Quantum inspired Machine Learning for failure Prediction in IoT machines

Vaishno Lukin Bussa
School of Computer Science Engineering
Vellore Instituite of Technology
Chennai, India
bussa.luckin@gmail.com

Pragna Tummala
School of Electronics Engineering
Vellore Instituite of Technology
Chennai, India
pragnatummala.36@gmail.com

Dr. Guga priya G
School of Electronics Engineering
Vellore Instituite of Technology
Chennai, India
gugapriya.g@vit.ac.in

***Abstract*— Industrial operations utilizing Internet of Things (IoT) technology now require effective fault detection systems due to its increased popularity. The research investigates quantum-inspired machine learning as an approach for detecting faults in Industrial IoT machines. The QML system trains through genuine sensor information based on measurements of air temperature alongside process temperature along with rotational speed and torque together with tool wear information to perform future fault predictions. This system runs through a web-based application developed with Django as the backend and HTML as the frontend to display machine status data directly retrieved from the Firebase database. Accurate fault detection through the proposed method helps achieve predictive maintenance which subsequently enhances the reliability of industrial machinery systems.**

***Index Terms*—Machine Learning, Fault Detection, Predictive Maintenance, Firebase, Real-time Monitoring, Quantum Neural Networks, Machine failure**

I. Introduction

Industrial Internet of Things (IIoT) has transformed every aspect of industry operational management through its ascension. Companies achieve better decision making at unprecedented speed because their interconnected machines generate non-stop data streams that provide enhanced equipment visibility. Real-time monitoring of systems along with prediction of maintenance needs before breakdowns occur represents the most critical factor for factory operators and heavy machinery managers. The success of equipment operations depends on this method to avoid system breakdowns while ensuring maximum performance. The traditional methods of detecting equipment issues often fail to provide useful predictions about equipment failures. IIoT setups generate substantial quantities of data which overwhelms monitoring systems as well as operational conditions that rapidly transform failing to detect issues in a timely fashion.

This research solution offers its contribution. I have analyzed how quantum-inspired machine learning known as QML presents a new method to monitor IIoT machine failures. QML incorporates several unusual concepts from quantum computing especially quantum entanglement together with superposition which elevates traditional ML methods to advanced levels. Research focuses on quantum neural networks because they serve as an outstanding piece of the quantum computing breakthroughs. The design of these neural networks matches previous classic neural networks with a quantum transformation that enables effective pattern finding capabilities. I want to validate whether predictive maintenance will reach its next advancement by combining QML with QNN technology to enhance its performance for modern industrial needs.

I designed this study's primary component which uses five essential factors to create a fault detection model that includes air temperature and process temperature alongside rotational speed and torque measurements and measuring tool wear. The machine's vital health indicators which reveal environmental response and operational lifespan deterioration serve as non-arbitrary measurement points. The sensors I implemented measure air temperature and process temperature in real time through sensors to support precise and current accuracy. Users need to enter three metrics manually which include rotational speed, torque and tool wear. I am planning to address this inadequacy in the system through improvements that will occur in the future. Firebase acts as the cloud database where all data stored in it gets organized from both sensor inputs and user manual entries.

A combination of the Django backend and HTML frontend enables users to access the practical system. The technical background of the system runs the data management operations but a basic login page lets users enter their measurements for viewing temperature readings and fault predictions from the QML model. The Django-Firebase solution delivers smooth performance to the system while quantum neural networks from QML help detect even faint warning patterns which traditional methods might miss. The application of quantum-inspired tools to analyze data has been exciting because it produces predictions that demonstrate advanced understanding of the information.

The future development for this project includes important modifications which I aim to implement. My main objective is to automate the data collection of air temperature along with process temperature and speed and torque indicators and wear measurements through sensor technology without requiring human input. When these five parameters operate automatically with sensors the monitoring system will become more precise while creating autonomy for self-diagnosed issues before system breakdowns occur. A successful implementation of automatic parameter collection through sensors would redefine our practices concerning maintenance operations in IIoT systems. The research begins a process to assess QML combined with QNN performance in addressing industrial machinery unpredictability which could lead to improved factory reliability in future generations.

II. Methodology

The objective of this project is to implement an IoT system based on predictive failure of a machine using a quantum neural network, real-time sensors, cloud storage, and an AI chatbot. Sensor readings are preprocessed and cleaned prior to feeding it into the Quantum Neural Network (QNN) for training and this enhances the performance in predictions since fine-grained patterns are captured.

The system is capable of interchanging data seamlessly with sensors, inputs from users, and forecasts. With AI, cloud storage, and quantum computing added into the system, it provides an efficient means to identify and escape machine failure in real time.

A. Data preprocessing

The data set is 10,000 rows by 10 columns, capturing machine operating conditions and failure information. It encompasses single-use identifiers (UDI, Product ID), machine model, and readings of sensors such as air temperature, process temperature, rotational speed, torque, and tool wear. "Target" is a failure occurrence flag and "Failure_Type" determines the type of failure, i.e., "No Failure" or some types of failure. This information is used in IoT machine failure prediction by the study of sensor data patterns. Data preprocessing is an important process to ensure that the sensor data gathered is clean, organized, and ready for training the Quantum Neural Network (QNN). Raw sensor reading from the DHT11 temperature and humidity sensor usually includes noise, missing value, and outliers, and these must be treated before inputting the data into the model. The preprocessing starts with data cleaning in which missing values are either deleted or filled by statistical means. Outliers are detected via Z-score analysis and handled in turn to prevent skewing the predictions Feature selection methods are used to determine critical parameters like air temperature, process temperature, and rotational speed, which are directly related to machine failures. The chosen characteristics are normalized with MinMaxScaler to guarantee all values lie between a uniform range to enhance model efficiency. The data is divided into training and testing sets so that the model can generalize effectively to new input.

B. Quantum neural network implementation

It uses a Quantum Neural Network (QNN) to improve accuracy and make better predictions. The three base layers of the QNN are a Quantum Embedding Layer, a Variational Quantum Circuit (VQC), and a Classical Output Layer. Quantum Embedding Layer places input sensor data into quantum states in such a way that the system exploits the parallel computing ability of quantum computers to speed up the calculations. The VQC is based on quantum transformations to collect subtle failure patterns which are potentially evasive of classical networks. Finally, the Classical Output Layer uses softmax activation to map machine failures into given categories such as "Heat Dissipation Failure" and "Power Failure." The training is done through Adam optimizer and Mean Squared Error (MSE) loss function. Quantum computing utilizes PennyLane library, and the training allows the network to make precise predictions regarding failures.

This research project analyzes quantum computing methodology adoption in deep learning through design of a hybrid Quantum Neural Network (QNN) for classification functions. A 5-qubit parameterized quantum circuit together with Angle Embedding and Strongly Entangling Layers as main components enables this model. The QNN has been constructed through the combination of PennyLane with PyTorch and it acts as a layer between classical parts of the neural network. The preprocessing phase begins with split and categorical encoding before scaling features based on which the data is divided into 90% for training and 10% for testing. The training process consisted of twenty epochs through which Adam optimizer enabled a 96.5% accurate prediction of test samples. Performance evaluation reveals the model's effectiveness through the assessment of classification reports together with confusion matrices. Additionally, the training time of 16.9 seconds highlights the computational feasibility of hybrid quantum-classical models. This research establishes quantum-enhanced machine learning as a promising technology by identifying future work to improve qubit numbers and develop optimized encoding techniques and deploy real quantum hardware.

Comparing with the Traditional Neural Network approach, A preprocessing step includes both feature scaling and categorical encoding followed by division of the dataset into training and testing parts. The ANN design consists of dense layers starting with 128 neurons in the input layer using ReLU activation followed by a hidden layer containing 64 ReLU neurons before finishing with an output layer having sigmoid activation for binary output. The training process lasted 20 epochs while using the Adam optimizer together with binary cross-entropy loss function and a batch size of 32. During the testing phase the model has reached 94.7% accuracy while operating within 30.03 seconds. The model's evaluation by confusion matrices together with classification reports reveals its success in identifying different failure categories.

The findings from Quantum Neural Network (QNN) show enhanced computational efficiency when it achieves 96.5% accuracy during training which lasts for 16.9 seconds. The optimal character of QNNs emerges from their ability to leverage quantum mechanics in performing swift and accurate decision-making processes.

C. Frontend design for web app

The frontend of the web application is designed to be user-friendly and responsive, and it offers a simple interface for users to track machine conditions. It is built with HTML, CSS, JavaScript, and Bootstrap to provide ease of navigation on any device Real-time sensor readings are made possible using Chart.js to enable observation of trends over temperature, speed, and other parameters of a machine. Features of authentication such as login, registration, and password reset functionality have been implemented using Django's own internal authentication system. A reports module is also included in the web app that allows users to see past failure history and analysis. A chatbot feature is implemented in the frontend where users are enabled to pose queries regarding machine failure and get responses immediately.

MODEL	ACCURACY	TRAINING TIME
Neural Networks	96.5	30.03 s
Quantum Neural Networks	96.52	16.98 s

Table 1: Comparison Between Two models

D. Backend design for web app

The backend of the web application is written with the Django framework and is responsible for authentication, data storage, and machine failure prediction. User login, registration, and resetting the password is handled by Django's authentication system. The trained and saved QNN model in a file called Model1.pkl is loaded by joblib in the Django backend. On users entering real-time machine parameters, backend does calculations over these parameters and returns a prediction of failure. The results are stored in the UserPredictModel database table to be analyzed at a later date. A separate API endpoint is developed to couple the chatbot model, returning responses to questions posed by the user regarding machine breakdowns and preventive measures.

1) Hardware Implementation

Hardware configuration is an ESP8266 NodeMCU microcontroller, gathers live temperature and humidity using DHT11 sensors. DHT11 sensors are connected to the ESP8266 on pins D3 and D4 and keep reading machine conditions. The read values are presented on an SSD1306 OLED display, which gives real-time visual output. The ESP8266 speaks on Wi-Fi and pushes sensor reading to Firebase Realtime Database for real-time storage and retrieval. The process is executed in a loop by the microcontroller, updating the OLED display and Firebase at regular intervals to ensure constant monitoring of machine conditions.

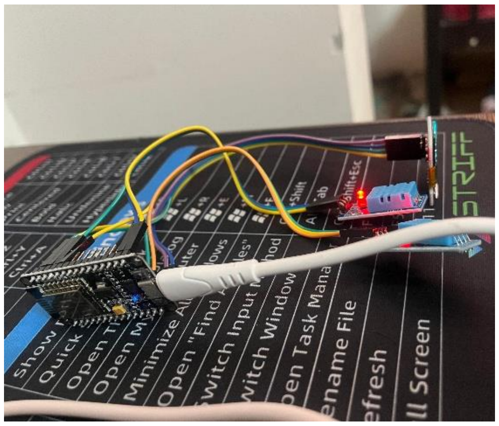


Fig. 1. Hardware setup

2) Firebase Integration

Firebase Realtime Database is also incorporated into the system to support real-time synchronization of data and cloud storage. Being a NoSQL cloud-based database, Firebase supports real-time updates to all connected clients such that sensor data is always updated. The recorded temperature and humidity values are kept in database paths like "Python/Temp1" and "Python/Temp2" to be easily accessed via the web application. Firebase accommodates offline synchronization, that is, data still gets recorded even during network disconnection. It is also scalable, thus it is possible to store extra machine parameters without interfering with the performance.

3) Integration with hardware part

The communication between the ESP8266 microcontroller and Firebase is via the Arduino IDE and C++ programming. Wi-Fi credentials (password and SSID) are configured in the ESP8266 code to create a stable link. Firebase authentication is done through an API key and database URL to verify data transmission. Sensor readings are sent as JSON objects before transmission to Firebase to achieve efficient storage and retrieval. The microcontroller operates in a loop indefinitely, sometimes reading the sensors, posting to Firebase, and displaying the latest values on the OLED screen. The integration guarantees smooth communication between the hardware units and the cloud storage system.

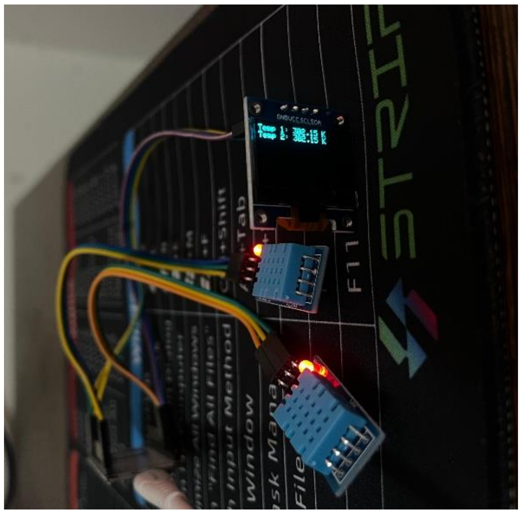
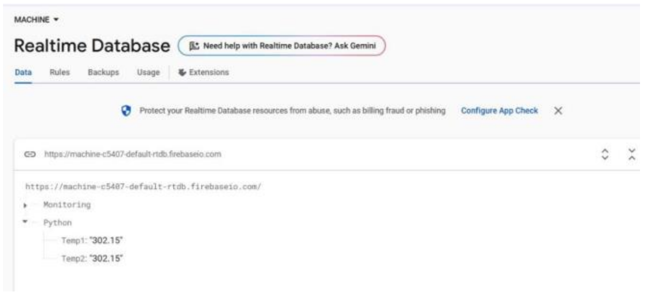


Fig. 2. Hardware readings in oled

E. *Firebase Integration with Django for web app*

To retrieve the latest sensor data in the Django web application, Firebase is linked via Firebase Admin SDK. Django's backend establishes a connection with Firebase by importing service account credentials stored in a JSON file. The fetch_firebase_data() function is programmed to invoke Firebase and retrieve the latest sensor readings. The data gathered is processed and presented on the web dashboard in real time by JavaScript and Django views. This integration enables users to view live machine conditions via the web interface without manual updates of data.


MACHINE
Realtime Database
Need help with Realtime Database? Ask Gemini
Data
Rules
Backups
Usage
Extensions
Protect your Realtime Database resources from abuse, such as billing fraud or phishing
Configure App Check

```
https://machine-c5407-default-rtdb.firebaseio.com

https://machine-c5407-default-rtdb.firebaseio.com/
▸ Monitoring
▾ Python
    Temp1: "302.15"
    Temp2: "302.15"
```

Fig. 3. Real time data obtained

F. *Chatbot model*

A chatbot powered by Artificial Intelligence is incorporated in the system to guide users regarding machine failure and requisite precautions. The chatbot is created on the basis of Natural Language Processing (NLP) techniques and follows a pipeline architecture. First, it loads pre-defined questions related to failure from an intents.json file. User inputs are managed by the chatbot using Natural Language Toolkit (NLTK), performing tokenization and lemmatization of input text for normalizing it. The text is then projected onto pre-defined failure categories with a deep learning-based neural network. The model of the chatbot is structured with three layers: an input layer of 128 ReLU-activated neurons, a hidden layer of 64 neurons, and an output layer using softmax activation for multi-class classification. To have more accurate outcomes, the model is trained on Stochastic Gradient Descent (SGD) and Nesterov momentum. When a user makes a query, the chatbot takes the input, determines the intent, and responds with an appropriate response in the form of JSON data. The interface of the chatbot is integrated very smoothly into the web application, and users can use it in real time.

Here our chatbot page in Our web app will give solutions about types of failures and also give us precautions and solutions for handling the machine failures

G.System Architecture

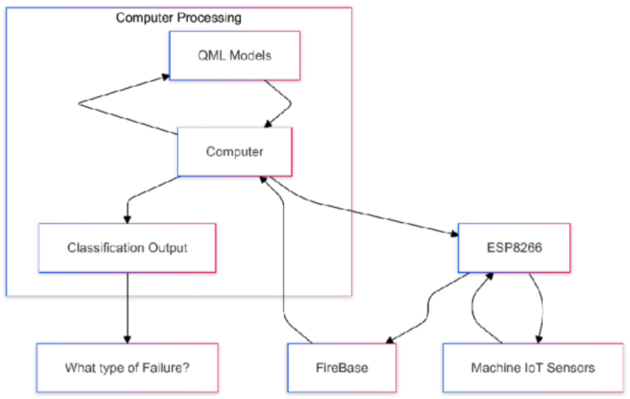


Fig 4: System Architecture

The design above depicts how failure classification systems combine IoT sensors and machine learning models with cloud processing operations.
A web app in computer receives an input from “QML Models” (Quantum Machine Learning Models), the starting point of the computer processing framework. After processing the information through the webapp the system generates a “Classification Output” to identify “What type of Failure?” which specifies the failure detected by the system.

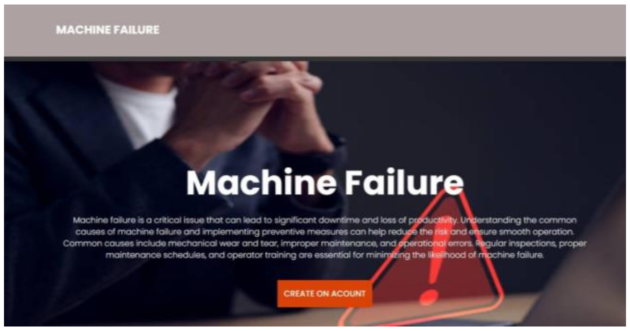
Machine IoT Sensors gather data which the ESP8266 microcontroller transmits to "ESP8266" via its connection to FireBase cloud services. The ESP8266 directs data to "FireBase" for cloud processing that connects to the computer system. The system operates through data arrows that show two-way communication between the ESP8266 and FireBase as well as between FireBase and the computer.

We can directly get the air_temperature and process_temperature from the firebase and other parameters like torque, rotational speed and tool wear should be manually entered by the user combining all these data then our model will predict the type of failure and then it will display the description of failure and symptoms of the failure and also the preventions and recommendations according to the failure detected. And also there is a chatbot in our webpage where it will clarify our doubts regarding the Types of machine failures and precautions for machine failures. Our Chatbot not only displays the answer on screen but also will answer using the voice which we have integrated.

III. Results and Discussion

A. *Website Interface*

1) The front end and backend implementation is done using the command python manage.py runserver and our web page consists of different pages like home, report, profile, prediction and data base page. After executing there will be a link displayed in the terminal we need to follow to that link for our webapp then it will ask us to sign up first and then sign in. If we created our user and logged in there will be display of pages like home, report, prediction, profile, database. So here are the images attached.

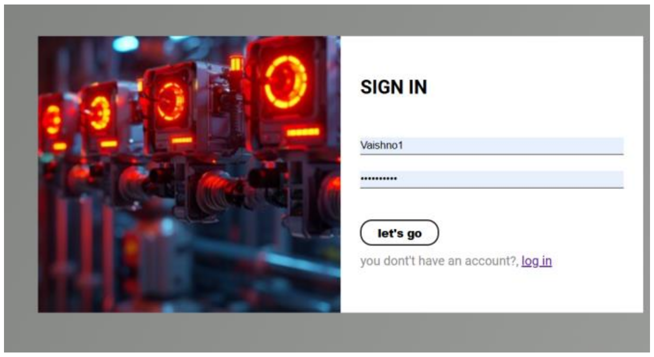

MACHINE FAILURE

Machine Failure

Machine failure is a critical issue that can lead to significant downtime and loss of productivity. Understanding the common causes of machine failure and implementing preventive measures can help reduce the risk and ensure smooth operation. Common causes include mechanical wear and tear, improper maintenance, and operational errors. Regular inspections, proper maintenance schedules, and operator training are essential for minimizing the likelihood of machine failure.

CREATE ON ACOUNT

Fig. 6. Web application


SIGN IN

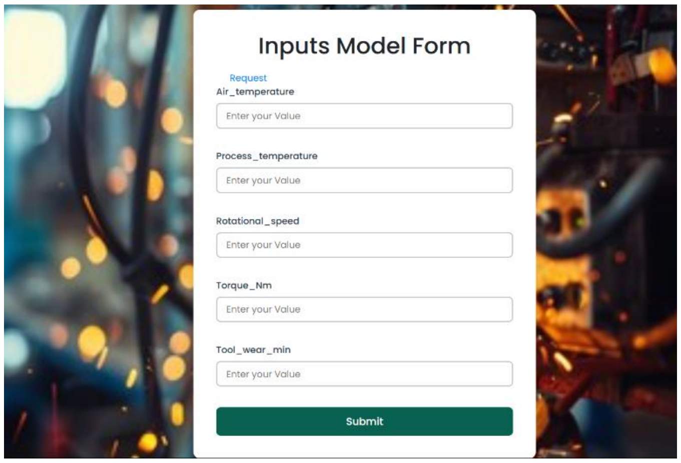
Vaishno1

••••••••••

let's go

you don't have an account?, log in

Fig. 6. Sign in Page

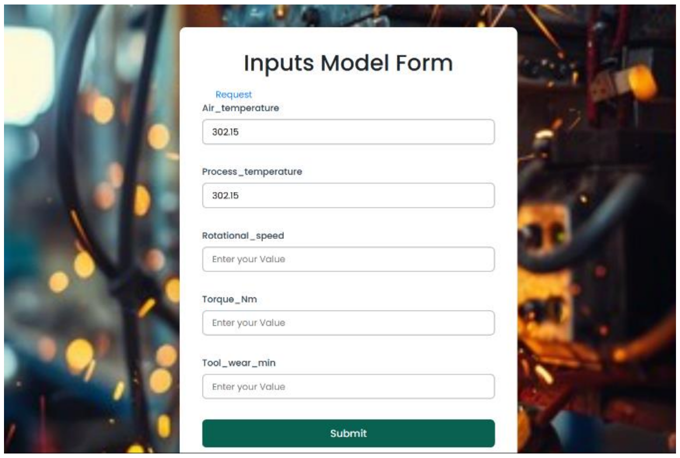

Inputs Model Form

Request

Air_temperature	Enter your Value
Process_temperature	Enter your Value
Rotational_speed	Enter your Value
Torque_Nm	Enter your Value
Tool_wear_min	Enter your Value

Submit

Fig 7 Prediction page

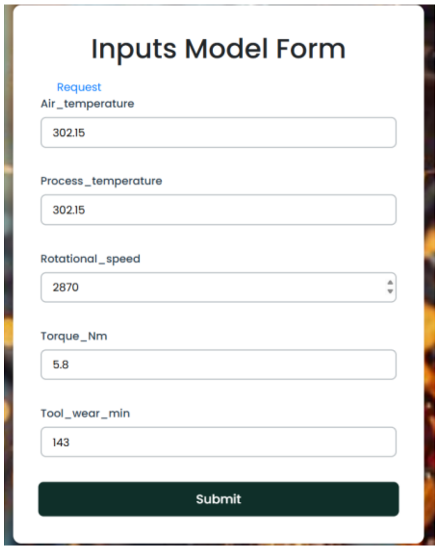

Inputs Model Form

Request

Air_temperature	302.15
Process_temperature	302.15
Rotational_speed	Enter your Value
Torque_Nm	Enter your Value
Tool_wear_min	Enter your Value

Submit

Fig 8 After requesting from Firebase

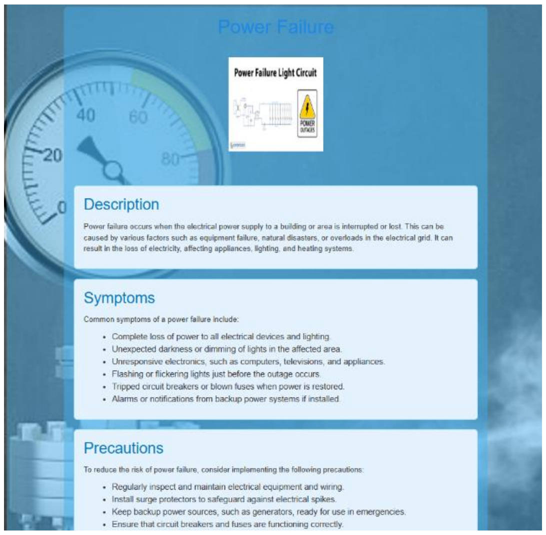

Inputs Model Form

Request

Air_temperature	302.15
Process_temperature	302.15
Rotational_speed	2870
Torque_Nm	5.8
Tool_wear_min	143

Submit

Fig 9 Entering manually



Power Failure

Description

Power failure occurs when the electrical power supply to a building or area is interrupted or lost. This can be caused by various factors such as equipment failure, natural disasters, or overloads in the electrical grid. It can result in the loss of electricity, affecting appliances, lighting, and heating systems.

Symptoms

Common symptoms of a power failure include:

- Complete loss of power to all electrical devices and lighting.
- Unexpected darkness or dimming of lights in the affected area.
- Unresponsive electronics, such as computers, televisions, and appliances.
- Flashing or flickering lights just before the outage occurs.
- Tripped circuit breakers or blown fuses when power is restored.
- Alarms or notifications from backup power systems if installed.

Precautions

To reduce the risk of power failure, consider implementing the following precautions:

- Regularly inspect and maintain electrical equipment and wiring.
- Install surge protectors to safeguard against electrical spikes.
- Keep backup power sources, such as generators, ready for use in emergencies.
- Ensure that circuit breakers and fuses are functioning correctly.

Fig 10 Failure Prediction

IV. CONCLUSION

Quantum Machine Learning (QML) enhances predictive maintenance in IoT networks with IoT sensor feedback, cloud storage, and quantum machine learning operations to discover and predict failures It accurately classifies faults such as random failure, power failure, heat dissipation failure, and tool wear failure efficiently with real-time and accurate fault detection using Firebase for data synchronization. Quantum computing solutions avoid pitfalls of failure analysis in high-dimensional space and computational bottlenecks, maximize operating efficiency, minimize downtime, and maximize plant safety. Within the Industry 4.0 and smart manufacturing backdrop, QML provides a new-generation predictive maintenance solution. A few issues such as a lack of direct access to the quantum hardware, scalability, and interfacing the QML to traditional cloud platforms still remain. Additionally, good-quality, diverse training data is also critical for trustworthy quantum predictions, and hence more research is needed for the industrial use of QML in fault detection in industrial IoT.

V. FUTURE SCOPE

In the future, certain modifications can be made in an attempt to further improve the system. One may research using hybrid quantum-classical architectures in order to harness the capability of the conventional deep learning models and merge them with the strength of a quantum computer to establish the system towards further viability and accessibility. Edge computing platforms may also be included to run the IoT sensor data locally before passing it to the quantum cloud to reduce latency and bandwidth. Integrating self-improving AI features in the model will also allow it to learn dynamically based on newly appearing patterns of failure, making it more accurate in the process.

ACKNOWLEDGMENT

We authors would like to take this moment to thank the School of Electronics Engineering, Vellore Institute of Technology (VIT), for their kind support and motivation during this research. Their technical support, infrastructure, and motivation have been the most important factors in the successful completion of this work. We express our deepest gratitude to our core faculty advisors for their fruitful debates and suggestions, which have helped refine our strategy. Furthermore, we are thankful for the co-operative environment and facilities provided by VIT that played a significant role in carrying out this study.

REFERENCES

[1] AU - Ghule, Vijaykumar, Mahalle, Parikshit, 2024/04/08, Fault Detection in IoT Sensor Networks with XAI-LCS: Explainable AI-driven Diagnosis for Low-Cost Sensor, Journal of Electrical Systems

[2] Tertytchny, Georgios, and Maria K. Michael. "dataset reduction framework for intelligent fault detection in IoT-based cyber-physical systems using machine learning techniques." 2020 International Conference on Omni-layer Intelligent Systems (COINS). IEEE, 2020.

[3] Mahmood, T., Li, J., Pei, Y., Akhtar, F., Butt, S. A., Ditta, A., & Qureshi, S. (2022). An intelligent fault detection approach based on reinforcement learning system in wireless sensor network. The Journal of Supercomputing, 78(3), 3646-3675.

[4] AU - Lallas, E.N., Chis, Adriana, Gonzalez-Velez, Horacio, 2019/05/02, Towards Distributed IoT/Cloud based Fault Detection and Maintenance in Industrial Automation, Procedia Computer Science

[5] Gaddam, Anuroop, et al. "Detecting sensor faults, anomalies and outliers in the internet of things: A survey on the challenges and solutions." Electronics 9.3 (2020): 511.

[6] Cerezo, M., Verdon, G., Huang, H. Y., Cincio, L., & Coles, P. J. (2022). Challenges and opportunities in quantum machine learning. Nature Computational Science, 2(9), 567-576.

[7] de Oliveira, W. R., Silva, A. J., Ludermir, T. B., Leonel, A., Galindo, W. R., & Pereira, J. C. (2008, October). Quantum logical neural networks. In 2008 10th Brazilian Symposium on Neural Networks (pp. 147-152). IEEE.

[8] Hou, Xuan. "Research of model of quantum learning vector quantization neural network." In Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, vol. 8, pp. 3893-3896. IEEE, 2011

[9] Kanawaday, Ameeth, and Aditya Sane. "Machine learning for predictive maintenance of industrial machines using IoT sensor data." 2017 8th IEEE international conference on software engineering and service science (ICSESS). IEEE, 2017.

[10] Kwon, Jung-Hyok, and Eui-Jik Kim. "Failure prediction model using iterative feature selection for industrial internet of things." Symmetry 12.3 (2020): 454.

[11] Gupta, Deepali. "Prediction of sensor faults and outliers in iot devices." 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO). IEEE, 2021.

[12] Bhavana, Kotte, Vinuthna Nekkanti, and N. Jayapandian. "Internet of things enabled device fault prediction system using machine learning." International conference on inventive computation technologies. Cham: Springer International Publishing, 2019.

[13] Abbas, Amira, et al. "The power of quantum neural networks." Nature Computational Science 1.6 (2021): 403-409.

[14] Bala, Indu, Kiran Ahuja, and Maad M. Mijwil. "Quantum Machine Learning for Industry 4.0." Quantum Computing and Artificial Intelligence: The Industry Use Cases (2025): 415-433.

[15] Beer K, Bondarenko D, Farrelly T, Osborne TJ, Salzmann R, Scheiermann D, Wolf R. Training deep quantum neural networks. Nature communications. 2020 Feb 10;11(1):808.