Fault Diagnosis of Industrial Robots Using a Digital Twin and GRU-Based Deep Learning

Ilhem Ben Hnaien[1][a], Eric Gascard[2][b], Zineb Simeu-Abazi[2][c] and Hedi Dhouibi[3]
[1]*NOCCS Laboratory, Natl. Engr. School of Sousse, University of Sousse, Sousse, Tunisia*
[2]*Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, Grenoble, France*
[3]*LARATSI Laboratory, High. Inst. of Appl. Sci. Technol. of Kairouan, University of Kairouan, Kairouan, Tunisia*

Keywords: Digital Twin, Fault Diagnosis, GRU Neural Network, Industrial Robot, Time-Series Classification.

Abstract: This paper proposes a fault diagnosis method for industrial robots based on the combination of a digital twin and a GRU-based deep learning model. A high-fidelity digital replica of the 6-DOF Stäubli TX60 robot was developed using MATLAB Simulink and Simscape Multibody to simulate both normal and faulty behaviors. A dedicated fault injection module was used to generate motor blockage scenarios at different time instants, creating a labeled dataset of 49 classes. The time-series data were then used to train a Gated Recurrent Unit (GRU) neural network, which is efficient for modeling temporal patterns. The trained model achieved an accuracy of 87.35%, with strong performance across different fault types. This approach enables reliable, non-invasive, and repeatable fault diagnosis and provides a solid foundation for future work on predictive maintenance and deployment on real robotic platforms.

1 INTRODUCTION

The increasing integration of industrial robots in manufacturing systems has raised new challenges in terms of reliability, safety, and maintenance. Joint-level failures, particularly actuator blockages, are among the most common and disruptive fault types, often leading to performance degradation or production downtime (Liu et al., 2021), (She et al., 2022). As production lines become more automated, there is a pressing need for intelligent diagnostic systems capable of detecting, isolating, and identifying such faults in real time, while minimizing the reliance on additional intrusive sensors or manual intervention. Our methodology can be seamlessly integrated with existing sensory feedback to enhance mission-critical decision-making and predictive maintenance capabilities (Yin and Kaynak, 2015), (Lee et al., 2015), (Sabry and Amirulddin, 2024). In this context, the digital twin paradigm has emerged as a powerful framework for predictive maintenance and fault diagnosis. A digital twin is a virtual replica of a physical system that enables real-time simulation and monitoring of system behavior under varying operating conditions (Grieves, 2022). Combined with artificial intelligence (AI), it facilitates the generation of realistic training data and enables non-destructive experimentation—two critical aspects for the development of robust diagnostic models.

Related Work. Previous studies have explored data-driven fault diagnosis in robotic systems using traditional classifiers, support vector machines (SVM), or statistical models. For example, Liu et al. (Liu et al., 2021) proposed sensorless force estimation using disturbance observers, while Sabry and Amirulddin (Sabry and Amirulddin, 2024) provided a comprehensive review of robot fault detection techniques. Our contribution differs by integrating a physics-based digital twin with temporal deep learning models, enabling non-invasive and repeatable diagnosis under complex dynamic conditions. Compared to conventional thresholding or signal modeling approaches, our method leverages simulation-driven data generation and time-series classification, providing higher scalability and adaptability to future robotic platforms.

The main objective of this study is to propose an integrated methodology that leverages a high-

[a] https://orcid.org/0009-0003-9972-2213
[b] https://orcid.org/0000-0003-4332-0752
[c] https://orcid.org/0000-0002-1660-3960

Ben Hnaien, I., Gascard, E., Simeu-Abazi, Z. and Dhouibi, H.
Fault Diagnosis of Industrial Robots Using a Digital Twin and GRU-Based Deep Learning.
DOI: 10.5220/0013705600003982

In *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2025) - Volume 2*, pages 487-494
ISBN: 978-989-758-770-2; ISSN: 2184-2809

fidelity digital twin and a deep learning model to diagnose actuator-level faults in a 6-DOF industrial robot (Stäubli TX60). While several works have explored the use of recurrent neural networks (RNNs) for time-series classification, we focus here on Gated Recurrent Units (GRU), a variant of RNNs known for its computational efficiency and ability to capture long-term dependencies in sequential data (Cho et al., 2014). GRU are particularly suitable for robotic fault diagnosis, where the temporal evolution of joint trajectories provides essential clues for identifying failure types and their onset times. The proposed contribution consists in (i) developing a comprehensive digital twin in MATLAB Simulink and Simscape Multibody, capable of replicating both nominal and faulty dynamics of the TX60 robot; (ii) implementing a structured fault injection module that introduces random motor blockages at selected instants (e.g., 0.1 s, 0.4 s, 0.5 s); (iii) generating a labeled dataset of time-series responses; and (iv) training a GRU-based neural network classifier in Python to distinguish between 49 different scenarios (48 faults + 1 normal state). Our methodology is illustrated in Figure 1, which summarizes the full pipeline—from fault simulation to neural classification. The real robot serves as a validation reference, while the digital twin produces repeatable scenarios for training and evaluation. Fault signatures extracted from joint trajectories are used to teach the model how to recognize distinct fault types and their corresponding time profiles. The remainder of this paper is structured as follows. Section 2 details the proposed methodology, including the digital twin setup, data generation strategy, and GRU-based classification approach. Section 3 presents the digital twin architecture and fault simulation procedures. Section 4 describes the data preparation pipeline and neural network training. Section 5 discusses the model's performance through accuracy plots, confusion matrix, and detailed classification metrics. Finally, conclusions and future directions are outlined in Section 6.

2 METHODOLOGY

The proposed methodology, illustrated in Figure 1, combines a physics-based digital twin with a data-driven deep learning classifier to enable accurate fault diagnosis on the Stäubli TX60 robotic arm. The real system serves as a reference for validating the behavior of a high-fidelity virtual replica developed in MATLAB Simulink and Simscape Multibody.

This digital twin replicates both nominal and faulty conditions and includes a fault injection module that simulates motor blockage at predefined instants (e.g., 0.1 s, 0.4 s, 0.5 s) to generate diverse and controlled fault scenarios. These simulations produce time series representing joint angles and end-effector positions, which are then collected and segmented into sequences of 10 time steps. The resulting labeled dataset, composed of 49 distinct classes (48 fault scenarios and one nominal state), is used to train a Gated Recurrent Unit (GRU) neural network. GRU is a variant of recurrent neural networks optimized for temporal pattern recognition, offering a simplified architecture compared to LSTMs while maintaining performance in sequence modeling (Cho et al., 2014). The model is trained in Python over 100 epochs using the Adam optimizer and categorical cross-entropy loss. Stratified splitting ensures class balance across training and test sets. This integrated simulation-learning framework provides a reproducible and non-destructive approach for diagnosing robotic joint faults under dynamic conditions.

3 DIGITAL TWIN AND FAULT SIMULATION

The concept of the digital twin plays a central role in modern fault detection strategies. It enables real-time simulation, fault injection, and behavioral analysis without relying on physical hardware. In this work, a digital twin of the Stäubli TX60 industrial robot was developed to replicate both its nominal and faulty operations. This twin was used to generate representative data under different scenarios for training and evaluating a neural network-based diagnoser.

3.1 Digital Model of the Robotic System

To accurately represent the behavior of the Stäubli TX60 robot, a high-fidelity digital twin was developed using MATLAB Simulink and Simscape Multibody (Boschetti and Sinico, 2024). This digital model reproduces the kinematic and dynamic characteristics of the robot under nominal conditions and serves as a reference for fault diagnosis. The CAD structure of the robot was imported in STEP format and integrated into the Simscape environment via the Simulink interface. The kinematic chain was reconstructed, and a dynamic model was built using polynomial trajectory generation, inverse and forward kinematics, and joint actuation. Each joint is actuated by a dedicated motor, and the control architecture relies on feedback loops to track desired trajectories. The model simulates the 3D motion of the end-effector as it follows predefined paths, such as square trajectories in Cartesian

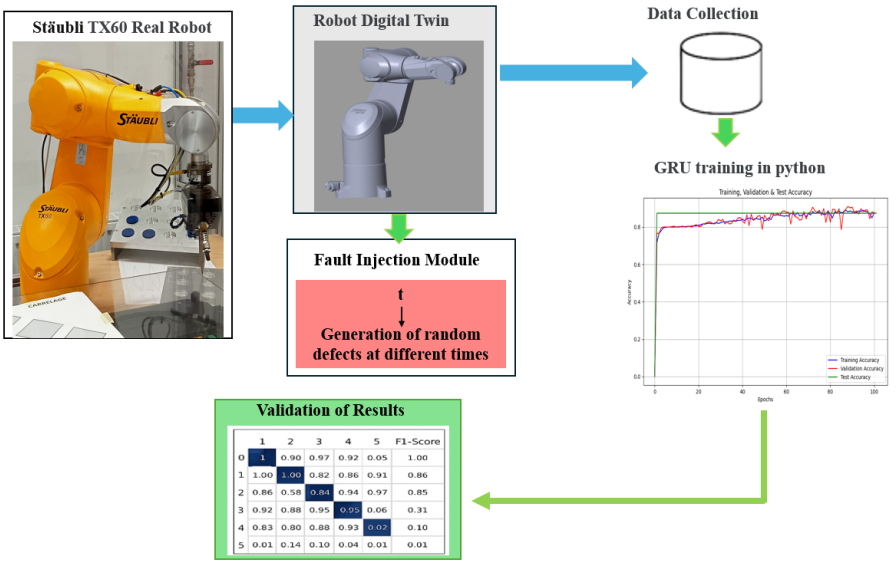

Figure 1: Overview of the Developed Digital Twin Architecture for Fault Diagnosis of the Stäubli TX60 Robot.

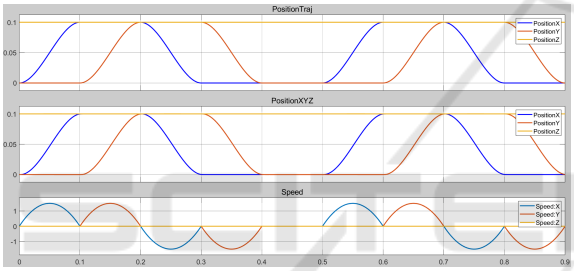

Figure 2: Cartesian End-Effector Trajectories: Two Square Paths Executed by the Robot.

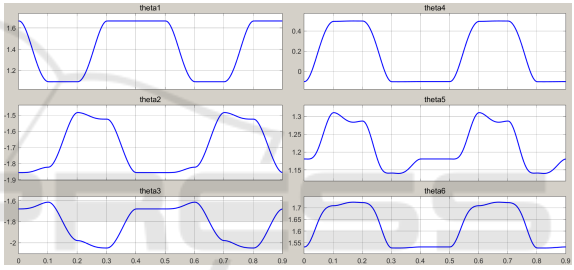

Figure 3: Joint Angle Responses (θ_1 to θ_6) During Nominal Operation for Two Square Trajectories.

space. An overview of the developed digital twin is illustrated in Figure 1, where the virtual model of the Stäubli TX60 highlights the overall diagnostic framework, including simulation and data extraction modules.

As shown in Figure 2, the robot executes two square trajectories. The first trajectory spans from t = 0 s to t = 0.4 s, and the second trajectory starts from t = 0.5 s to t = 0.9 s. These paths are designed to test the system's behavior under both normal and faulty conditions. The corresponding joint angles for both trajectories are presented in Figure 3, which provides insights into the coordinated movements of the robot's six joints (θ_1 to θ_6).

In our previous work (Hnaien et al., 2024), the TX60 was used as a physical reference model for developing a digital twin focused specifically on the arm and shoulder subsystems, with an emphasis on a fault scenario involving the position sensor. Building on that foundation, this study extends the digital twin to the full robot structure and explores motor-level faults under dynamic conditions.

3.2 Fault Injection and Signature Generation

To simulate fault conditions, a dedicated fault injection module was integrated into the Simulink-based digital twin. This module enables the insertion of specific motor faults—namely, blockage of one of the six actuators—at controlled time instants. Each fault scenario corresponds to one motor being locked (from motor 1 to motor 6), allowing the system to exhibit a distinct dynamic deviation in its joint trajectory. These predefined scenarios were injected both manually and randomly through a centralized subsystem managing all fault modes. This ensures consistency across simulations and enables capturing the temporal evolution of joint responses after fault injection.

As illustrated in Figure 4, the angular trajectory of joint 4 (θ_4) deviates noticeably when a blockage fault is introduced at t = 0.2 s, showcasing a fault-induced divergence from the nominal behavior. This approach ensures temporal diversity essential for training a robust classification model. The labeled time series

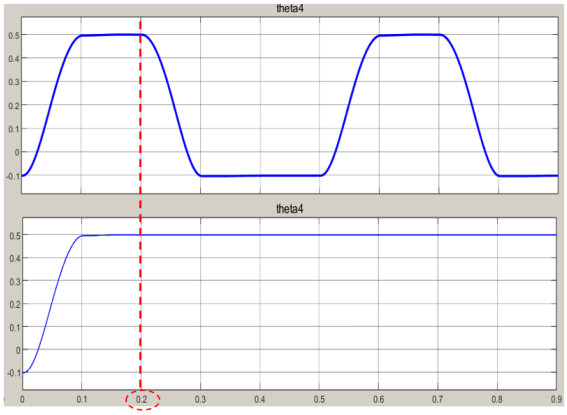

Figure 4: Variation of Joint 4 Angular Trajectory Under Normal Conditions and Following a Motor Blockage at t = 0.2 s.

formed the input to the GRU-based neural network. Although the fault injection times were discretely selected (from 0.1 s to 0.8 s), this strategy provides a structured dataset with consistent fault labels. However, it inherently limits the realism of fault onset behavior, as real-world faults may appear gradually or at arbitrary times. This constraint will be addressed in future work through continuous-time fault injection mechanisms or stochastic fault onset modeling to better reflect practical scenarios.

4 DATA PREPARATION AND DEEP LEARNING MODEL

The success of a data-driven fault classification system critically depends on the quality of the dataset and the architecture of the learning model. This section describes the procedures followed for generating, preprocessing, and structuring the data, as well as the design and training of the GRU-based neural network classifier.

4.1 Data Collection and Preprocessing

The dataset was generated entirely from simulations conducted within the digital twin environment of the 6-degree-of-freedom (6-DOF) Stäubli TX60 robot. The resulting labeled dataset was composed of 49 distinct classes—including 48 fault scenarios generated by injecting motor blockage at 8 time instants for each of the 6 actuators (6×8), plus one nominal condition (see Section 4 for details). In each simulation run, a single fault was injected into one of the six joints (θ_1 to θ_6) by applying a motor blockage at specific time instants ranging from 0.1 to 0.8 seconds. This strategy led to 48 distinct faulty configurations, each corresponding to a unique (joint, time) pair, in addition to a nominal operating state—resulting in a total of 49 target classes.

Each simulation produced continuous multivariate time series of joint angles and end-effector positions. These signals were segmented into overlapping sequences of 10 consecutive time steps to capture the temporal patterns associated with fault dynamics. In order to ensure label consistency within each segment, only sequences for which at least 80% of the samples belonged to the same class were retained. This criterion reduced label noise and improved the robustness of the classifier.

The resulting dataset was composed of samples shaped as 3D tensors of size (number of sequences, 10, number of features). All features were normalized using the StandardScaler method, which standardizes input values to have zero mean and unit variance. This step mitigates the effect of scale differences among input signals and facilitates faster convergence during training. The categorical class labels were converted into one-hot encoded vectors to be compatible with the softmax output layer used in the GRU network.

To ensure fair and balanced evaluation, the dataset was split into training and testing subsets using stratified sampling with an 80/20 ratio. This maintained an even distribution of all fault types across the splits and prevented class imbalance from biasing the learning process.

4.2 GRU-Based Fault Classifier

To classify the fault scenarios from sequential input data, a deep learning model based on Gated Recurrent Units (GRU) was adopted. GRU is a class of Recurrent Neural Networks (RNNs) that have been specifically designed to overcome the vanishing gradient problem often encountered in traditional RNNs during long sequence learning. They achieve this by incorporating update and reset gates, which allow the model to adaptively control the flow of information over time without the complexity of Long Short-Term Memory (LSTM) cells (Cho et al., 2014). The detailed architecture of the model is presented in the Table 1.

The architecture was designed based on preliminary experiments and insights from prior literature, with the goal of balancing model complexity and representational capacity. Specifically, the network comprises two successive GRU layers with 64 and 32 units, respectively. This configuration was found to be sufficient for capturing temporal dependencies in the data while mitigating the risk of overfitting.

Table 1: Details of each layer.

Layer Type	Configuration
GRU	64 units, return sequences, input shape = (10, n_{features})
GRU	32 units
Dense	64 neurons, ReLU activation
Dense	Output layer with 49 neurons, Softmax activation

A fully connected dense layer with 64 neurons and a ReLU activation function was added following the GRU layers to enhance non-linear feature extraction before classification. The final output layer was tailored to the multi-class classification task using a softmax activation.

The model was implemented in `Python` using the TensorFlow/Keras framework. Training was performed using the Adam optimizer, which adaptively adjusts learning rates to accelerate convergence. Categorical cross-entropy was employed as the loss function, suitable for multi-class scenarios. The batch size was set to 32, considering both empirical performance and hardware constraints. Training proceeded for a maximum of 100 epochs.

To promote generalization and prevent overfitting, an early stopping strategy was applied. In particular, 20% of the training set was reserved as a validation set, stratified to preserve the class distribution. The model's validation loss was monitored during each epoch, and training was halted if no improvement was observed for 10 consecutive epochs. This regularization technique effectively reduced overfitting and improved the model's ability to generalize to unseen fault conditions.

Overall, the proposed architecture and training strategy achieve a compromise between model complexity, classification accuracy, and computational efficiency, enabling robust detection of fine-grained fault patterns in short time-series data.

While the current architecture was selected based on preliminary experiments and literature insights, a formal ablation study on GRU units, sequence lengths, and dense layers will be conducted in future work to assess model sensitivity and optimize performance.

5 RESULTS AND PERFORMANCE EVALUATION

To assess the effectiveness of the proposed GRU-based diagnostic model, a comprehensive evaluation was conducted using training history analysis, confusion matrix interpretation, and class-wise classification metrics. These analyses provide insights into the model's ability to generalize, its robustness across fault classes, and its performance consistency on unseen data.

5.1 Training and Validation Accuracy

The model's learning behavior throughout the training process is depicted in Figure 5, which shows the evolution of training accuracy (blue), validation accuracy (red), and final test accuracy (green). The training curve exhibits a rapid rise in accuracy during the early epochs, reaching a plateau at approximately 87.3%, indicating that the model effectively learned the temporal patterns of the input sequences. The validation curve closely follows the training curve, with minimal divergence, suggesting that the model exhibits low overfitting and strong generalization capabilities. Notably, the test accuracy—calculated on a separate held-out dataset—remained consistent at 87.3%, further reinforcing the model's robustness on unseen fault scenarios.

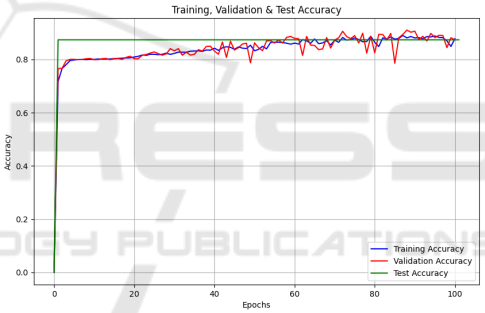

Figure 5: Training, Validation, and Test Accuracy Evolution Over Epochs.

5.2 Confusion Matrix and Class Performance

The classification outcomes for all 49 classes—comprising 48 distinct fault cases generated by blocking one of six motors at eight time steps (0.1 s to 0.8 s), along with one nominal class—are visualized in Figure 6 via the confusion matrix. A strong diagonal dominance is observed, reflecting high precision and recall for most fault categories. The limited misclassifications that occur are primarily concentrated between neighboring time instances within the same joint, which can be attributed to the natural similarity in system dynamics for temporally adjacent faults. For example, a blockage introduced at 0.3 s may produce a response signal closely resembling that of a blockage at 0.4 s for the same joint. Despite these inherent chal-

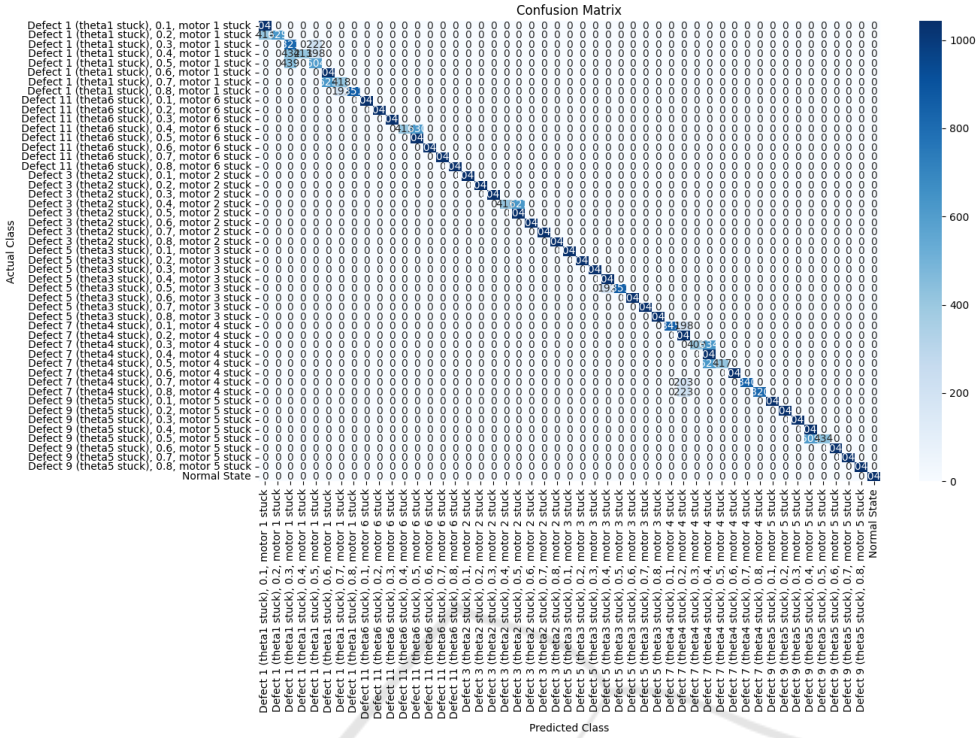

Figure 6: Confusion Matrix of the GRU-Based Fault Classification Model.

lenges, the classifier maintained a high level of class discrimination, even for closely spaced temporal faults.

5.3 Classification Metrics

The quantitative evaluation of the model's predictive performance is summarized in Table 2, which reports the overall and average classification metrics. The model achieved a macro-average F1-score of 86.85%, indicating balanced performance across all classes without bias toward more frequent labels. Precision and recall values exceeded 91% and 87% respectively in their weighted averages, highlighting the model's effectiveness in both correctly identifying fault types and minimizing false negatives. Importantly, the model reached perfect classification for the nominal (fault-free) class, affirming its capability to distinguish between healthy and faulty states with high reliability.

Slightly lower performance was observed in a few fault classes, notably involving motor 1 and motor 4, particularly at mid-range fault injection instants (e.g., 0.4 s or 0.5 s). This degradation is likely due to overlaps in the fault signatures at these time intervals. A deeper analysis using attention or saliency maps could help understand which features contribute most to confusion in these specific fault classes. Moreover, data augmentation or multi-modal inputs (e.g., torque or velocity signals) could improve discriminability.

Nonetheless, the model consistently produced high-quality predictions and maintained overall accuracy at 87.35%, which is considered competitive for high-resolution multi-class time-series classification.

Table 2: Classification Report.

Metric	Value
Accuracy	87.35%
Macro Avg Precision	91.64%
Macro Avg Recall	87.35%
Macro Avg F1-Score	86.85%
Weighted Avg Precision	91.64%
Weighted Avg Recall	87.35%
Weighted Avg F1-Score	86.85%

5.4 Comparison with LSTM Baseline

To contextualize the GRU model's performance, we trained a baseline Long Short-Term Memory (LSTM) model using the same dataset, sequence length, and preprocessing pipeline. The LSTM model achieved a higher test accuracy of 90.51%, slightly outperforming the GRU classifier.

Classification metrics—including precision, re-

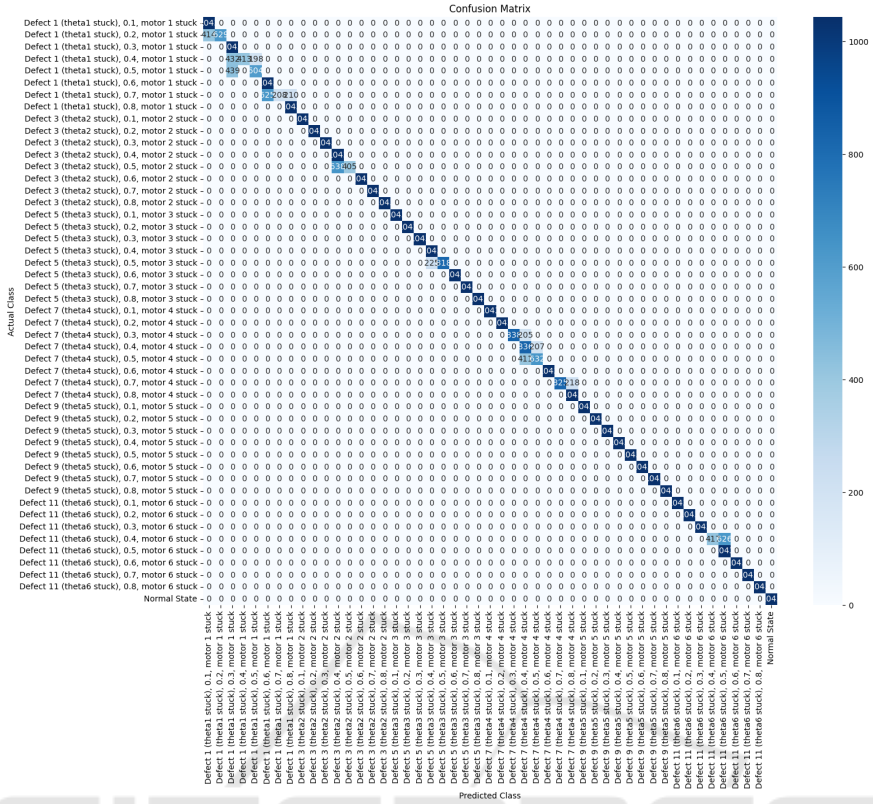

Figure 7: Confusion Matrix of the LSTM-Based Fault Classification Model.

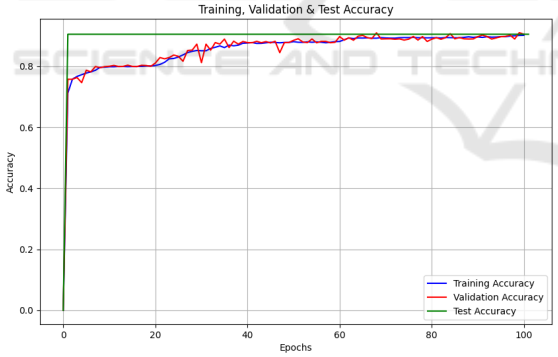

Figure 8: Training, Validation, and Test Accuracy of the LSTM Model Across Epochs.

call, and F1-score—were consistently high across most of the 49 classes, and the model particularly excelled in handling overlapping temporal patterns. The confusion matrix of the LSTM classifier (Figure 7) demonstrates strong diagonal dominance, indicating excellent class separability. In parallel, Figure 8 illustrates the evolution of training, validation, and test accuracy over epochs, revealing stable convergence and effective generalization.

These results highlight that LSTMs, with their sophisticated memory mechanisms, offer improved capacity to capture long-term dependencies in sequential joint signals. However, despite this marginal gain in accuracy, GRUs remain advantageous due to their lower computational complexity and fewer parameters, which make them more suitable for real-time deployment, especially in embedded systems with constrained resources.

Overall, the inclusion of this LSTM baseline confirms the robustness of recurrent architectures for robotic fault classification and reinforces the validity of the GRU-based approach adopted in this study.

These results validate the efficacy of combining a simulation-based data generation approach with GRU neural networks for robotic fault classification. The methodology demonstrates strong potential for deployment in real-time monitoring systems where early and accurate fault detection is critical.

6 CONCLUSION

This work presented a hybrid methodology that combines a high-fidelity digital twin with a GRU-based deep learning architecture for real-time fault diagnosis of the Stäubli TX60 industrial robot. The proposed approach enables the simulation of realistic motor

blockage faults and the generation of large-scale labeled datasets under controlled conditions. These data were used to train a Gated Recurrent Unit (GRU) classifier capable of recognizing 49 distinct operating states, including both nominal and faulty behaviors. The model achieved high classification accuracy (87.35%) and strong generalization across multiple fault scenarios, demonstrating the relevance of temporal features captured through joint trajectories.

The digital twin not only ensured repeatability and cost-effective experimentation but also allowed for fine-grained control over fault injection, improving the quality of diagnostic data. Our findings highlight the effectiveness of combining physics-based simulation and data-driven learning for complex fault diagnosis tasks in robotics. Unlike traditional methods relying on thresholding or signal modeling, this approach enables the automatic identification of fault type, location, and onset timing using only sequential joint data.

While the current GRU classifier was trained on isolated motor jamming faults at discrete time points, this work lays the foundation for future extensions. We acknowledge that compound, cascading, and evolving faults were not covered, and that fault realism was limited by predefined injection times. To enhance generalization, future research will explore (i) dynamic fault injection mechanisms, (ii) extended fault types such as sensor degradation or torque anomalies, and (iii) learning from mixed-fault conditions.

Additionally, although the digital twin offers controlled and repeatable experimentation, validation solely in simulation restricts real-world applicability. We plan to implement hardware-in-the-loop validation and on-site robotic tests to evaluate robustness under physical uncertainties and sensor noise.

In terms of learning architecture, beyond recurrent models like GRU and LSTM, we will investigate more recent temporal models such as Transformer-based encoders and State-Space Sequence Models (SSSM). In particular, the Mamba architecture—which captures temporal dynamics through continuous-time formulations inspired by ordinary differential equations—presents a promising direction. Its alignment with the physics-based nature of robotic systems could enable better interpretability and improved generalization under varying time resolutions.

Overall, this work represents a promising step toward deploying intelligent, real-time fault detection and predictive maintenance in industrial robotic systems.

REFERENCES

Boschetti, G. and Sinico, T. (2024). Designing digital twins of robots using simscape multibody. *Robotics*, 13(4):62.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Grieves, M. (2022). Digital twin: manufacturing excellence through virtual factory replication. 2014. *White Paper*.

Hnaien, I. B., Gascard, E., Simeu-Abazi, Z., and Dhouibi, H. (2024). Methodology of construction of a digital twin: Application to the stäubli robot's arm and shoulder. In *2024 IEEE International Conference on Artificial Intelligence & Green Energy (ICAIGE)*, pages 1–6. IEEE.

Lee, J., Bagheri, B., and Kao, H.-A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23.

Liu, S., Wang, L., and Wang, X. V. (2021). Sensorless force estimation for industrial robots using disturbance observer and neural learning of friction approximation. *Robotics and Computer-Integrated Manufacturing*, 71:102168.

Sabry, A. H. and Amirulddin, U. A. B. U. (2024). A review on fault detection and diagnosis of industrial robots and multi-axis machines. *Results in Engineering*, page 102397.

She, J., Miyamoto, K., Han, Q.-L., Wu, M., Hashimoto, H., and Wang, Q.-G. (2022). Generalized-extended-state-observer and equivalent-input-disturbance methods for active disturbance rejection: Deep observation and comparison. *IEEE/CAA Journal of Automatica Sinica*, 10(4):957–968.

Yin, S. and Kaynak, O. (2015). Big data for modern industry: challenges and trends [point of view]. *Proceedings of the IEEE*, 103(2):143–146.