International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

Fault Detection and Classification in Robotic Manipulator Tasks using Multi Class Classification

Dr. Phalguna Rao[1], Alladi Navateja[2], Byreddy Nishitha Reddy[2], Neduri Jeshwanth[2], Theegala Akshaya[2], Vadla Vamshi Chary[2]

[1]Professor,[2]UG Student,[1,2]Department of Computer Science and Engineering (AI & ML)
[1,2]Malla Reddy Engineering College and Management Science, Kistapur, Medchal-501401, Hyderabad, Telangana, India

ABSTRACT

Robotic manipulators play a vital role in modern industrial automation, and ensuring their reliable performance is crucial for achieving efficient and safe operations. Fault detection and classification in robotic manipulator tasks are essential to identify anomalies and prevent potential failures that could lead to hazardous situations or costly production errors. By analyzing force and torque measurements, these systems can identify different types of failures occurring during critical phases of robotic operations. Such knowledge empowers operators to take timely corrective actions, enhancing the safety, reliability, and productivity of robotic manipulators in industrial settings. In this research, we propose a multi-class classification approach for fault detection and classification in robotic manipulator tasks using the provided force and torque measurements. Our method leverages AI models to automatically extract informative features from the sequential force/torque data. By training the model, each representing a different learning problem (LP1 to LP5), the system can effectively learn to differentiate between distinct types of failures. Here, LP1 indicates failures in approach to grasp position, LP2 indicates failures in transfer of a part, LP3 indicates position of part after a transfer failure, LP4 indicates failures in approach to ungrasp position, and LP5 indicates failures in motion with part.

Keywords: Torque Measurements, Reliability, Manipulator, Production, Statistical, Anomalies.

1. INTRODUCTION

The research topic, "Fault Detection and Classification in Robotic Manipulator Tasks using Multi Class Classification," signifies a significant stride in the domain of robotics and automation, aiming to enhance the reliability and performance of robotic manipulators. Robotic manipulators play a pivotal role in various industries, including manufacturing, healthcare, and logistics, where precision and uninterrupted operation are imperative. This research focuses on developing a multi-class classification framework that can identify and categorize faults in robotic manipulator tasks, ultimately optimizing safety, efficiency, and productivity [1]. The motivation for this research is deeply rooted in the increasing integration of robotic systems into critical industrial processes. As automation technologies become more prevalent, the ability to detect and respond to faults in robotic manipulator tasks becomes paramount. Faults, whether caused by mechanical wear, sensor malfunctions, or external factors, have the potential to disrupt operations, compromise safety, and lead to costly downtime. Therefore, there is a compelling need to develop robust and efficient fault detection and classification mechanisms to ensure the dependable performance of robotic manipulators [2].

This research recognizes that multi-class classification techniques, driven by machine learning and data analysis, offer a promising solution. By training models to distinguish between various types of faults, such as sensor failures, joint misalignments, or external interferences, it becomes possible to identify issues early in the robotic task execution process. This, in turn, enables timely interventions, prevents catastrophic failures, and facilitates predictive maintenance, ultimately minimizing operational disruptions and maintenance costs [3]. Furthermore, the research underscores the ethical and practical implications of deploying fault detection systems in robotics. It emphasizes the importance of safety, risk mitigation, and ethical considerations in the design and implementation of robotic systems, ensuring that technological advancements align with ethical standards and the well-being of human operators and bystanders [4]. In this introductory overview, we will delve into this research's key components and objectives. We will explore the significance of fault detection in robotic manipulator tasks, introduce the role of multi-class classification, and highlight the transformative potential of this

research in improving the reliability and safety of robotics in various industries. Additionally, we will underscore the ethical considerations and real-world applications of this research, which extend across manufacturing, healthcare, logistics, and beyond [5]. The "Fault Detection and Classification in Robotic Manipulator Tasks using MultiClass Classification" represents a pioneering effort to enhance the dependability of robotic manipulators. By developing advanced fault detection mechanisms, this research aims to optimize the performance of robotic systems, minimize operational disruptions, and promote safety, all while upholding ethical standards and responsible technology use in the field of robotics and automation.

The motivation for conducting research on "Fault Detection and Classification in Robotic Manipulator Tasks using MultiClass Classification" arises from the ever-increasing integration of robotics and automation into diverse industrial sectors. Robotic manipulators have become indispensable in industries such as manufacturing, healthcare, and logistics, where they perform intricate tasks with precision and efficiency. However, the robustness and reliability of these systems are continually challenged by the possibility of faults or anomalies that can arise due to various factors, including wear and tear, sensor errors, or unexpected external disturbances [7]. The core motivation for this research lies in the critical importance of maintaining the operational integrity of robotic manipulators. Any fault or malfunction in these systems can lead to costly downtime, production delays, or even safety hazards for human operators working alongside the robots. As industries increasingly rely on automation for efficiency and productivity, the need to detect, classify, and respond to faults in real-time becomes paramount.

2. LITERATURE SURVEY

Adam, et al. [11] proposed a two-dimensional Convolutional Neural Network (2DCNN) diagnostic model to effectively improve the diagnostic accuracy of predicting faults on a tabular dataset with multiple fault classes. A simple sliding window approach is proposed to effectively transform and reshape the features in the dataset as inputs to the proposed model architecture. The method's feasibility was evaluated using data from an industrial robotic fuse quality test bench. Anđelić, et al. [12] proposed a Classification of Faults in Operation of a Robotic Manipulator Using a Symbolic Classifier. The application of a genetic programming algorithm (symbolic classifier) with a random selection of hyperparameter values and trained using a 5-fold cross-validation process is proposed to determine expressions for fault detection during robotic manipulator operation, using a dataset that was made publicly available by the original researchers. The original dataset was reduced to a binary dataset (fault vs. normal operation); however, due to the class imbalance random oversampling, and SMOTE methods were applied. Alfarizi, et al. [13] proposed an integrated fault diagnosis system based on extreme gradient boosting for an automated fuse test bench to solve those challenges. The proposed diagnosis system is then validated using the dataset from PHM 2021 Data Challenge. Performance comparison of the fault diagnosis system with other standard approaches in practice is also carried out. Experimental results show that the diagnostic accuracy of the proposed system outperforms several standard fault diagnostic approaches.

Glučina, et al. [14] proposed Detection and Classification of Printed Circuit Boards Using YOLO Algorithm. the performance, memory size, and prediction of the YOLO version 5 (YOLOv5) semantic segmentation algorithm are tested for the needs of detection, classification, and segmentation of PCB microcontrollers. YOLOv5 was trained on 13 classes of PCB images from a publicly available dataset that was modified and consists of 1300 images. The training was performed using different structures of YOLOv5 neural networks, while nano, small, medium, and large neural networks were used to select the optimal network for the given challenge. Cohen, et al. [15] proposed a new data-driven augmented intelligence framework called EveSyncIAI that uses a Timed Petri Net of the normal process to enable selective extraction of discriminating time delay features, which are then used alongside machine learning for fault diagnosis. The developed methodology is applied to a case study of an operational discrete manufacturing system in the semiconductor industry. Multiclass machine learning approaches as well as binary anomaly detection algorithms with varying levels of supervision are directly compared with and without EveSyncIAI-extracted features.

Park, et al. [16] proposed A fault diagnosis algorithm using a deep neural network for an octocopter Unmanned Aerial Vehicle (UAV). The latest angle time history is fed to the proposed algorithm to determine rotor failure in real-time. The normal case and fault case of each rotor are considered with appropriate output pairs to form a dataset. The proposed classifier can distinguish a failed rotor from the others with the help of different patterns of Euler angles during the training process. S. Naddaf-Sh, et al. [17] proposed a machine learning-based method that uses experimental data to detect and classify defective stud welds. the source of data is a series of experimental arc stud welds conducted by a precision robotic arm. The welding data log is recorded through various sensors to shape a multivariate dataset for arc stud welding. Then, wavelet transforms as well as synthetic signal generation are applied for feature extraction. For each set of extracted features, suitable models are designed, trained, and evaluated for automatic root cause classification. Next, the models and approaches are compared on synthetic measurement generation, feature extractions, and feature importance. Numerical results show that XGBoost, the best model, has an 85.41% F1-score on the test.

3. PROPOSED METHODOLOGY

3.1 Overview

This project's procedure encompasses data preprocessing to prepare the dataset, the application of an existing KNN algorithm for fault detection and classification, and the introduction of a proposed approach using XGBoost for the same task. The goal is to develop an effective fault detection and classification system for robotic manipulator tasks, comparing the performance of the existing and proposed methods. This work has implications in robotics and automation, where accurate fault detection is critical for ensuring the safety and efficiency of robotic operations. Figure 4.1 shows the proposed system model. The detailed operation illustrated as follows:

step 1. Preprocessing: The project begins with data preprocessing, where the raw data collected from robotic manipulator tasks is cleaned and transformed to prepare it for analysis. Preprocessing steps may include:

- Data cleaning to handle missing or erroneous data points.
- Feature extraction to identify relevant features or characteristics of the manipulator tasks.
- Data normalization or scaling to ensure that all features are on a consistent scale.
- Encoding categorical variables if any exist in the dataset.

step 2. Existing KNN (K-Nearest Neighbors): In this step, an existing K-Nearest Neighbors (KNN) algorithm is applied to the preprocessed data for fault detection and classification. KNN is a supervised machine learning algorithm used for classification tasks.

- The KNN algorithm assigns a class label to each data point based on the majority class among its k-nearest neighbors, where k is a hyperparameter.
- The performance of the existing KNN algorithm is evaluated in terms of metrics such as accuracy, precision, recall, and F1-score.

step 3. Proposed XGBoost: Following the evaluation of the existing KNN algorithm, a proposed approach using XGBoost is introduced for fault detection and classification in robotic manipulator tasks.

- XGBoost is a powerful gradient boosting algorithm known for its efficiency and performance in classification tasks. It works by building an ensemble of decision trees to make predictions.
- The project likely involves fine-tuning hyperparameters of the XGBoost model to optimize its performance.
- The performance of the proposed XGBoost model is evaluated using the same metrics as the existing KNN algorithm, allowing for a direct comparison of their effectiveness.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

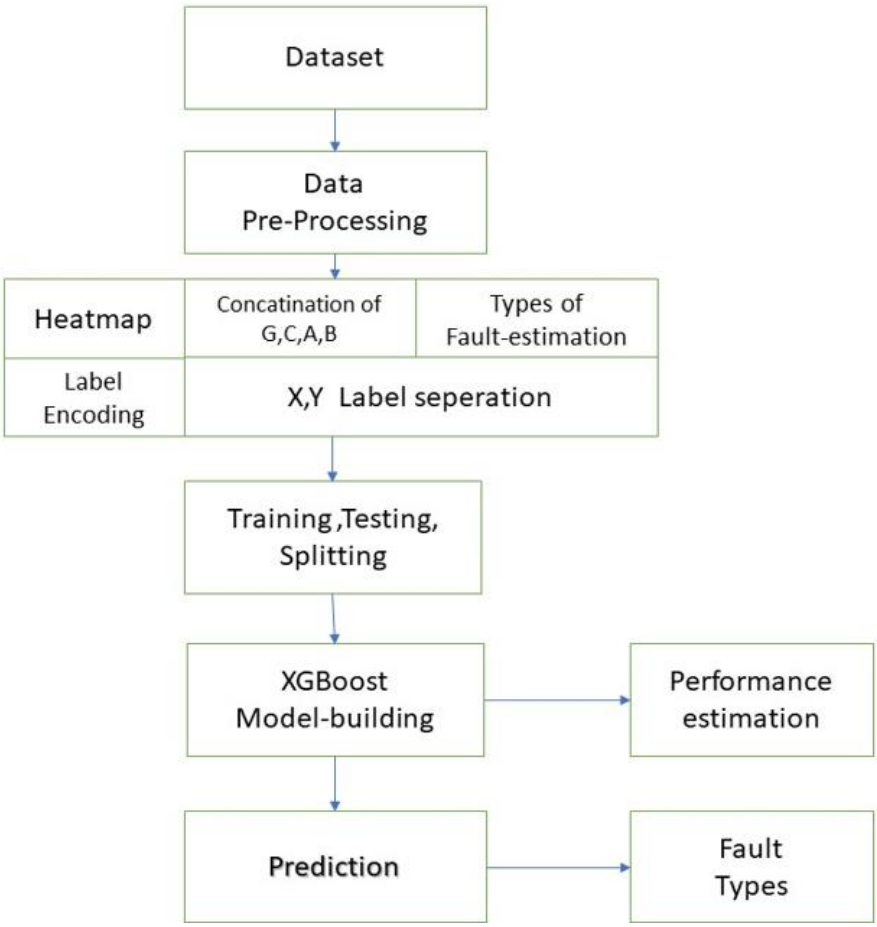

Figure 1. Proposed System model.

3.2 Data Preprocessing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task. A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

Step 1: Data Description: Generates summary statistics for all columns in the dataset. This step provides statistics such as count, mean, standard deviation, minimum, and maximum values for both numeric and categorical columns, giving an overview of the dataset's characteristics.

Step 2: Missing Value Analysis: Visualizes missing values using a matrix plot and displays the total count of missing values for each column. These steps help identify and visualize missing data in the dataset, which is essential for data cleaning and imputation.

Step 3: Data Types: Inspects the data types of columns and converts object-type columns to the category data type. Converting object-type columns to the category data type can reduce memory usage and improve analysis efficiency.

Step 4: Data Preprocessing: Preprocesses the dataset by extracting and transforming date components, removing '$' from ticker values, and converting data types. These steps prepare the data for analysis by making it more suitable for visualization and modeling.

Step 5: Feature Engineering: Removes the 'date' column from the dataset. Feature engineering involves selecting, transforming, or removing features to prepare the dataset for modeling.

Step 6: Final Data Check: Checks for missing values and inspects unique values in specific columns. These final checks ensure that the data is in a suitable state for further analysis and modeling.

3.3 Dataset Splitting

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

3.4 XGBoost Model

XGBoost is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "XGBoost is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the XGBoost takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. XGBoost, which stands for "Extreme Gradient Boosting," is a popular and powerful machine learning algorithm used for both classification and regression tasks. It is known for its high predictive accuracy and efficiency, and it has won numerous data science competitions and is widely used in industry and academia. Here are some key characteristics and concepts related to the XGBoost algorithm:

- **Gradient Boosting:** XGBoost is an ensemble learning method based on the gradient boosting framework. It builds a predictive model by combining the predictions of multiple weak learners (typically decision trees) into a single, stronger model.
- **Tree-based Models:** Decision trees are the weak learners used in XGBoost. These are shallow trees, often referred to as "stumps" or "shallow trees," which helps prevent overfitting.
- **Objective Function:** XGBoost uses a specific objective function that needs to be optimized during training. The objective function consists of two parts: a loss function that quantifies the error between predicted and actual values and a regularization term to control model complexity and prevent overfitting. The most common loss functions are for regression (e.g., Mean Squared Error) and classification (e.g., Log Loss).
- **Gradient Descent Optimization:** XGBoost optimizes the objective function using gradient descent. It calculates the gradients of the objective function with respect to the model's predictions and updates the model iteratively to minimize the loss.
- **Regularization:** XGBoost provides several regularization techniques, such as L1 (Lasso) and L2 (Ridge) regularization, to control overfitting. These regularization terms are added to the objective function.
- **Parallel and Distributed Computing:** XGBoost is designed to be highly efficient. It can take advantage of parallel processing and distributed computing to train models quickly, making it suitable for large datasets.

- **Handling Missing Data:** XGBoost has built-in capabilities to handle missing data without requiring imputation. It does this by finding the optimal split for missing values during tree construction.
- **Feature Importance:** XGBoost provides a way to measure the importance of each feature in the model. This can help in feature selection and understanding which features contribute the most to the predictions.
- **Early Stopping:** To prevent overfitting, XGBoost supports early stopping, which allows training to stop when the model's performance on a validation dataset starts to degrade.
- **Scalability:** XGBoost is versatile and can be applied to a wide range of machine learning tasks, including classification, regression, ranking, and more.
- **Python and R Libraries:** XGBoost is available through libraries in Python (e.g., **xgboost**) and R (e.g., **xgboost**), making it accessible and easy to use for data scientists and machine learning practitioners.

XGBoost, which stands for eXtreme Gradient Boosting, is a popular machine learning algorithm that is particularly effective for structured/tabular data and is often used for tasks like classification, regression, and ranking. It is an ensemble learning technique based on decision trees. Here's how XGBoost operates:

- **Ensemble Learning**: XGBoost is an ensemble learning method, which means it combines the predictions from multiple machine learning models to make more accurate predictions than any single model. It uses an ensemble of decision trees, known as "boosted trees."
- **Boosting**: Boosting is a sequential technique in which multiple weak learners (usually decision trees with limited depth) are trained one after the other. Each new tree tries to correct the errors made by the previous ones.
- **Gradient Boosting**: XGBoost is a gradient boosting algorithm. It minimizes a loss function by adding weak models (trees) that minimize the gradient of the loss function at each stage. This is done by fitting a tree to the residuals (the differences between the predicted and actual values) of the previous model.
- **Regularization**: XGBoost includes L1 (Lasso regression) and L2 (Ridge regression) regularization terms in the objective function to prevent overfitting. These regularization terms help control the complexity of individual trees and reduce the risk of overfitting the training data.
- **Tree Pruning**: XGBoost uses a technique called "pruning" to remove branches of the trees that do not contribute significantly to the model's predictive power. This reduces the complexity of the trees and helps prevent overfitting.
- **Feature Importance**: XGBoost provides a feature importance score, which helps you understand the contribution of each feature (input variable) in making predictions. You can use this information for feature selection and interpretation.

3.5 Advantages of proposed system

The advantages of a proposed system can vary depending on the specific context and goals of the system. However, here are some common advantages that a well-designed and implemented system can offer:

- Improved Efficiency: A well-designed system can automate tasks and processes, reducing manual effort and improving efficiency. This can lead to cost savings and increased productivity.
- Enhanced Accuracy: Automation reduces the likelihood of human errors, leading to more accurate results and data.
- Scalability: A good system can grow and adapt to changing needs and increased workload. It can handle larger datasets or user loads without a significant drop in performance.
- Consistency: Automated systems can consistently apply rules and processes, ensuring uniformity in operations and reducing the risk of inconsistency or bias.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

- Reduced Costs: Automation can lead to cost savings by reducing labor costs and minimizing errors that can be expensive to correct.
- Improved Data Management: A well-implemented system can help with data storage, retrieval, and management, making it easier to access and analyze data when needed.
- Enhanced Security: Systems can incorporate security measures to protect data and prevent unauthorized access, reducing the risk of data breaches and other security incidents.
- Real-time Monitoring and Reporting: Many systems offer real-time monitoring and reporting capabilities, allowing users to make informed decisions quickly.
- Better Customer Service: Systems can improve customer service by providing faster response times, better access to information, and more personalized interactions.
- Data Analysis and Insights: Systems can assist in data analysis and provide insights that help in decision-making and strategic planning.

4. RESULTS AND DISCUSSION

	G	C	B	A	Ia	Ib	Ic	Va	Vb	Vc
0	1	0	0	1	-151.291812	-9.677452	85.800162	0.400750	-0.132935	-0.267815
1	1	0	0	1	-336.186183	-76.283262	18.328897	0.312732	-0.123633	-0.189099
2	1	0	0	1	-502.891583	-174.648023	-80.924663	0.265728	-0.114301	-0.151428
3	1	0	0	1	-593.941905	-217.703359	-124.891924	0.235511	-0.104940	-0.130570
4	1	0	0	1	-643.663617	-224.159427	-132.282815	0.209537	-0.095554	-0.113983
...	...	...	...	...	...	...	...	...	...	...
7856	0	0	0	0	-66.237921	38.457041	24.912239	0.094421	-0.552019	0.457598
7857	0	0	0	0	-65.849493	37.465454	25.515675	0.103778	-0.555186	0.451407
7858	0	0	0	0	-65.446698	36.472055	26.106554	0.113107	-0.558211	0.445104
7859	0	0	0	0	-65.029633	35.477088	26.684731	0.122404	-0.561094	0.438690
7860	0	0	0	0	-64.598401	34.480799	27.250065	0.131669	-0.563835	0.432166

7861 rows × 10 columns

Figure 2: Sample dataset for Fault Detection and Classification in Robotic

KNeighborsClassifier Classification Report:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	469
1	0.96	0.96	0.96	197
2	0.34	0.35	0.34	223
3	0.92	0.99	0.95	221
4	0.93	0.93	0.93	219
5	0.35	0.31	0.33	244
accuracy			0.78	1573
macro avg	0.75	0.76	0.75	1573
weighted avg	0.78	0.78	0.78	1573

Figure 3: Classification report of KNN classifier

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

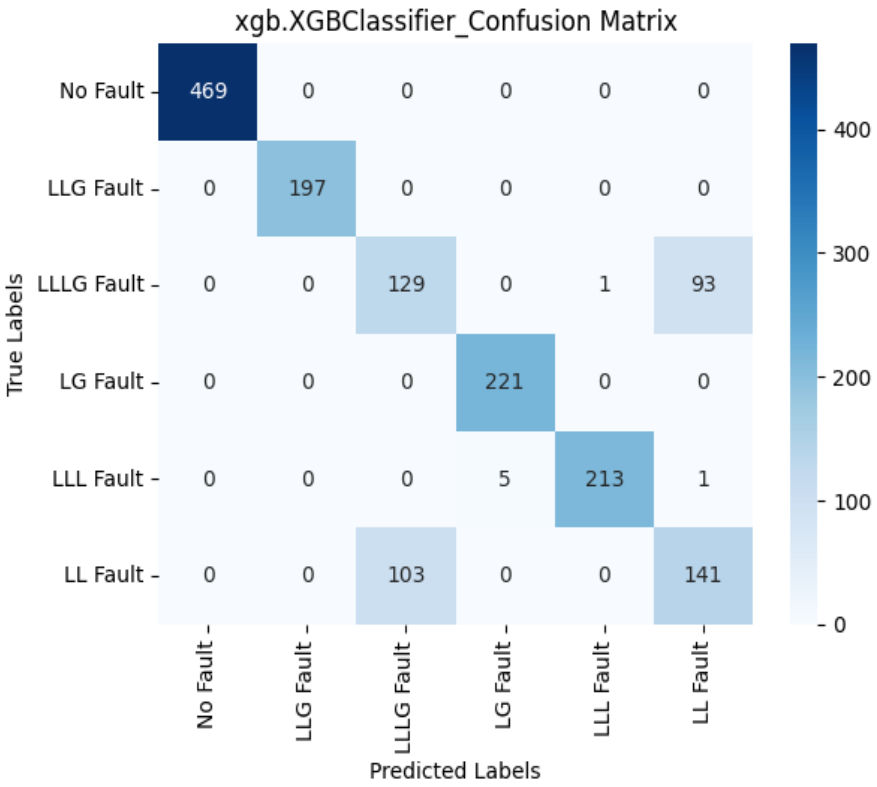

Figure 4: Confusion matrix of XGBoost classifier

XGBClassifier Classification Report:

	precision	recall	f1-score	support
No Fault	1.00	1.00	1.00	469
LLG Fault	1.00	1.00	1.00	197
LLLG Fault	0.56	0.58	0.57	223
LG Fault	0.98	1.00	0.99	221
LLL Fault	1.00	0.97	0.98	219
LL Fault	0.60	0.58	0.59	244
accuracy			0.87	1573
macro avg	0.85	0.85	0.85	1573
weighted avg	0.87	0.87	0.87	1573

Figure 5: Classification report of XGBoost classifier

Table 1: Performance comparison of quality metrics obtained using linear regressor (LR) model and random forest regressor (RFR) model.

Model	Accuracy	Precision	Recall	F1 Score
KNN Classifier	78	78	78	78
XGBoost Classifier	87	87	87	87

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

```
#dataset = pd.read_csv(filename)
A='No Fault'
B='LLG Fault'
C='LLLG Fault'
D='LG Fault'
E='LLL Fault'
F='LL Fault'
predict = xgb_model.predict(X_test.iloc[1:20,:])
for i in range(len(predict)):
    if predict[i] == 0:
        print("{} :{} ".format(multi_data.iloc[i,:],A))
    elif predict[i]== 1:
        print("{} :{} ".format(multi_data.iloc[i, :],B))
    elif predict[i]== 2:
        print("{} :{} ".format(multi_data.iloc[i, :],C))
    elif predict[i]==3:
        print("{} :{} ".format(multi_data.iloc[i, :],D))
    elif predict[i]== 4:
        print("{} :{} ".format(multi_data.iloc[i, :],E))
    else:
        print("{} :{} ".format(multi_data.iloc[i,:],F))
```

```
faultType        1001
Name: 0, dtype: object :LLLG Fault
G                   1
C                   0
B                   0
A                   1
Ia           -336.186
Ib           -76.2833
Ic            18.3289
Va           0.312732
Vb          -0.123633
Vc          -0.189099
faultType        1001
Name: 1, dtype: object :No Fault
```

Figure 6: Prediction results using XGBoost classifier.

5. Conclusion

The project, "Fault Detection and Classification in Robotic Manipulator Tasks using Multi-class Classifications," has demonstrated a structured workflow for addressing fault detection challenges in the context of robotic manipulator tasks. Commencing with data preprocessing to refine and format the raw data, the project implemented both an existing KNN algorithm and a proposed XGBoost approach for fault detection and classification. The existing KNN algorithm leveraged proximity-based classification, while the proposed XGBoost model harnessed gradient boosting techniques for enhanced performance. Through comprehensive evaluation using metrics like accuracy, precision, recall, and F1-score, the project assessed the effectiveness of both approaches. This project contributes significantly to the field of robotics by offering a systematic methodology for detecting and classifying faults in robotic manipulator tasks, enhancing the reliability and safety of such operations.

REFERENCES

[1] Liu, Ying, et al. "An event-driven Spike-DBN model for fault diagnosis using reward-STDP." ISA Transactions (2023).
[2] Lin, Haoyu, et al. "Collision Localization and Classification on the End-Effector of a Cable-Driven Manipulator Applied to EV Auto-Charging Based on DCNN–SVM." Sensors 22.9 (2022): 3439.
[3] Zhou, Zhongxiang, et al. "Open-Set Object Detection Using Classification-Free Object Proposal and Instance-Level Contrastive Learning." IEEE Robotics and Automation Letters 8.3 (2023): 1691-1698.
[4] Miao, Zhaoming, et al. "Multi-heterogeneous sensor data fusion method via convolutional neural network for fault diagnosis of wheeled mobile robot." Applied Soft Computing 129 (2022): 109554.
[5] Rakshit, Arnab, et al. "Autonomous grasping of 3-D objects by a vision-actuated robot arm using Brain–Computer Interface." Biomedical Signal Processing and Control 84 (2023): 104765.
[6] Kozák, Viktor, et al. "Towards Visual Classification Under Class Ambiguity." 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023.
[7] Kæseler, Rasmus L., et al. "Feature and classification analysis for detection and classification of tongue movements from single-trial pre-movement EEG." IEEE Transactions on Neural Systems and Rehabilitation Engineering 30 (2022): 678-687.

[8] Tang, Wenshuo, et al. "Characterisation of composite materials for wind turbines using frequency modulated continuous wave sensing." Journal of Composites Science 7.2 (2023): 75.
[9] Vijayanti, Veernala, Akankshya Kar, and K. N. S. S. Navya. "Analysis of Deep Learning Based Garbage Detection in Water Bodies." 2023 5th International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, 2023.
[10] Andrussow, Iris, et al. "Minsight: A Fingertip-Sized Vision-Based Tactile Sensor for Robotic Manipulation." Advanced Intelligent Systems (2023): 2300042.
[11] Adam, Hosameldin Eltayeb A., James K. Kimotho, and Jackson G. Njiri. "Multiple faults diagnosis for an industrial robot fuse quality test bench using deep-learning." Results in Engineering 17 (2023): 101007.
[12] Anđelić, Nikola, et al. "Classification of Faults Operation of a Robotic Manipulator Using Symbolic Classifier." Applied Sciences 13.3 (2023): 1962.
[13] Alfarizi, Muhammad Gibran, Jrn Vatn, and Shen Yin. "An extreme gradient boosting aided fault diagnosis approach: A case study of fuse test bench." IEEE Transactions on Artificial Intelligence (2022).
[14] Glučina, Matko, et al. "Detection and Classification of Printed Circuit Boards Using YOLO Algorithm." Electronics 12.3 (2023): 667.
[15] Cohen, Joseph, Baoyang Jiang, and Jun Ni. "EveSyncIAI: Event synchronization industrial augmented intelligence for fault diagnosis." IEEE Transactions on Semiconductor Manufacturing 35.3 (2022): 446-456.
[16] Park, Jongho, Yeondeuk Jung, and Jong-Han Kim. "Multiclass classification fault diagnosis of multirotor UAVs utilizing a deep neural network." International Journal of Control, Automation and Systems 20.4 (2022): 1316-1326.