fault and large-amplitude sinusoid (A = 0.1°, f = 1 Hz) disturbance

to Joint 3–Joint 6 confusion where orientation cues outweigh path-specific translational differences. Orientation is encoded via Euler angles (Z–Y–X), which may be locally ill-conditioned near singular attitudes. The classifier is feedforward and does not explicitly model temporal dependencies, leaving headroom under strong or fast disturbances. Experiments prioritised single-joint faults, concurrent multi-joint faults were not systematically benchmarked.

C. Future Work

Workspace-wide validation should introduce multi-segment and curved paths (splines or arcs) to diversify wrist postures and approach vectors, and to reassess confusion patterns while quantifying the 200 Hz versus 800 Hz cost/benefit at scale. Orientation should be re-parameterised using unit quaternions or rotation vectors to improve numerical smoothness and avoid singularities. Lightweight temporal modelling (for example, a shallow GRU or TCN layer with the same inputs and computational budget) could stabilise decisions in regimes where $A = 0.3°$ and $f \geq 3$–10 Hz degrade separability. Multi-fault recognition should be reformulated as a multi-label problem and evaluated with reproducible precision–recall benchmarks. Noise and domain

robustness should be enhanced via sensor noise injection, bias drift, and domain randomisation, normalisation schemes such as sliding-window standardisation could be compared against global min–max under distribution shift. Finally, hardware-in-the-loop experiments should port the Simulink model to real-time targets to quantify end-to-end latency, jitter, and throughput alongside accuracy, linking algorithmic performance to system-level metrics.

D. Conclusions

The study demonstrates that a fused-pose, lightweight BPNN can localise joint faults on a UR10 with high accuracy under static conditions and robust performance under small dynamic disturbances, while exposing clear limits under larger and faster oscillations. The frequency–accuracy–cost trade-off is quantified, with 200 Hz providing a practical balance and 800 Hz yielding improved sensitivity to 0.1° faults at increased computational expense. The real-time Simulink model confirms online feasibility and provides interpretable, operator-facing feedback. For applications, the proposed process does not rely on additional sensors or complex models and has good feasibility for integration and deployment. For research, this paper clearly identifies the weak links that need to be prioritized for breakthrough (pose representation, temporal modeling, concurrent faults and domain shift robustness), and based on this, plans out feasible improvement routes. These conclusions lay a solid foundation for promoting a deployable, scalable, and robust robot joint fault diagnosis system that remains stable under more complex disturbance scenarios.

APPENDIX

Appendix 1. Publication List Jinchao Yun & Quanmin Zhu, "Neural Natwork enhanced fault detection on robot arms." to be submitted for publication.

Appendix 2. Acknowledgement I would like to express my deepest gratitude to my supervisor, Professor Quanmin Zhu, for his invaluable guidance, encouragement, and insightful advice throughout the course of this research. His expertise and support have been fundamental to the completion of this dissertation. I am also sincerely grateful to the University of Bristol and the University of the West of England for providing the MSc Robotics programme, through which I gained the knowledge and skills in robotics that enabled this work. My heartfelt thanks extend to my classmates and friends, whose help and companionship during my studies and experiments made this journey both productive and enjoyable. I am profoundly indebted to my parents for their unwavering support and understanding, which has been a constant source of strength. Finally, I wish to acknowledge my own persistence and dedication, without which the pursuit and completion of this master's degree would not have been possible.

Appendix 3. Experimental supporting materials The experimental code and project files are attached together at `https://github.com/JinchaoYun/EMATM0055_Dissertation.git` for evaluation or reproduction.

REFERENCES

[1] Y. Zhang and Q. Zhu, "Neural network-enhanced fault diagnosis of robot joints," *Algorithms*, vol. 16, no. 10, p. 489, 2023.
[2] M. Hu, J. Peng, Y. Zhang, and Q. Zhu, "Fault diagnosis of robot joint based on bp neural network," *Robotica*, vol. 40, no. 12, pp. 2400–2416, 2022.
[3] J. Pan, L. Qu, and K. Peng, "Sensor and actuator fault diagnosis for robot joint based on deep cnn," *Entropy*, vol. 23, no. 6, p. 751, 2021.
[4] ——, "Deep residual neural–network–based robot joint fault diagnosis method," *Scientific Reports*, vol. 12, no. 1, p. 17158, 2022.
[5] J. Jiao and X.-J. Zheng, "Fault diagnosis method for industrial robots based on dbn joint information fusion technology," *Computational Intelligence and Neuroscience*, vol. 2022, p. 4340817, 2022.
[6] C. Eang and S. Lee, "Predictive maintenance and fault detection for motor drive control systems in industrial robots using cnn–rnn–based observers," *Sensors*, vol. 25, no. 1, p. 25, 2024.
[7] S.-H. Choi, J. Park, T.-K. Kim, and B. Jung, "Fault diagnosis method for human coexistence robots based on cnn using time–series data generation and image encoding," *Applied Sciences*, vol. 13, no. 10, p. 5792, 2023.
[8] C. Urrea and C. Domínguez Acosta, "Fault diagnosis in a four–arm delta robot based on wavelet scattering networks and artificial intelligence techniques," *Machines*, vol. 12, no. 3, p. 380, 2024.
[9] D. Łuczak, P. Hanczakowski, W. Kania, M. Brockmeyer, and Z. Kulesza, "Cloud–based fault diagnosis by convolutional neural network as time–frequency rgb image recognition of industrial machine vibration with iot connectivity," *Sensors*, vol. 23, no. 7, p. 3755, 2023.
[10] X. Liu, Y. Liang, and Y. Chen, "Fault types and diagnostic methods of manipulator robots: A review," *Machines*, vol. 11, no. 9, p. 844, 2023.
[11] Y. Qin and O. Wang, "Federated fault diagnosis method for collaborative self–diagnosis and cross–robot peer diagnosis," *PLOS ONE*, vol. 20, no. 7, p. e0322484, 2025.
[12] S. C. Nandakumar, D. Mitchell, M. S. Erden, D. Flynn, and T. Lim, "Anomaly detection methods in autonomous robotic missions," *Sensors*, vol. 24, no. 4, p. 1330, 2024.
[13] J. O'Keeffe, "Anticipating degradation: A predictive approach to fault tolerance in robot swarms," *arXiv preprint arXiv:2504.01594*, 2025.
[14] A. Mitrevski and P. G. Plöger, "Data-driven robot fault detection and diagnosis using generative models: A modified sfdd algorithm," *arXiv preprint arXiv:2311.13866*, 2023.
[15] K. Mc Court, X. Mc Court, S. Du, and Z. Zeng, "Use digital twins to support fault diagnosis from system-level condition-monitoring data," *arXiv preprint arXiv:2411.01360*, 2024.
[16] Z. Chen, H. Fu, and Z. Zeng, "A domain adaptation neural network for digital twin-supported fault diagnosis," *arXiv preprint arXiv:2505.21046*, 2025.
[17] P. Kumar, "Temporal modeling for domain-invariant fault diagnosis in robotics digital twin systems," *arXiv preprint*, 2025.
[18] W. Liu, B. Han, A. Zheng, Z. Zheng, S. Chen, and S. Jia, "Fault diagnosis of reducers based on digital twins and deep learning," *Scientific Reports*, vol. 14, p. 24406, 2024.
[19] L. Zuo, Y. Ding, M. Jing, K. Yang, B. Chen, and Y. Yu, "Toward end-to-end bearing fault diagnosis for industrial scenarios with spiking neural networks," *arXiv preprint arXiv:2408.11067*, 2024.
[20] C. Deng, J. Song, C. Chen, T. Wang, and L. Cheng, "Semi-supervised informer for the compound fault diagnosis of industrial robots," *Sensors*, vol. 24, no. 12, p. 3732, 2024.