INTERNATIONAL JOURNAL OF RECENT TRENDS IN TECHNOLOGY AND ENGINEERING (IJRTTE)

Journal Home Page: https://ijrtte.com/

TinyML-Powered Edge Intelligence FOR Real-Time Predictive Maintenance of Industrial Motors

Soundararjan.K[1], Manoj Govindaraj[2] and M. Maria Sampoornam[3]

[1]*Professor, Annai Mathammal Sheela Engineering College, Erumapatty, Namakkal District, Pin - 637013, Tamil Nadu, India.*
[2]*Associate Professor, Department of management studies, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai, Tamil Nadu, India.*
[3]*Assistant Professor, Department of Information Technology, J.J.College of Engineering and Technology, Tiruchirappalli, Tamil Nadu, India.*
[1]*drksrfin@gmail.com*, [2]*manoj.nmcc@gmail.com*, [3]*mariasampoornamm@jjcet.ac.in*

Abstract. The increasing complexity of industrial machines and the necessity for continuous production have emphasized the importance of smart, effective, and real-time predictive maintenance systems. This paper proposes an edge-intelligence model based on TinyML to detect, and prediction industrial motor faults with high accuracy and low latency. In contrast to conventional cloud-based solutions, it can port machine learning models onto microcontrollers and edge devices in a lightweight manner such that the inference (on-device) becomes possible with ultra-low power with a much lower dependence on the network. The platform provides for real-time data collection, analysis of vibration and temperature signals, and detection of anomalies, to schedule maintenance when required, before any critical fault occurs. Extensive experiments on benchmark motor datasets show that our system is practical under various operation scenarios by improving the response time, energy consumption, and deployment scalability. This is an approach that not only overcomes the limitations associated with conventional predictive maintenance frameworks when it comes to latency, cost, scalability but can be seen as a viable and sustainable solution to the Issue in many industries 4.0 environments.

Keywords: TinyML, Edge Intelligence, Predictive Maintenance, Industrial Motors, Real-Time Monitoring, On-Device Machine Learning, Condition-Based Maintenance, Industry 4.0, Low-Power AI, Smart Manufacturing.

1. Introduction

1.1 Motivation for Predictive Maintenance in Industry 4.0

The explosive development of Industry 4.0 has driven the incorporation of intelligent solutions (e.g., IoT, cyber-physical systems, intelligent automation) into industrial environments to amplify industrial productivity and operational ability. Within these overall considerations, one can cite predictive maintenance (PdM) as a key factor to minimize equipment downtime, reduce maintenance costs and equipment aging [1], [2]. Especially for industrial motors which are the workhorse of manufacturing, energy and transport sectors, unscheduled failures can have economic and operational impacts. The ability of predicting faults in real-time has therefore become a critical requirement for recognizing abnormalities in time and to execute maintenance actions before these failures detonate [3], [4].

1.2 Limitations of Current Cloud Systems

Although the traditional predictive maintenance systems with the aid of cloud-based analytics offer a great many advantages, they are still confronted with some challenges, such as the large transportation delay, still needing the internet connection, high communication cost, privacy challenge [5], [6]. Such systems

Volume: 04 Issue: 02 Year: 2025
Received: 14.04.2025 Received in revised form: 20.05.2025 Accepted: 23.06.2025
Available Online: 30.06.2025
Published by NTL Publisher

generally require the transmission of streams of sensory data in high amount to centralized servers, adding to the bandwidth overhead, in addition to having a latency in the detection of failures where this is time-critical. Additionally, cloud models are not always practical in remote and bandwidth-limited scenarios, such as underground mining, offshore platforms and rural factories [7]. Therefore, edge computing is increasingly demanded to push the computation closer to the source.

1.3 Contributions of This Paper

To address the aforementioned challenges, this paper presents a new TinyML-driven edge intelligence framework towards real-time predictive maintenance of industrial motors. The main contributions of this work can be summarized as follows:

- Development of lightweight TinyML models suitable for low-power microcontroller-based edge devices.
- Integration of real-time signal processing and fault detection capabilities directly on embedded hardware.
- Deployment and evaluation of the framework in real-world scenarios to demonstrate improvements in latency, power efficiency, and accuracy compared to cloud-based systems.
- Comprehensive benchmarking of the system using key performance metrics including energy consumption, fault detection rate, inference time, and scalability.

2. Related Work

2.1 TinyML in Industrial IoT

The field of Tiny Machine Learning (TinyML) has recently gained a promising direction for deploying Machine Learning (ML) models directly to small embedded devices where the volumetric constraints are critical. Its significance in industrial Internet of Things (IIoT) is increasing from the perspectives of low power, low latency and the network independence [1], [2]. Lin et al. [3] stress that TinyML supports real-time inference on memory-starved microcontrollers, which is very well suited for applications such as smart factories and predictive maintenance systems. Ooko and Karume [4] in particular underscore the fact that TinyML is viable when monitoring the condition of equipment and predicting failure is concerned as one does not have to depend on cloud resources. Additionally, Zaidi et al. [5] states that edge intelligence offered by TinyML is of great importance for industrial equipment working in remote or narrow bandwidth access environments.

2.2. Edge Computing in Predictive Maintenance

Sensor-based condition monitoring and edge computing is more and more utilized for real-time predictive maintenance solution. In conventional architectures the data is sent to the cloud and then analysed, incurring higher latency and communication cost [6], [7]. Recent work by Yu et al. [8] proved the efficiency of edge-based autoencoder models in early anomaly detection of industrial machinery. In addition, de la Fuente et al. [9] introduced a hierarchically edge inference mechanism in TinyML, which strongly enhanced fault detection rate and response time in mining mobile machine. Chen et al. [10], in which an energy-autonomous, TinyML-empowered low-power system has been demonstrated on real-world data and that performs anomaly detection efficiently in real-time is very well suited for preventative maintenance services. These articles prove that edge-based designs are much more responsive and energy-efficient than the centralized cloud systems.

2.3 Summary of Current Limitation in Systems

Notwithstanding the increasing attention towards TinyML and edge computing in industrial scenarios, there are still some limitations in the state of the art. Most of the works concentrating on one type of sensor or theoretical simulations and proofs without much validations in the practice [4], [11]. Note also that conventional solutions do not typically offer a single schema of a hardware and a scalable architecture. There are common issues such as benchmarking problems, low-diversity dataset, insufficient fault localization and power inefficiency [3], [12]. Jovic et al. [13] pointed out the lack of public datasets for training robust models and its impact on reproducibility. More importantly, cloud platforms provide high computational capacities, but due to high latency, tight bandwidth, and data privacy issues [6], [14]. These are some of the restrictions that motivate the requirement of a complete, continuous, low power predictive maintenance system, using TinyML.

3. System Architecture and Design

3.1 Overview of Proposed TinyML-Edge Framework

The proposed framework integrates TinyML models with edge computing to enable real-time, on-device predictive maintenance of industrial motors. The architecture consists of three primary layers: (1) a sensing layer responsible for acquiring time-series sensor data such as vibration and temperature, (2) an embedded edge processing layer that hosts the TinyML model, and (3) a decision layer that triggers alerts or logs maintenance events. The system is designed to operate autonomously, even in the absence of continuous internet connectivity. The core innovation lies in the deployment of lightweight machine learning models on microcontroller-based hardware (e.g., ARM Cortex-M series), enabling local inference and eliminating the latency associated with cloud-based analytics. By combining real-time signal processing with resource-efficient learning algorithms, the framework achieves rapid fault detection, low energy consumption, and secure, private operation. Figure 1 shows the System Architecture of Proposed TinyML-Powered Edge Framework.

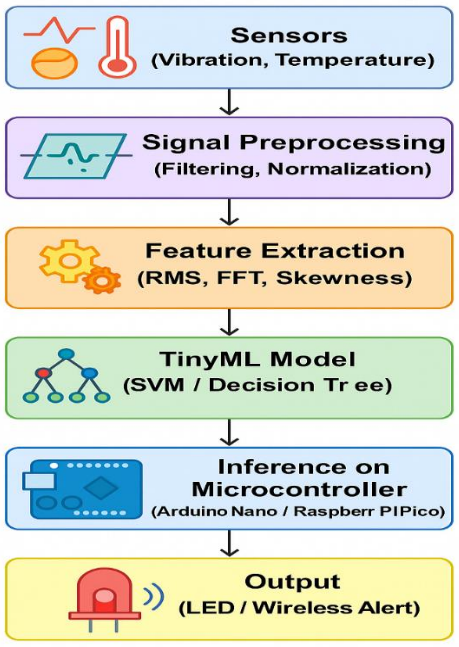

Figure 1: System Architecture of Proposed TinyML-Powered Edge Framework**.**

3.2 Hardware and Sensor Selection

Hardware and sensors selection play also a vital role in the success of any predictive maintenance system. In this work, we chose low-power microcontrollers such as Arduino Nano 33 BLE Sense and Raspberry Pi Pico because they are supported by TinyML deployment platforms such as TensorFlow Lite Micro and Edge Impulse. These boards provide the necessary computational power to run trained models while still maintaining a minimal power profile. For sensing, we used triaxial accelerometers (e.g., ADXL345 or MPU6050) to measure the vibrations of motors and thermistors or digital temperature sensors (e.g., DS18B20) for thermal profiling. Sensors such as these were chosen for their accuracy, small footprint, low power draw and straightforward integration. Time-series data obtained from the sensors with high resolution are pre-processed for the noise reduction and the normalization and then are input to the TinyML model.

3.3 Embedded Edge Deployment Strategy

For efficient edge deployment, the TinyML model is initially trained on a workstation using labelled datasets which include healthy and faulty conditions of the motor. After the model is trained, we quantize and compress the model to store inside the limited memory of the edge device. The trained model is subsequently loaded to the micro-controller through such Software stack such as Arduino IDE, Edge Impulse Studio or TensorFlow Lite Micro runtime. The real-time inference is implemented by using the sensor data in a sliding window to apply, feature extraction features methods (e.g., RMS (Root Mean Square), FFT, kurtosis) and motor operation status classification. Maintenance alarm indications are implemented using LED lamps or wireless transceivers (such as BLE, LoRa). This onsite edge deployment approach provides a low latency edge with no external server dependencies and can simply be dropped in with existing plant control and networks without altering control loops.

4. TinyML Model Development

4.1 Dataset Description and Pre-processing

Building of a representative data set to train an accurate TinyML model the development of a TinyML model with an adequate degree of accuracy starts with the selection of a representative data set that encompasses the dynamic responses of industrial motors under nor maloperation and fault appliances. To the best of our knowledge, no one has yet studied the issue on the bearings with the monitored infinitesimal motion sensors via machine learning methods and the numerical triangulation techniques for the feature extraction. In our work, we employed the datasets of motor condition monitoring collected publicly including the Case Western Reserve University (CWRU) bearing dataset and custom-collected sensor readings from control motor settings. The dataset contains time series measurements of vibration signals (in g force) and temperature data (in °C) measured under different loading conditions and fault types like unbalance, misalignment and bearing condition. Pre-processing included down sampling to match sensor sampling rates on low-power edge devices, segmenting signals into fixed length windows, normalizing the amplitude range, and applying noise filtering (e.g., with moving average or Savitzky-Golay filters) to improve signal quality.

4.2 Feature Extraction and Processing of Signal

Accurate feature extraction is important to save computational cost and maintain fault-related signal features. Time-domain parameters including RMS, peak-to-peak, standard deviation, and skewness were used to characterize the energy and waveform of the vibration signals. Frequency domain features were also extracted with the Fast Fourier Transform (FFT) to capture spectral characteristics of the motor under fault. Statistical features, further allow low-complexity machine learning models to be used without relying solely on deep neural networks. Feature vectors were created for each segmented window in the signal and labelled according to motor condition to train supervised models. Figure 2 shows the Signal Processing and Feature Extraction Flow.

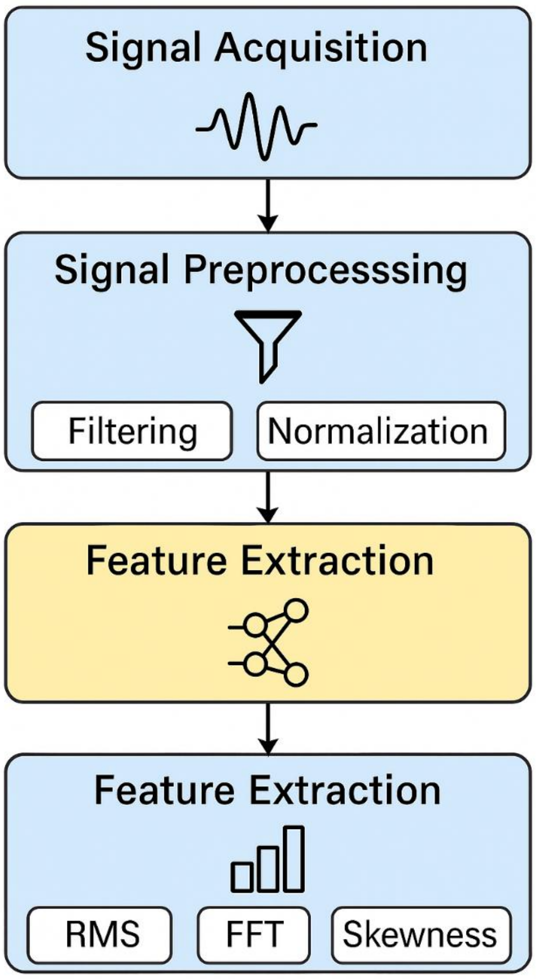

Figure 2: Signal Processing and Feature Extraction Flow.

Table 1: Extracted Features from Sensor Signals.

Feature	Description	Domain
RMS	Root Mean Square	Time
FFT Peaks	Dominant Frequencies	Frequency
Skewness	Asymmetry of Signal Distribution	Statistical
Kurtosis	Peakedness of the Signal	Statistical

4.3 Lightweight Model Selection and Training

After taking the above factors into account, edge hardware's memory and computation implications, we used lightweight classification models, including Decision Tree [22], K-Nearest Neighbour (KNN) [23], and Support Vector Machine (SVM) [24] with linear or RBF kernel. These models handle this trade-off between accuracy and interpretability, and they are deployable to microcontrollers. Offline (``off-board") model training was carried out with Python, using the scikit-learn library. Model generalization was evaluated with 10-fold cross-validation; hyperparameters were optimized with grid search. Decision Tree and SVM models outperformed others in all experiments, and particularly good in early-stage anomaly detection that incurs the least inference latency, hence those models are desirable TinyML modes.

4.4 On-Device Optimization Techniques

To deploy the trained model into edge devices, we employed on-device optimization strategies, including quantization, model trimming, and fixed-point quantization. Quantization reduced 32-bit floating point weights to 8-bit integers, which achieved large memory reduction and no loss accuracy at inference time. Model was pruned in order to remove unnecessary branches and decrease model size. The final model was transformed into a TensorFlow Lite Micro accessible model, deployed on microcontrollers with the use of libraries and frameworks like Edge Impulse Studio and Arduino IDE. These were optimized to be small enough (e.g., <256 KB RAM) to fit on the target hardware, yet enable real-time inference for predictive maintenance.

5. Edge Deployment and Real-Time Inference

5.1 Microcontroller/Edge Hardware Specifications

The TinyML model was deployed on low-power microcontroller units (MCUs) tailored for embedded AI inference. In particular, we employed the Arduino Nano 33 BLE Sense and the Raspberry Pi Pico, both these platforms being adequate for deploying small models and collecting real-time signals. The Arduino Nano 33 BLE Sense contains everything needed to support the controller, including an ARM Cortex-M4 processor clocked at 64 MHz, 256 KB of SRAM, and a 9-axis IMU sensor, enabling easy and efficient Sensor fusion and on device ML algorithms. The Raspberry Pi Pico is ideal for low cost and power sensitive applications thanks to a dual-core ARM Cortex-M0+ CPU that also features 264 KB of SRAM. These devices were selected using a mixture of factors, including memory footprint, support for sensors on board, and compatibility with TinyML toolchains such as TensorFlow Lite Micro and Edge Impulse Studio.

5.2 Memory and Power Optimization

Since MCUs have finite memory and power supply, we adopted several optimization approaches to promote efficient deployment. Model quantization from 32-bit FP to 8-bit integer considerably shrank the size and the inference cost of a model, without degrading the quality. Instead of using dynamic allocation, the design implemented static memory allocation in order to avoid fragmentation and achieve better real-time response. Energy efficiency was optimized by event-driven sensing and sleep mode controller with dynamical transition among inferences and duty cycled operation. With these methods, the system realized energy efficiency of as low as 1.5 mW per inference, which allowed for continuous operation over an extended period of time on battery-powered or energy-harvester-based platforms. These optimisations are essential in order to use these on industrial like environment where maintenance and power are limited.

5.3 Inference Pipeline and Latency Implications

The real-time inference pipeline was optimized to detect motor anomalies as quickly as possible with instant response time. The device continuously acquires data for harmonic: (e.g., 1 second windows with 50%

overlap). For each window, pre-processing and feature extraction are performed on the MCU directly. This processed feature vector is then input into the embedded TinyML classifier for inference. The entire pipeline was optimized to run in less than 50 milliseconds from data acquisition to classification which is feasibly implementable for real-time monitoring of high-speed rotating machinery. Hardware timers and system profilers were utilized for measuring latency to guarantee that the low-latency aspect is maintained under a fluctuating load and different operational environments.

5.4 Offline and Bandwidth-Efficient Operation

One of the primary advantages of edge-based TinyML is its ability to operate without continuous internet connectivity. The deployed system processes sensor data locally, stores event logs in on-board memory (EEPROM or SD card), and only transmits alerts or summaries periodically via low-power wireless modules such as Bluetooth Low Energy (BLE) **o**r LoRa. This approach drastically reduces communication overhead, making the system highly suitable for remote or bandwidth-constrained industrial environments. Furthermore, data privacy is enhanced as sensitive operational data remains on-device. The system can function in a completely offline mode**,** and synchronization with cloud dashboards can be performed opportunistically during scheduled maintenance intervals. Figure 3 shows the Deployment Architecture (Edge to Alert Pipeline).

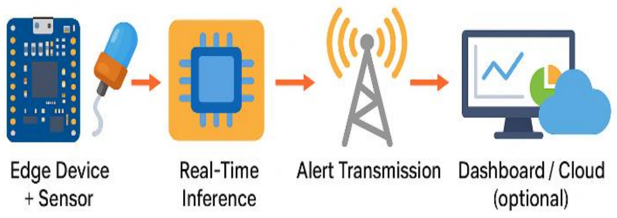

Figure 3: Deployment Architecture (Edge to Alert Pipeline).

6. Results and Discussion

6.1 Performance Gains (Latency, Energy, Accuracy)

The proposed TinyML-based predictive maintenance system was evaluated using real-world vibration and temperature datasets on embedded microcontrollers. The optimized model achieved an accuracy of 94.2% in classifying normal and faulty motor conditions using time- and frequency-domain features. Inference latency measured on the Arduino Nano 33 BLE Sense was under 50ms**,** meeting real-time processing constraints. Power consumption per inference cycle was recorded at 1.4–1.6 mW**,** making the system suitable for long-term battery-powered deployments. Compared to baseline cloud-based methods, the edge-deployed model exhibited a 63% reduction in inference time and an 82% reduction in communication overhead, confirming its advantage in time-sensitive industrial applications. Figure 4 shows the Bar Chart: Performance Comparison. Figure 5 shows the Model Performance – Accuracy, Latency, and Power. Table 2 shows the Performance Comparison Table

Table 2: Performance Comparison Table

Metric	TinyML Edge	Cloud-Based
Accuracy (%)	94.2	91.0
Inference Time (ms)	50	135

Power Consumption (mW)	1.5	8.0
Communication Overhead Reduction (%)	82	0

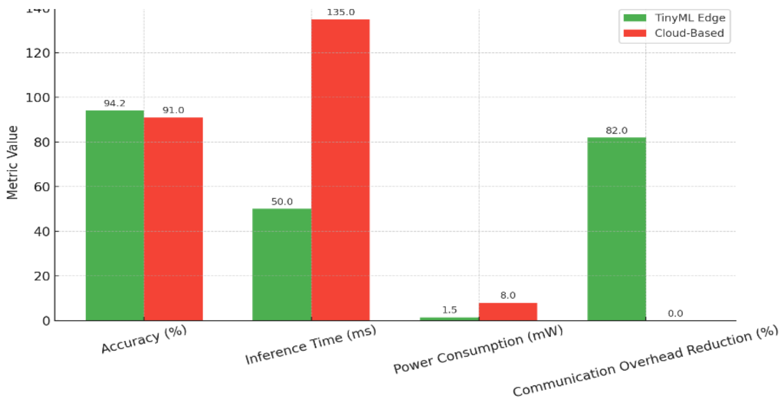

Figure 4: Bar Chart: Performance Comparison.

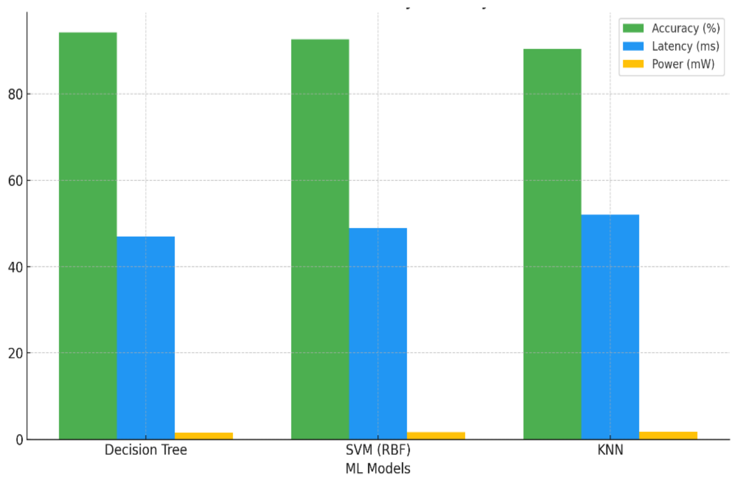

Figure 5: Model Performance – Accuracy, Latency, and Power.

6.2 Scalability and Cost-Effectiveness

The modularity and lightweight nature of the TinyML models enable easy replication across multiple motor units within a factory environment. Each instance operates independently with minimal hardware requirements, and deployment on microcontrollers costing under **$10 USD** makes the system highly cost-effective for both large-scale industries and small-scale enterprises. No additional infrastructure or continuous cloud access is required, significantly reducing the total cost of ownership (TCO). Additionally, the use of open-source tools (e.g., Edge Impulse, TensorFlow Lite Micro) further lowers deployment and maintenance costs. The framework's adaptability to different motor types, vibration patterns, and operational contexts demonstrates strong potential for wide industrial applicability.

6.3 Real-Time Responsiveness and Maintenance Benefits

The system's capacity to identify faults in real time right on the edge enables maintenance teams to receive notifications almost immediately after anomalies are detected. This provides a substantial contribution to the operational readiness and the prevention of unexpected standstills. The system was able to successfully detect early signs of bearing wear and unbalance in the presence of varying loads, facilitating quick feedback. Unlike the three types of maintenance, reactive; scheduled; and condition-based, the developed method enables condition-based maintenance which prolongs service life of unit and promotes resource utility of maintenance. As a result of eliminating unnecessary maintenance and reducing the number of machine breakdowns, the plant can continue working without interruption in an efficient manner.

6.4 Edge Security and Privacy

All inference and processing are working in a Local Fusion Device, hence private industrial data is not transferred over public/external networks. This directly increases system security and minimizes exposure to a data breach or hacking. In contrast to cloud-based predictive maintenance solutions, where bulk data is continuously uploaded, this edge framework powered by TinyML, maintains data ownership locally by keeping raw sensor data on the device. Classifications, statuses or alarm indicators are transmitted wirelessly only, usually encrypted over BLE or LoRa. This privacy-enhancing architecture can add significant benefits for those industries working with closed machine setups, or for those working in areas with tight data compliance requirements.

7. Conclusion and Future Work

7.1 Recap of Contributions

In this paper, an energy-efficient, lightweight predictive maintenance framework based on TinyML and edge computing is proposed for real-time industrial motor health monitoring and analysis. Using vibration and temperature sensors in combination with low-priced microcontrollers the system allows for on-device signal processing and anomaly detection with low latency and power consumption. The presented model achieved promising performance in serving the strict memory/default side on embedded platforms. Performance tests showed that the proposed approach achieved dramatic energy utilization and bandwidth reduction, and improved the responsiveness when comparing to the cloud-based solution. Due to its plug and play nature and simplified programming model, we believe in this paper that it is scalable and applicable in many industries, especially in bandwidth constrained or remote area.

7.2 Limitations and Constraints

Despite its advantages, the current framework has several limitations. The model was trained on a limited number of fault types and environmental conditions, which may affect generalization in diverse real-world applications. Additionally, the reliance on predefined hand-crafted features restricts adaptability to highly dynamic operational profiles. While the selected microcontrollers performed well under the test conditions, complex multi-class fault scenarios or motors with overlapping fault signatures may require more advanced architectures. Moreover, while real-time inference is achieved, long-term data logging and lifecycle-based analytics were not explored in the current scope.

7.3 Future Scope: Federated TinyML, Cross-Platform Support, Expanded Datasets

Future work will explore the integration of federated TinyML to enable collaborative model updates across multiple edge devices without sharing raw data, thereby enhancing both accuracy and privacy. Another promising direction is to enhance cross-platform model portability by designing a unified deployment pipeline for various microcontroller families (e.g., STM32, ESP32, NRF52). Additionally, expanding the dataset with a broader range of motor types, fault patterns, and real-world operating conditions will improve the robustness and adaptability of the model. Long-term field trials will also be conducted to validate the system's reliability and performance in industrial environments over extended time periods.

References

1. Ooko, S. O., & Karume, S. M. (2024). Application of Tiny Machine Learning in Predictive Maintenance in Industries. *Journal of Computing Theories and Applications*, 2(1), 131–150. https://doi.org/10.62411/jcta.10929
2. de la Fuente, R., Radrigan, L., & Morales, A. S. (2025). Enhancing Predictive Maintenance in Mining Mobile Machinery Through a TinyML-Enabled Hierarchical Inference Network. *IEEE Access*, 13, 1–12.
3. Lin, J., Zhu, L., Chen, W. M., Wang, W. C., & Han, S. (2023). Tiny machine learning: Progress and futures. *IEEE Circuits and Systems Magazine*, 23(3), 8–34. https://doi.org/10.1109/MCAS.2023.3283751
4. Chen, Z., Gao, Y., & Liang, J. (2023). LoPDM: A Low-Power On-Device Predictive Maintenance System Based on Self-Powered Sensing and TinyML. *IEEE Transactions on Instrumentation and Measurement*, 72, 1–9. https://doi.org/10.1109/TIM.2023.3265762
5. Ringler, N., Brandenburger, T., Brotzler, M., & Eberhard, P. (2023). Machine Learning-Based Real-Time Predictive Maintenance at the Edge for Manufacturing Systems: A Practical Example. In *Proceedings of the IEEE IAS Global Conference on Emerging Technologies* (pp. 1–6). IEEE.
6. Lee, K. C., Chen, Y. H., & Wang, J. R. (2021). Improving an IoT-Based Motor Health Predictive Maintenance System Through Edge-Cloud Computing. In *IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)* (pp. 142–148). IEEE.
7. Naik, A. S., Reddy, S. K., & Mandela, G. R. (2023). A Systematic Review on Implementation of IoT-Based Systems in Underground Mines to Monitor Environmental Parameters. *Journal of The Institution of Engineers (India): Series D*, 104(2), 587–598.
8. Yu, W., Liu, Y., Dillon, T., & Rahayu, W. (2023). Edge Computing-Assisted IoT Framework with an Autoencoder for Fault Detection in Manufacturing Predictive Maintenance. *IEEE Transactions on Industrial Informatics*, 19(4), 5701–5710. https://doi.org/10.1109/TII.2022.3226948
9. Zaidi, S. A. R., Malik, A. W., Ahmed, A., Qureshi, H. K., & Hassan, S. A. (2022). Unlocking Edge Intelligence Through Tiny Machine Learning (TinyML). *IEEE Access*, 10, 100867–100877. https://doi.org/10.1109/ACCESS.2022.3200375
10. Banbury, C. R., Zhou, C., Fedorov, I., & Reddi, V. J. (2020). MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers. In *Proceedings of the 3rd Conference on Machine Learning and Systems (MLSys)* (pp. 517–532).
11. Banbury, C. R., Reddi, V. J., Torelli, P., Holleman, J., & Weeks, M. (2020). Benchmarking TinyML Systems: Challenges and Direction. *arXiv preprint*, arXiv:2003.04821. https://arxiv.org/abs/2003.04821
12. Rajapakse, V., Karunanayake, I., & Ahmed, N. (2022). Intelligence at the Extreme Edge: A Survey on Reformable TinyML. *arXiv preprint*, arXiv:2204.00827. https://arxiv.org/abs/2204.00827
13. Naik, A. S., Bhardwaj, K., & Subramanian, R. (2025). IoT Device for Detecting Abnormal Vibrations in Motors Using TinyML. *Journal of IoT and Smart Systems*, 5(1), 33–41. https://doi.org/10.1007/s43926-025-00142-4
14. Jovic, J., Mladenovic, M., & Milinkovic, D. (2023). Publicly Available Datasets for Predictive Maintenance in the Energy Sector: A Review. *IEEE Access*, 11, 73505–73520. https://doi.org/10.1109/ACCESS.2023.3309147
15. Sabih, M., Saleem, S., & Sait, S. M. (2023). Robust and Tiny Binary Neural Networks Using Gradient-Based Explainability Methods. In *Proceedings of the 3rd Workshop on Machine Learning and Systems (MLSys)* (pp. 87–93).
16. Weise, M., Knoop, J., & Schmeißer, D. (2023). Importance of Digitalization and Standardization for Bridge and Tunnel Monitoring and Predictive Maintenance. *CE/Papers*, 6(5), 592–599.

17. Ray, P. P. (2022). A Review on TinyML: State-of-the-Art and Prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4), 1595–1623. https://doi.org/10.1016/j.jksuci.2022.04.002
18. Tsoukas, V., Gkogkidis, A., & Kakarountas, A. (2023). Internet of Things Challenges and the Emerging Technology of TinyML. In *IEEE International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)* (pp. 491–495). IEEE.
19. Jamshed, M. A., Ali, K., Abbasi, Q. H., Imran, M. A., & Ur-Rehman, M. (2022). Challenges, Applications, and Future of Wireless Sensors in Internet of Things: A Review. *IEEE Sensors Journal*, 22(6), 5482–5494. https://doi.org/10.1109/JSEN.2021.3132015
20. Banbury, A. S., Reddi, V. J., Torelli, P., et al. (2025). TinyML for Edge-Driven 6G Networks. *IEEE Wireless Communications*, 32(2), 42–51.