Atef Gharbi[1,2]
Faheed A.F. Alrslani[3]
Mohamed Ayari[3,4]
Yamen El Touati[5,6]
Abdulsamad Ebrahim Yahya[3]
Zeineb Klai[5]

Ameliorating Robot Execution Failures with Artificial Neural Networks: A Comprehensive Comparative Study

Abstract: **-** The aim of this study is to investigate how ANNs can be trained on a personalized data set that contains several failure scenarios to predict and prevent execution errors in robotic systems. Three ANN architectures: ANN1 (single hidden layer), ANN2 (two hidden layers) and ANN3 (three hidden layers) were evaluated to determine their effectiveness in distinguishing between normal operating states and various fault conditions such as collisions and obstructions. The performance of the models is rigorously analyzed using several indices, including the average square error (MSE), precision, accuracy and F1 score. The results show that the ANN1 model has a simpler architecture and is more accurate in making predictions than more complex models. This shows that personalized machine learning approaches can make robotic systems safer and more reliable.

Keywords: Robot execution failures, artificial neural network (ANN) model, Fault prediction, performance metrics.

1. Introduction

Robotic systems are increasingly deployed across many industries, such as manufacturing, healthcare, logistics, and services, to enhance accuracy, speed, and efficiency. However, reliability and safety become increasingly important as robots operate in dynamic and unpredictable environments. It is common for robotic systems to encounter errors, including actuator failures, sensor malfunctions, and process interruptions caused by collisions, obstructions, or object handling issues. Consequently, these errors can cause costly downtime, reduced productivity, and expensive repairs.

For robotic systems to function safely and reliably, it is crucial that faults are detected and fixed as soon as possible. The traditional methods of fault detection, such as model-based approaches, rely on mathematical models to predict expected behavior. As an example, observer-based methods estimate system states based on actual measurements [1], parity space techniques generate residuals to indicate faults when predefined thresholds are exceeded [2], and Kalman filters provide optimal state estimation despite noise [3]. However, these methods have significant limitations when applied to complex, real-world scenarios where system dynamics are difficult to model precisely and are often affected by external factors and intrinsic uncertainties [4].

The use of data-driven approaches, such as Artificial Neural Networks (ANNs), has emerged as an effective tool for fault detection. ANNs are capable of modeling complex, nonlinear relationships in data without requiring precise system models, which makes them particularly suitable for predicting and detecting robotic faults [5]. ANNs provide a more robust and flexible solution for dynamic environments since they can identify fault patterns by leveraging large datasets. Fuzzy logic [6], genetic algorithms [7], and neural networks [8] have shown promise in analyzing large and intricate datasets to diagnose faults in dynamic environments.

[1]Department of Information Systems, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia, atef.gharbi@nbu.edu.sa
[2]LISI Laboratory, National Institute of Applied Sciences and Technology (INSAT), University of Carthage, Carthage 1054, Tunisia
[3]Department of Information Technology, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia
[4]SYSCOM Laboratory, National Engineering School of Tunis, University of Tunis El-Manar, Tunis 1068, Tunisia.
[5]Department of Computer Sciences, Faculty of Computing and Information Technology, Northern Border University, Rafha, Saudi Arabia
[6]OASIS Laboratory, National Engineering School of Tunis, University of Tunis El Manar, Tunisia.

Furthermore, recent studies have demonstrated the effectiveness of hybrid and deep learning-based approaches to enhance fault detection accuracy in industrial robots. For instance, Long Short-Term Memory (LSTM) networks have been employed for model-based error detection in automation systems [9], while complex pattern recognition has demonstrated success in diagnosing robotic faults using deep convolutional neural networks and residual-convolutional neural networks [10]. It is important to note, however, that deep learning models generally require considerable labeled data and computational resources, which are limiting their application in real-time or resource-constrained situations.

ANNs are used in this study to predict and mitigate robotic execution faults. Our research examines three ANN architectures with varying complexities to detect collisions and obstructions, including a single hidden layer, two hidden layers, and three hidden layers. The goal of our study is to identify an optimal balance between simplicity and performance, in order to develop robotic systems that are more reliable and safer.

The remainder of the paper is organized as follows: Section 2 analyzes the research on fault detection techniques used in robotic systems, emphasizing their advantages and disadvantages. Detailed descriptions of the models and configurations of artificial neural networks, as well as the design choices and training procedures, are presented in Section 3. In Section 4, the experimental results and analysis are presented, and the abilities of various ANN architectures to predict failure are compared. The final section of the study discusses the major findings and provides suggestions for future research.

2. Related Works

Detecting faults in robotic systems is critical to their safe and reliable operation. In recent years, researchers have developed a wide range of methods to enhance fault detection, ranging from traditional model-based methods to modern data-driven approaches, including machine learning and deep learning.

Model-Based Approaches:

In the field of fault detection, mathematical models have played a key role in predicting expected behavior and detecting anomalies. For example, Doostmohammadian and Meskin [11] used networked estimation techniques to detect and isolate sensor faults in sensor networks, resulting in a robust approach to handling fault scenarios. [12] Wang and Shen discussed parity space techniques for fault detection, in which residuals from system outputs are compared against predefined thresholds to identify anomalies. Accordingly, Rezvani et al. [13] showed that Kalman-based approaches can be used in dynamic environments with measurement noise to estimate vertical land motion and overcome residual systematic errors.

While model-based approaches are effective, developing accurate mathematical models is challenging in complex, nonlinear systems. As an example, Liu et al. [14] pointed out that model-based methods may not be sufficient for advanced robotic systems involving skill transfer learning and human-robot cooperation.

Data-Driven and Machine Learning Approaches:

Data-driven approaches are gaining popularity as a result of the limitations of model-based techniques. In their study of neural networks and fault diagnosis for mechanical systems, Xu et al. [15] demonstrated that neural networks are capable of modeling complex, nonlinear relationships without requiring precise system models. Similarly, Koppen-Seliger and Frank [16] demonstrated the use of fuzzy logic combined with neural networks to enhance fault detection capabilities, enabling more flexible and adaptive systems that can learn from large amounts of information.

These data-driven methods have been refined further by recent advances. Based on an adaptive genetic algorithm optimized back-propagation neural network, Yu and Wang [17] demonstrated the potential for combining optimization algorithms with neural networks to improve the speed and accuracy of fault detection in liquid rocket engines. A comprehensive survey of run-time monitoring of machine learning models for robotic perception was provided by Rahman et al. [18], illustrating emerging trends in using AI to improve robotic fault detection.

Hybrid and Deep Learning Approaches:

To improve fault detection in industrial robots, hybrid approaches have also been developed that combine different machine learning techniques. Using machine learning to predict outages and assess reliability factors, Aliev and

Antonelli [19] proposed a monitoring system for collaborative robots. Ding et al. [20] demonstrated that model-based error detection can be achieved using Long Short-Term Memory (LSTM) networks, which can handle sequential data and capture temporal dependencies.

Fault detection applications have shown promising results using deep learning methods, especially those involving convolutional neural networks (CNNs). Wang et al. [21] used deep learning to identify faults in ultrasonic welding using complex data patterns for early event detection. In addition, Koppen-Seliger and Frank [22] explored hybrid approaches for fault detection that incorporate neural networks and fuzzy logic, allowing for more accurate and adaptable systems.

However, deep learning models are also challenged by data scarcity and computational overhead. Using deep learning methods for fault detection in industrial robotic systems, Zhang et al. [23] examined their potential and limitations. A study by He et al. [24] examined how deep convolutional neural networks can be employed to detect faults in robotic joints, whereas a study by Chen et al. [25] discussed the limitations of deep learning models in fault detection due to the need for large datasets and significant computational resources.

Researchers have developed more efficient and adaptive deep learning models to address these challenges. As an example, Oh et al. [26] developed a motion-adaptive fault detection method that can be applied across a variety of industrial robots by using residual convolutional neural networks. Patel et al. [27] developed a hybrid artificial neural network-based approach for controlling aerial robots by combining neural networks with adaptive learning techniques. Also, Xia et al. [28] demonstrated the use of digital twins to train deep reinforcement learning agents for smart manufacturing plants, providing a virtual environment to improve fault detection and control.

3. Artificial Neural Network Model

In Artificial Neural Networks (ANNs), artificial neurons or nodes perform parallel computation and knowledge representation for a wide range of data processing tasks. They are inspired by the biological neural framework and consist of artificial neurons or nodes. Throughout these synthetic neurons, weighted connections facilitate information exchange and interaction. When dealing with noisy data, complex relationships, and scenarios where traditional mathematical methods are inadequate, artificial neural networks have proven both reliable and effective in tasks such as prediction, classification, and clustering.

Through iterative weight adjustments between neurons, ANNs discover hidden patterns in data. They are particularly suitable for predicting robot execution failures due to their nonlinearity, parallelism, robustness, fault tolerance, learning capability, adaptability to imprecise information, and generalizability. In an ANN, the activity of each neuron is determined by the weighted sum of its inputs and the value of its activation function. In ANNs, the weighted sum determines the output of the neuron, while the activation function determines its amplitude.

3.1 ANN Model Representation

In ANN models, input is defined for each neuron in the input layer, and connections to neurons in subsequent layers are established via weighted connections. Weights range from -1 to 1, indicating the significance of each connection. An activation function is applied to the weighted input to calculate the output of each neuron. The neuron is activated if the resulting value exceeds its internal threshold; otherwise, it remains inactive.

In the output layer, the difference between the actual output and the desired output is quantified as an error. A process known as backpropagation adjusts the weights in the input layer based on the error propagated from the output layer. Iteratively, the error is reduced until it falls below a threshold. Upon completion of the training phase, the validation and testing phases begin. Parameters such as training duration, learning rates, and the number of hidden layer neurons are fine-tuned during validation. Testing involves evaluating how well the network generalizes to new samples. The final solution is the model that demonstrates optimal performance.

Figure 1 illustrates the main phases involved in configuring an artificial neural network (ANN). Defining the network architecture, selecting activation functions, and setting the number of epochs all require manual adjustments, while other steps require automatic adjustments, such as loading, normalizing, and initializing weights, based on Python code. Among the input characteristic vectors for the neural network, there is one group representing normal operation, and three groups representing different types of errors: collision, obstruction, and

frontal collision. As training data, each group contains 100 samples. Python's random number generation functions are used to initialize the biases and weights during configuration.

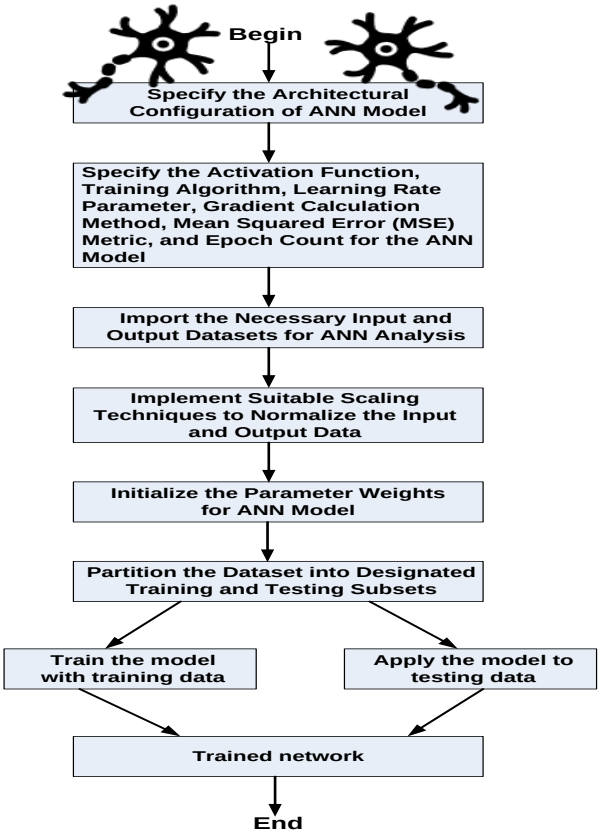

Figure 1. Development Steps of the ANN Model

3.2 Dataset Segmentation

An ANN is employed in this study to predict, detect, classify, and monitor robot failures during specific tasks. This dataset captures the evolution of forces (Fx, Fy, Fz) and torques (Tx, Ty, Tz) during task execution, especially failures related to the approach to the grasp position. A dataset containing 88 instances contains 15 sensor measurements collected at regular intervals. A sample consists of three force and torque values, resulting in 90 features per sample. It contains 1,320 examples divided into 88 instances, available through a reputable machine learning repository [29]. The distribution of samples across different categories is unbalanced: collision (24%), front collision (19%), normal (18%), and obstruction (39%). To prevent bias and ensure balanced training, the dataset is divided into three subsets: training, validation, and testing.

3.3 Normalization

The normalization of input data ensures that all features contribute equally to the training process, thus preventing feature dominance and numerical overflow problems caused by excessively large weights. According to Eq. (1), input data are normalized to a range of 0 to 1 using a Z-score normalization method.

$$Z = \frac{x - \bar{x}}{\sigma} \qquad (1)$$

where x is a random variable, μ is the sample mean, and σ is the standard deviation of the random variable x. In addition to its faster convergence rate, Z-score normalization is also less sensitive to outliers. There are various termination criteria during training, such as achieving a specific mean square error (MSE), reaching a predefined number of epochs, or satisfying a minimum performance gradient threshold. MSE is calculated as shown in Eq. (2):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2 \qquad (2)$$

The total number of output neurons is n, the actual output is y_i, and the predicted output is x_i. New input data must be normalized using the same method after training, and output results must be denormalized.

3.4 Prediction Procedure

A multilayer feedforward artificial neural network (ANN) designed for fault detection in robotic systems is illustrated in figure 2. Three primary components make up the network: the input layer, one or more hidden layers, and the output layer.

- The input layer represents the initial stage of the neural network, in which the input data is fed into the network. Figure 2 shows six input nodes, representing six input features:

✓ Fx, Fy, and Fz are measurements of force along x, y, and z axes.

✓ Tx, Ty, and Tz represent torque measurements around the x, y, and z axes, respectively.

Input layer nodes take values from the dataset, representing force and torque measurements recorded by sensors during the robot's operation.

- A hidden layer is composed of multiple neurons (nodes) that process inputs from an input layer. Neurons in the hidden layer receive weighted inputs from all nodes in the input layer. Weights are adjusted between the input layer and hidden layer during the training process to minimize the error in the network's predictions. A hidden layer applies an activation function to these weighted inputs, introducing nonlinearity and allowing the network to learn complex patterns. Based on the input it receives, the activation function determines whether a neuron should be activated.

- The output layer consists of four nodes that represent the output classes or scenarios the ANN is designed to classify:

✓ Normal: Represents normal operational conditions without any faults detected.

✓ Collision: It represents a collision fault scenario in which an unexpected collision occurs between the robot and another object.

✓ Obstruction: Represents a situation where an object obstructs the robot's movement.

✓ Frontal Collision: Robot encounters a head-on collision in a frontal collision fault scenario.

Based on the input data, each node in the output layer generates a probability score or output value.

With ANNs, the output neurons are mapped so that the first neuron indicates normal operation, and the subsequent neurons indicate simulated faults such as collisions or obstructions. As an example, a normal operation will result in [1, 0, 0, 0], but a collision fault will result in [0, 1, 0, 0]. Input training vectors have a dimensionality of (1,6), comprising force and torque values (Fx, Fy, Fz, Tx, Ty, Tz), whereas output training vectors have a dimensionality of (1,4).

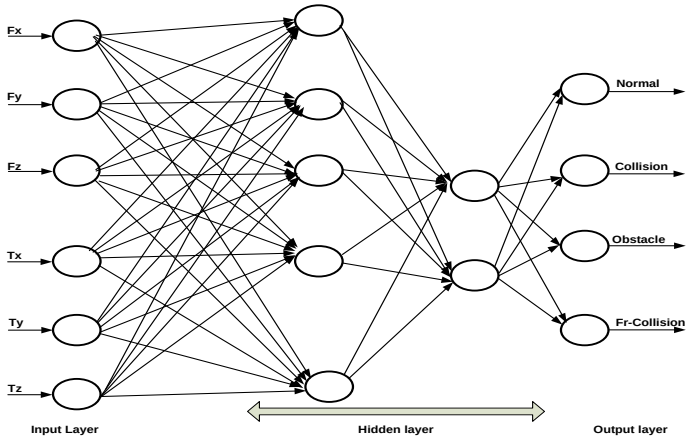

Figure 2. Structure of the Proposed Multilayer Feedforward Neural Network

Different learning algorithms are evaluated to determine the ANN's optimal performance, including those effective at classifying, recognizing patterns, and approximating nonlinear functions. In the Python implementation, sigmoid and linear activation functions are used in the hidden and output layers, respectively. To terminate training, either a target MSE of 10^{-5} or a maximum of 1,000 iterations must be reached.

In this paper, three ANN models are developed, each with a different architectural configuration: ANN1, ANN2, and ANN3. ANN1 has a single hidden layer with 10 neurons, ANN2 has two hidden layers with 10 neurons each, and ANN3 has three hidden layers with 10 neurons each. As outlined in Table 1, the complexity of the model increases from ANN1 to ANN3, impacting the network's ability to recognize intricate data patterns.

Table 1. ANN Configuration Parameters

Parameter	**Value**
Number of input layer neurons	6
Number of output layer neurons	4
Number of hidden layer neurons	1..3
Number of neurons by hidden layer	10
Hidden layer activation function	Sigmoid
Output layer activation function	Linear
Learning rate	0.05
MSE stopping criteria	10^{-4}
Maximum number of epoch	6000

This setup enables effective learning and optimal model performance, ensuring the ANN models' suitability for detecting and classifying faults in robotic systems.

4. Assessment of ANN Models

An extensive investigation was conducted to determine the optimal architecture for the artificial neural network (ANN) model. Different ANN models were developed, varying in the number of hidden layers (from 1 to 3) and neurons per layer (from 10 to 30). Different activation functions and training algorithms were used to train each model. Based on the parameters specified in Table 1, each ANN model was initialized with ten different random sets of weights and biases. Based on the variations in hidden layers and neuron configurations, three ANN architectures were systematically evaluated. There were two distinct sets of experimental trials, each designed to achieve a different goal. In the first set, the training algorithms were assessed for convergence rates, while in the second set, the models were evaluated for classification accuracy.

4.1 Assessment Based on the Mean Square Error Performance Index

ANN models were trained over a maximum of 5,000 iterations in the initial phase of the experiment. At each iteration, the mean square error (MSE) between the predicted outputs and the target vectors was calculated. These MSE values were averaged over 100 iterations and 50 epochs to provide a stable measure of performance. Analyzing the learning rate of the three different ANN models was based on their average MSE values.

All three ANN models stabilized at approximately 0.05 after around 2,800 iterations, as shown in Figure 3. ANN1 was the fastest to achieve the lowest MSE, followed by ANN2 and ANN3. Accordingly, the ANN1 model, with a simpler architecture, minimizes error during training more efficiently.

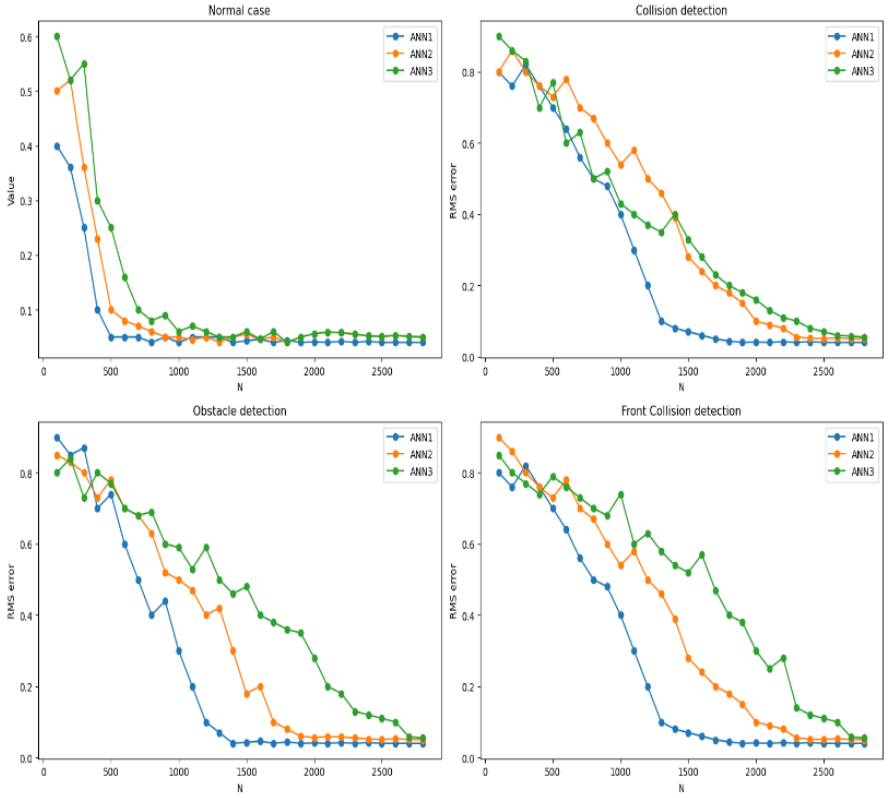

Figure 3. Comparative Analysis of ANN Models for Normal, Collision, Obstruction, and Front Collision Detection

Based on the analysis, the MSEs for each model tend to decrease as the sample size increases, which makes sense since more data is available for training. Overall, ANN1 consistently achieved lower MSE values than ANN2 and ANN3 across most sample sizes, indicating superior performance in predicting normal operations, collisions, obstructions, and front collisions. ANN architectures, including ANN1, ANN2, and ANN3, converge faster under normal conditions, but more slowly under fault conditions.

Figure 4 shows the number of iterations each ANN model needs to achieve an MSE value below 0.05. MSE serves as a criterion for evaluating the learning process. The classification accuracy was evaluated with 3,000 repetitions of iteration. In order to confirm the distinct convergence characteristics of each feature vector and class identifier, new data was collected.

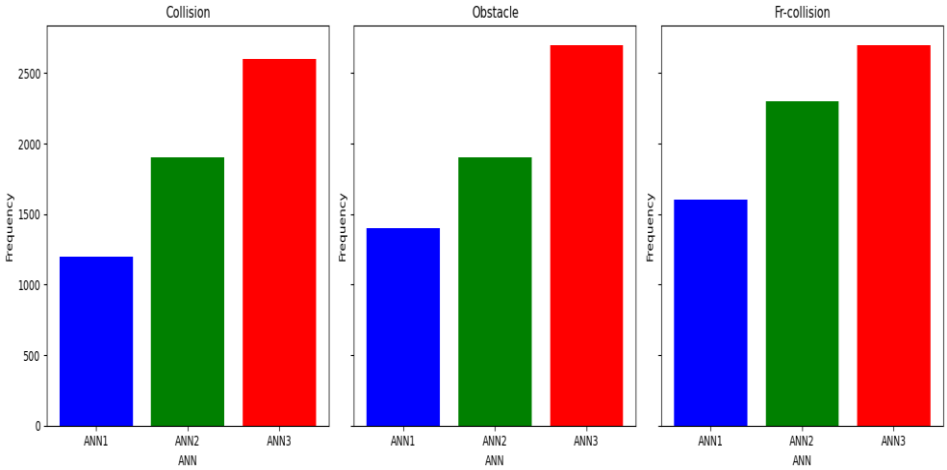

Figure 4. Number of Iterations to Achieve (RMS < 0.06)

4.2 Assessment Based on Performance Metrics

Figure 5 presents a confusion matrix that shows the classification outcomes of robot failure scenarios during testing to provide a clearer visual representation of the performance of the ANN models. The confusion matrix is an essential tool for evaluating a classification model's performance, specifically when assessing its accuracy across various test instances. In addition, it provides a comprehensive visual representation of how well the algorithm categorizes various events.

Figure 5 shows four distinct output classes: normal operation, collision faults, obstruction faults, and front-collision faults. Those events classified correctly are indicated by the diagonals, whereas those classified incorrectly are represented by the off-diagonals. ANN models are used to predict values based on actual values. In some cases, faults are misclassified as normal operational events, highlighting areas for further refinement.

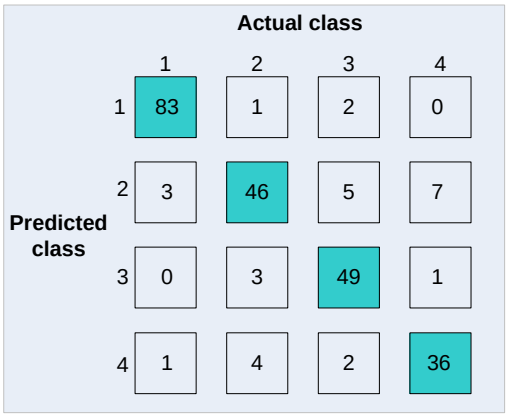

Figure 5. Confusion Matrix for Failure Classification with Test Data

According to the confusion matrix, 214 of the 243 samples were correctly classified, resulting in an overall accuracy of 88.06%. It is noteworthy that the classifiers effectively differentiate between specific classes, for example, class 1 (normal operation) versus class 4 (front-collision faults), and class 3 (obstruction faults) based on zero entries in certain off-diagonal elements, such as M[1,4] = 0 and M[3,1] = 0). It is necessary, however, to improve the model's performance, particularly in class 2 (collision faults) that has 15 misclassified samples, the most of all the classes.

A binary-class confusion matrix can be used to analyze classification performance more granularly (Figure 6). It provides a detailed evaluation of a model's performance for each class by calculating class-specific metrics such as precision, recall, and accuracy.

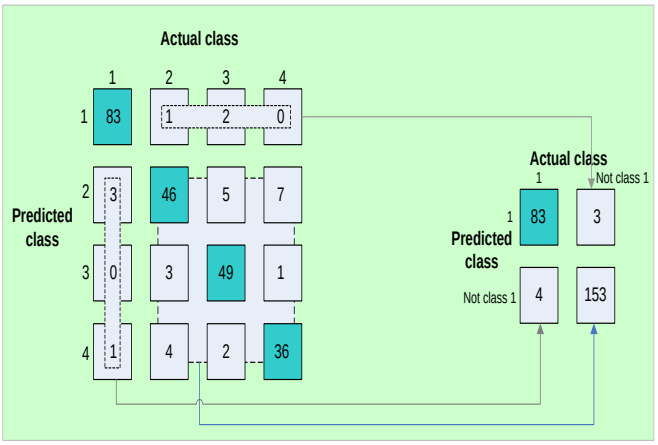

Figure 6. Binary-Class Confusion Matrix

The Binary-class confusion matrix reveals true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) of the model's performance. Analyzing the model's classification functions in detail goes beyond

basic accuracy measurements. Figure 7 shows the binary-class confusion matrix, which represents the model's performance in binary classification scenarios.

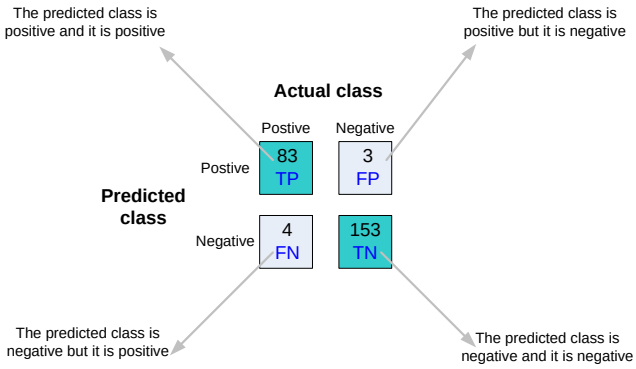

Figure 7. Confusion Matrix for a Binary Classifier

A comprehensive analysis of the output predictions of four ANN models is provided in Table 2. There are performance metrics for each ANN model in each class, such as the percentage of correct and incorrect predictions.

Table 2. Comparative Analysis of Output Predictions Across Three ANN Models

Class	Class 1	Class 2	Class 3	Class 4
ANN1 prediction	83	46	49	36
ANN2 prediction	82	45	43	34
ANN3 prediction	80	42	41	33

Performance Metrics Overview:

- **Precision:** Measures the accuracy of positive predictions by indicating the proportion of correctly identified positive cases among all cases predicted as positive.

Precision = TP/(TP+FP)

- **Accuracy:** Represents the overall correctness of the predictions made by the model across all classes.

Accuracy= (TP+TN)/Total

- **Sensitivity (Recall or TPR):** Measures the proportion of actual positive cases that the model correctly identified.

Sensitivity = TP/(TP+FN)

- **F1 Score:** The harmonic mean of precision and sensitivity, providing a balance between precision and recall.

F1 score = 2*(Precision * Recall)/(Precision + Recall)

- **Error Rate:** Quantifies the proportion of misclassifications made by the model across all classes.

Error rate = (FP+FN)/Total

- **Miss Rate:** Represents the proportion of false negatives, indicating cases where the model failed to correctly identify positive instances.

Miss rate = FN/(TP+FN)

Table 3 shows that ANN1 consistently outperforms ANN2 and ANN3 in precision, accuracy, sensitivity, and F1 score across most classes. The performance of ANN3 is slightly lower, especially in precision and sensitivity, whereas ANN2's metrics align closely with those of ANN1, with some minor differences in precision and accuracy. Across all three models, class 1 (normal operation) generally exhibits the highest precision and accuracy, indicating effective differentiation. Among the three fault scenarios, classes 2, 3, and 4 demonstrate varying performance across the models, although classes 2 and 4 are more difficult to classify accurately. In order to improve the models' ability to accurately predict different classes, further analysis is needed.

Table 3. Performance metrics used to compare the three ANN models.

	TP	**FP**	**FN**	**TN**	**Precision**	**Accuracy**	**Sensitivity**	**F1 score**	**Error rate**	**Miss rate**
ANN1	83	3	4	153	0.97	0.97	0.95	0.96	0.03	0.05
ANN2	82	4	6	151	0.95	0.96	0.93	0.94	0.04	0.07
ANN3	80	6	10	147	0.93	0.93	0.89	0.91	0.07	0.11

5. Conclusion

Our study of artificial neural networks (ANNs) for predicting robot execution failures has demonstrated that simpler models, such as the ANN1, are effective at identifying normal and fault conditions. The results of these experiments demonstrate the potential for ANNs to improve the safety and reliability of robotic systems. To enhance prediction accuracy and generalizability across platforms and environments, future research should incorporate advanced neural network architectures, such as convolutional and recurrent neural networks.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number “NBU-FFR-2024-2441-03“.

References

[1] Wei, G., Liu, L., Wang, L., & Ding, D. (2020). Event-triggered control for discrete-time systems with unknown nonlinearities: An interval observer-based approach. International Journal of Systems Science, 51(6), 1019-1031.

[2] Jihani, N., Kabbaj, M. N., & Benbrahim, M. (2023). Sensor fault detection and isolation for smart irrigation wireless sensor network based on parity space. International Journal of Electrical and Computer Engineering, 13(2), 1463.

[3] Xu, Y., Shmaliy, Y. S., Bi, S., Chen, X., & Zhuang, Y. (2023). Extended Kalman/UFIR filters for UWB-based indoor robot localization under time-varying colored measurement noise. IEEE Internet of Things Journal, 10(17), 15632-15641.

[4] Abid, A., Khan, M. T., & Iqbal, J. (2021). A review on fault detection and diagnosis techniques: basics and beyond. Artificial Intelligence Review, 54(5), 3639-3664.

[5] Miri, D., Khedher, A., & BenOthman, K. (2021, March). Tracking of trajectory and fault estimation of MIABOT robot using an artificial neural network. In 2021 18th International Multi-Conference on Systems, Signals & Devices (SSD) (pp. 1296-1301). IEEE.

[6] Nasser, A. R., Azar, A. T., Humaidi, A. J., Al-Mhdawi, A. K., & Ibraheem, I. K. (2021). Intelligent fault detection and identification approach for analog electronic circuits based on fuzzy logic classifier. *Electronics*, *10*(23), 2888.

[7] Liu, P., & Zhang, W. (2019). A fault diagnosis intelligent algorithm based on improved BP neural network. International Journal of Pattern Recognition and Artificial Intelligence, 33(09), 1959028.

[8] Lu, K., Chen, C., Wang, T., Cheng, L., & Qin, J. (2022). Fault diagnosis of industrial robot based on dual-module attention convolutional neural network. Autonomous Intelligent Systems, 2(1), 12.

[9] Ding, S., Morozov, A., Vock, S., Weyrich, M., & Janschek, K. (2020). Model-based error detection for industrial automation systems using LSTM networks. In Model-Based Safety and Assessment: 7th International Symposium IMBSA 2020 (pp. 212-226). Springer International Publishing.

[10] Xu, Y., Yan, X., Sun, B., & Liu, Z. (2022). Global contextual residual convolutional neural networks for motor fault diagnosis under variable-speed conditions. Reliability Engineering & System Safety, 225, 108618.

[11] Doostmohammadian, M., & Meskin, N. (2020). Sensor fault detection and isolation via networked estimation: Full-rank dynamical systems. IEEE Transactions on Control of Network Systems, 8(2), 987-996.

[12] Wang, Z., & Shen, Y. (2022). Parity Space Approaches. In Model-Based Fault Diagnosis: Methods for State-Space Systems (pp. 173-200). Singapore: Springer Nature Singapore.

[13] Rezvani, M. H., Watson, C. S., & King, M. A. (2021). Estimating Vertical Land Motion and Residual Altimeter Systematic Errors Using a Kalman-Based Approach. Journal of Geophysical Research: Oceans, 126(6), e2020JC017106.

[14] Liu, Y., Li, Z., Liu, H., & Kan, Z. (2020). Skill transfer learning for autonomous robots and human–robot cooperation: A survey. Robotics and Autonomous Systems, 128, 103515.

[15] Xu, X., Cao, D., Zhou, Y., & Gao, J. (2020). Application of neural network algorithm in fault diagnosis of mechanical intelligence. Mechanical Systems and Signal Processing, 141, 106625.

[16] Koppen-Seliger, B., & Frank, P. M. (2020). Fuzzy logic and neural networks in fault detection. In Fusion of Neural Networks, Fuzzy Systems, and Genetic Algorithms (pp. 169-210). CRC Press.

[17] Yu, H., & Wang, T. (2021). A method for real-time fault detection of liquid rocket engines based on adaptive genetic algorithm optimizing back propagation neural network. Sensors, 21(15), 5026.

[18] Rahman, Q. M., Corke, P., & Dayoub, F. (2021). Run-time monitoring of machine learning for robotic perception: A survey of emerging trends. IEEE Access, 9, 20067-20075.

[19] Aliev, K., & Antonelli, D. (2021). Proposal of a monitoring system for collaborative robots to predict outages and assess reliability factors exploiting machine learning. Applied Sciences, 11(4), 1621.

[20] Ding, S., Morozov, A., Vock, S., Weyrich, M., & Janschek, K. (2020). Model-based error detection for industrial automation systems using LSTM networks. In Model-Based Safety and Assessment: 7th International Symposium IMBSA 2020 (pp. 212-226). Springer International Publishing.

[21] Wang, B., Li, Y., Luo, Y., Li, X., & Freiheit, T. (2021). Early event detection in a deep-learning driven quality prediction model for ultrasonic welding. Journal of Manufacturing Systems, 60, 325-336.

[22] Koppen-Seliger, B., & Frank, P. M. (2020). Intelligent fault detection: Hybrid approaches combining neural networks and fuzzy logic. In Hybrid Intelligent Systems (pp. 231-246). Springer.

[23] Zhang, H., Li, Y., & Liu, J. (2021). A survey on deep learning methods for fault detection in industrial robotic systems. Robotics and Computer-Integrated Manufacturing, 68, 101847.

[24] He, X., Wu, L., Zhang, Y., & Liu, H. (2022). Deep convolutional neural networks for fault detection in robotic joints. IEEE Transactions on Robotics, 38(3), 678-690.

[25] Chen, G., Huang, S., & Li, X. (2021). Limitations of deep learning models for fault detection in robotic systems: Data scarcity and computational overhead. IEEE Transactions on Automation Science and Engineering, 18(2), 570-583.

[26] Oh, Y., Kim, Y., Na, K., & Youn, B. D. (2022). A deep transferable motion-adaptive fault detection method for industrial robots using a residual-convolutional neural network. ISA Transactions, 128, 521-534.

[27] Patel, S., Sarabakha, A., Kircali, D., & Kayacan, E. (2020). An intelligent hybrid artificial neural network-based approach for control of aerial robots. Journal of Intelligent & Robotic Systems, 97, 387-398.

[28] Xia, K., Sacco, C., Kirkpatrick, M., Saidy, C., Nguyen, L., Kircaliali, A., & Harik, R. (2021). A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment interfaces and intelligence. Journal of Manufacturing Systems, 58, 210-230.

[29] https://www.kaggle.com/datasets/prashant111/robot-execution-failures