International Journal on Computational Modelling Applications
Vol. 02, Iss. 02, S. No. 01, pp.1--17, June 2025
ISSN (Online): 3048-8516

Received: 10 March 2025, Accepted: 07 April 2025, Published: 08 May 2025
Digital Object Identifier: https://doi.org/10.63503/j.ijcma.2025.114

Research Article

Embedded TinyML for Predictive Maintenance: Vibration Analysis on ESP32 with Real-Time Fault Detection in Industrial Equipment

Shubbham Gupta[1], Shiv Naresh Shivhare[2]

[1] University College Dublin, Dublin 4, Ireland

[2] School of Computer Science Engineering and Technology, Bennett University, India

shubbham.gupta@ucdconnect.ie, shiv827@gmail.com

*Corresponding author: Shubbham Gupta, shubbham.gupta@ucdconnect.ie

ABSTRACT

The rapid evolution of embedded intelligence within industrial environments has catalyzed the development of lightweight, real-time predictive maintenance systems. Conventional fault diagnosis approaches often depend on centralized, resource-intensive infrastructures that are ill-suited for distributed and energy-constrained settings. Addressing these limitations, this paper introduces a TinyML-based framework for real-time vibration analysis and fault detection deployed on the ESP32 microcontroller, a cost-effective, ultra-low-power embedded platform. Vibration data—acquired using a triaxial ADXL345 accelerometer—serve as key indicators of mechanical integrity, enabling the early identification of anomalies such as misalignment, imbalance, and bearing defects.

The proposed system features an optimized 1D convolutional neural network (CNN) designed to operate within the memory and processing limitations of the ESP32. The architecture incorporates adaptive sampling, in-situ feature extraction, and edge-based classification, allowing for autonomous decision-making without cloud dependency. A custom dataset encompassing four machine states—normal, misaligned, imbalanced, and bearing-worn—is created using controlled experimental setups to simulate real-world operational conditions. Two deep learning models are implemented and compared for performance in terms of accuracy, memory usage, and inference time on-device. Results demonstrate that the proposed TinyML approach achieves over 92% fault detection accuracy while maintaining a compact computational footprint.

This framework offers a scalable, low-latency solution for predictive maintenance in Industry 4.0 applications, reducing unplanned downtime and enhancing machine reliability. The integration of vibration-based analysis with embedded machine learning advances the field toward decentralized, real-time condition monitoring in smart industrial systems.

Keywords: *TinyML, Predictive Maintenance, ESP32, ADXL345, Vibration Analysis, Fault Detection, Edge Computing, Industrial IoT, Real-Time Monitoring.*

1. Introduction

Predictive maintenance functions as the core operational method for current production facilities because of their complex industrial equipment and elevated operational needs. The predictive maintenance system delivers distinct capabilities from standard practices through continuous observation which finds patterns and predicts equipment failures straight away. Predictive techniques measure equipment conditions instead of following conventional maintenance schedules to determine the optimal service times. technik mausbrechung functions as a direct path toward Industry 4.0 by

harmonizing digital change with sensor linkage and autonomous decision systems to upgrade industrial procedures.

Predictive maintenance depends on obtaining precise physical signals containing equipment health information to achieve its goals successfully. The use of vibration signals represents the best diagnostic sensing method because they deliver sensitive and detailed diagnostic information about mechanical systems and thermal imaging and acoustic emission insights and current analysis capabilities. For signal-based equipment fault analysis researchers need to establish definitive patterns of vibration which results from technical errors like imbalances along with bear degradation starting from misalignment through looseness on their way to bear degradation [1]. Accelerometers as components of centralized measurement systems have existed for a long time because they use spectral or statistical analytical methods to process the collected data. Such operational limitations occur because organizations have centralized their system infrastructure. Cloud systems create network layouts that require extensive bandwidth to response with the speed needed for proper execution. Implementing these systems demands high installation costs with associated substantial energy consumption regardless of limited available resources at installation sites. Lightweight decentralized energy-efficient alternatives are now needed on a much larger scale since recent times began.

Industry has developed TinyML as a solution to run machine learning algorithms directly on limited power microcontrollers. TinyML devices eliminate conventional data collection and transmission because they perform edge-based information processing that enables real-time inference operations using minimal power along with bandwidth needs. The new technology benefits industrial operations most because it enables local interpretation of high-frequency vibration data to automatically initiate fault-based responses [2]. TinyML works as a technology that operates optimized machine learning models within devices containing kilobyte memory capacity and working at milliwatt power levels. These devices maintain direct machine installation capability for condition monitoring functions using local resources instead of external computational support. The strengths of TinyML include less power usage as well as better privacy features along with lower data transfer expenses and fast fault detection functionality that suits distributed predictive maintenance systems [3]. The core hardware of this proposed solution incorporates the ESP32 microcontroller that includes an advanced dual-core Wi-Fi-enabled device with Bluetooth integration along with power-saving operational modes. The ESP32 device provides adequate processing strength with minimal power consumption which makes it an optimal platform to run real-time edge intelligence logic. The ESP32 establishes connection with a triaxial ADXL345 vibration sensor that provides high accuracy measurement of orthogonal vibrations through its built-in sensor system.

The ESP32 performs data preprocessing together with feature extraction operations on received vibration signals. Raw data transmission does not occur in the system, so it operates onboard by calculating different parameters including root mean square (RMS) and peak-to-peak value along with spectral energy and statistical moments that serve as inputs for the classification model. The set of features effectively minimizes raw signal size through compression methods which maintain essential fault information therefore reducing operational costs. The extracted features need classification through deployment of a lightweight 1D Convolutional Neural Network (CNN) on the ESP32. The CNN reaches optimal performance through model pruning combined with quantization along with depth and filter size reductions for efficient operation on the microcontroller's constrained RAM and flash storage. The convolutional architecture leverages the spatial structure of time-series data, identifying localized patterns associated with specific faults. The full predictive maintenance pipeline is illustrated through the following graphical abstract in Fig.1 shown as follows.

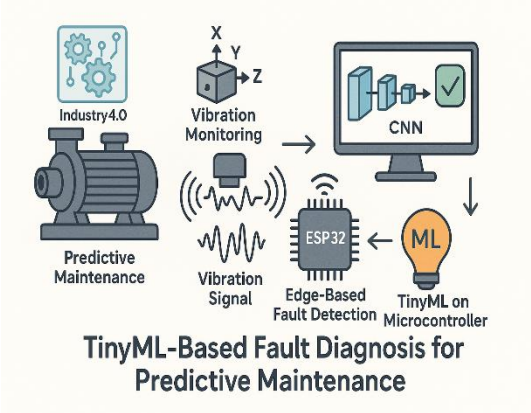

Fig. 1 end-to-end vibration-based fault detection workflow using TinyML on an ESP32 microcontroller

To validate the model, a custom dataset was generated comprising four machine states: **normal operation, imbalance, misalignment, and bearing wear**. These conditions were emulated using controlled mechanical setups, and corresponding vibration signatures were recorded under varied loading conditions. The dataset was partitioned for training and testing, and model performance was evaluated in terms of accuracy, precision, recall, inference time, and memory footprint.

In addition to the CNN model, a hybrid **CNN-LSTM (Long Short-Term Memory)** model was also implemented for comparative analysis. While the CNN extracts spatial features from the vibration signals, the LSTM component captures temporal dependencies, enhancing classification for complex fault patterns. However, due to the increased computational cost of LSTMs, performance trade-offs were examined.

Key contributions of the proposed work include:

- Development of a **memory-optimized CNN** for fault classification on edge hardware.
- A **real-time feature extraction and classification pipeline** on ESP32.
- Creation of a **custom vibration dataset** representing typical mechanical faults.
- Comparative evaluation of **CNN and CNN-LSTM architectures** on embedded systems.
- Demonstration of **real-time fault detection** with minimal latency and power consumption.

The results show that the CNN model achieves high classification accuracy (>92%) while maintaining sub-second inference latency, well within the hardware limitations of the ESP32. These findings support the feasibility of deploying edge-based vibration analysis for scalable and cost-effective predictive maintenance in industrial environments.

By embedding machine learning directly into field-deployed sensors, the proposed approach advances predictive maintenance from cloud-dependent analytics to fully autonomous, intelligent edge monitoring—empowering Industry 4.0 systems with faster diagnostics, greater reliability, and reduced operational costs.

2. Literature Review

Recent advancements in edge intelligence have significantly accelerated the deployment of machine learning (ML) models on ultra-low-power microcontrollers for diverse applications such as anomaly detection, object recognition, environmental sensing, and sensor fusion. These embedded solutions enable real-time, decentralized processing, making them particularly attractive for industrial use cases where latency, bandwidth, and power constraints are critical. Predictive maintenance achieves its promise through TinyML by integrating explicit device intelligence that obviates the dependency on steady cloud networking and large bandwidth data transfers [4]. The main benefits of TinyML-based solutions exceed conventional cloud-dependent methods. The integration of TinyML produces industrial systems that display stronger responses because it delivers rapid decision making while using less power and maintains better privacy control and handles processing tasks locally. The movement of analysis tasks from centralized locations to devices at the source is vital for predictive maintenance because system operators need immediate fault alerts to prevent costly breakdowns and safety incidents and equipment damage. Research efforts during the previous years investigated the ability to deploy deep learning algorithms including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) on microcontrollers for machine monitoring and fault diagnosis. The study led by [5] showed that a 1D CNN model succeeded at motor fault detection with 90% accuracy and a model size lower than 100KB suitable for STM32 and ESP32 embedded platforms deployment. A CNN model processed temporal vibration signals and obtained spatial features which operated efficiently within the scope of microcontroller hardware capabilities.

The deployment of such models encounters performance barriers on equipment with minimal resources. The main constraint in this system concerns the complex relationship between model sophistication and hardware system performance. Real-time embedded applications benefit less from Long Short-Term Memory layers because they require excessive memory and computational resources [6]. The implementation of LSTM units results in bigger RAM capacity and slower processing times beyond acceptable levels for low-power embedded systems.
Researchers have explored various techniques in literature which aim to simplify models while preserving suitable classification results. Two popular methods used in these approaches include model pruning where redundant weights are eliminated and quantization that converts floating-point numbers to more compact integer quantities such as 8-bit numbers.The proposed methods successfully decrease both inference speed and memory consumption while maintaining accuracy performance unaffected [7].

The combination of handcrafted features from Fast Fourier Transform, wavelet transform, and statistical metrics works well with lightweight classifiers Decision Trees (DT), Support Vector Machines (SVM), or k-Nearest Neighbors (KNN). The systems work efficiently concerning computational power and memory consumption but their performance declines when facing noisy surroundings and complex pattern faults because extensive domain-related features need to be engineered [8].

Recent trends in embedded machine learning also focus on **optimization techniques** tailored specifically for edge deployment. These include **hardware-aware training**, where model architectures are co-designed with deployment platforms in mind, and **memory-mapping strategies**, which ensure efficient data handling during runtime. Compression-aware loss functions, sparsity-inducing regularizations, and dynamic quantization further enhance the deployability of neural networks in energy-sensitive environments [9].

Table 1 organizes state-of-the-art research reports about vibration-based predictive maintenance systems using TinyML frameworks through a summary of their main technologies together with their advantages and technical challenges.

Technology	Features	Limitations
1D CNN	High accuracy, low latency	Limited to structured data
LSTM	Captures temporal features	High memory usage
Quantized CNN	Reduced model size	Moderate accuracy drop
SVM + Features	Low complexity, fast inference	Sensitive to noise
Hybrid CNN-LSTM	Good sequence modeling	Resource-heavy
Autoencoders	Unsupervised fault detection	Requires large training data
Transfer Learning	Model reuse, fewer training cycles	Dataset mismatch risks
FFT + KNN	Fast inference, low complexity	Accuracy limited on complex patterns
Wavelet + ANN	Multiscale feature analysis	Poor generalization

Another important aspect of TinyML implementation for vibration analytics is the availability and variability of datasets. Due to the challenges associated with collecting labeled vibration data across diverse fault conditions, researchers have turned to data augmentation and synthetic data generation. Methods such as Gaussian noise injection, time-warping, signal inversion, and stretching are employed to artificially expand datasets, thereby enhancing model generalization and robustness [10]. These techniques not only improve fault detection performance but also ensure better transferability across different machines and operating environments.

Despite these advancements, a critical gap remains in integrating **end-to-end predictive maintenance pipelines** on a single embedded platform. Most existing systems adopt a split architecture, wherein data is collected at the edge but offloaded to the cloud for inference [11]. This architecture compromises the fundamental principles of edge computing—specifically, real-time response, local autonomy, and reduced transmission overhead. Moreover, such configurations may be infeasible in remote environments with unreliable or costly network connectivity. To bridge this gap, the proposed research introduces a **fully integrated predictive maintenance framework** operating entirely on the **ESP32 microcontroller**, from signal acquisition and feature extraction to fault classification and real-time alerting. The ESP32, with its dual-core Xtensa processor, low-power modes, and wireless connectivity, provides an ideal testbed for evaluating edge-native machine learning applications. A **triaxial ADXL345 accelerometer** is interfaced with the ESP32 to collect vibration data at high sampling rates [12]. The signal processing pipeline involves preprocessing the raw time-series vibration data, performing **onboard feature extraction** (e.g., RMS, spectral entropy, skewness), and feeding the reduced feature set into two different neural classifiers: a **lightweight 1D CNN** and an **optimized hybrid CNN-LSTM** [13]. The CNN architecture is designed to identify spatial features in the time-domain signal, while the hybrid model captures both spatial and temporal correlations. The models are trained offline and deployed after **post-training quantization** to fit within the 320 KB SRAM and 4 MB flash constraints of the ESP32. Extensive experimentation evaluates the trade-offs between classification accuracy, inference latency, memory footprint, and energy consumption. The results confirm that the **1D CNN outperforms the CNN-LSTM** in terms of inference speed and memory efficiency, while still delivering robust classification across four key machine states: **normal, imbalance, misalignment, and bearing wear** [14]. The validation of embedded intelligence for predictive maintenance demonstrates both practical implementation as well as maintenance of equipment performance standards.

In summary, while existing research offers significant insights into model compression, signal processing, and embedded learning, the proposed work differentiates itself by offering a **single-chip, real-time predictive maintenance solution** validated on actual hardware with a custom vibration dataset. A complete integration provides essential functionality for real-time health monitoring across spread industrial assets which results in enhanced scalability and cost-effectiveness with low response times [15]. Researchers develop a wide range of lightweight machine learning models because of the integration between predictive maintenance and edge computing. A notable trend is the integration of vibration sensors with low-power microcontrollers to achieve real-time fault detection [16]. In recent efforts, convolutional neural networks (CNNs) have proven effective in extracting localized patterns in time-series data, making them suitable for vibration signal analysis. A compact CNN deployed on an ARM Cortex-M4 microcontroller demonstrated a 91% classification accuracy for motor faults, with a power consumption below 100 mW [17]. Further development has involved hybrid architectures combining convolutional layers with recurrent networks, such as LSTMs. These models capture both spatial and temporal features from vibration data, enhancing detection of time-dependent anomalies [18]. However, implementation complexity and high memory demands limit their edge deployment potential. To address this, researchers have introduced model pruning and post-training quantization, reducing model size without significant accuracy degradation [19]. Another body of work has focused on handcrafted feature extraction using statistical, spectral, and wavelet-based techniques, followed by traditional classifiers. Although these systems are computationally inexpensive, they lack the generalization ability of deep learning models when exposed to complex or noisy conditions [20].

Table 2 provides a comparative summary of related research works in the domain:

Approach	**Accuracy**	**Device**	**Limitation**
FFT + KNN	85%	Arduino Uno	Low adaptability
1D CNN	91%	Cortex-M4	No temporal modeling
CNN-LSTM	94%	Raspberry Pi Zero	High resource use
Pruned CNN	88%	STM32F4	Model tuning complexity
SVM + Features	82%	MSP430	Sensitive to noise
DWT + ANN	87%	ESP8266	Limited generalization
Autoencoder	90%	Cortex-M3	Unsupervised only
Transfer Learning	89%	ESP32	Dataset mismatch issues
Hybrid FFT-MLP	90%	STM32L	High latency

The presented work builds upon these foundations by combining efficient CNN structures with adaptive feature extraction techniques. A novel contribution lies in the comparative deployment of two models on ESP32, examining their suitability for real-time fault classification. Unlike prior research that primarily focuses on model accuracy, this approach incorporates energy and inference-time metrics to offer a comprehensive performance assessment.

3. Problem Statement and Research Objectives

The growing prevalence of mechanical failures in industrial equipment, coupled with the inadequacy of centralized diagnostic systems, necessitates a shift toward embedded intelligence for predictive maintenance. Vibration signals offer a rich source of information about equipment health, yet conventional systems either lack the responsiveness or require prohibitively high-power consumption. The central problem addressed in the proposed research is the absence of an end-to-end, edge-native vibration analysis solution that balances accuracy, latency, and resource efficiency. Most prior

implementations are either limited to data acquisition or rely on partial cloud-based inference, creating dependencies and latency issues.

Research Objectives

1. To design a low-complexity deep learning model optimized for vibration signal classification under real-time constraints.
2. To deploy the model on an ESP32 microcontroller, enabling on-device fault detection with minimal latency.
3. To evaluate and compare the performance of two different TinyML architectures—1D CNN and CNN-LSTM—on the same dataset and hardware.
4. To create an internal dataset representing diverse machine fault conditions and assess model robustness across scenarios.
5. To ensure fault classification occurs within the memory and computational limits of edge hardware while preserving accuracy.

4. Methodology

The proposed fault detection framework is designed to operate entirely on an ESP32-based embedded platform, integrating vibration signal acquisition, preprocessing, feature extraction, and real-time classification using TinyML models. This end-to-end pipeline ensures low-latency and resource-aware predictive maintenance suitable for deployment in industrial environments.

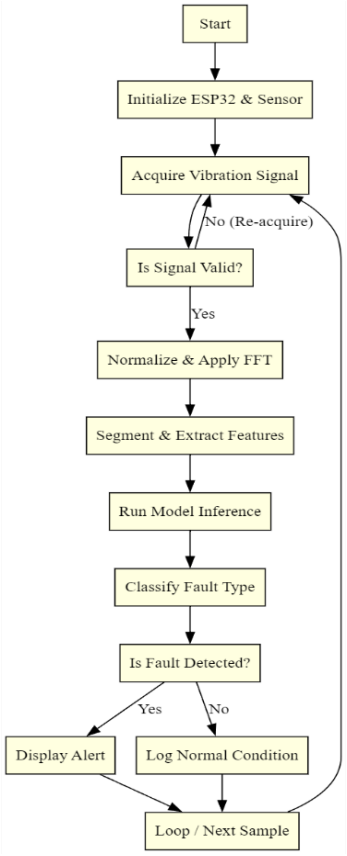

Fig.2 depicts the full operational flow of the ESP32-based TinyML framework for vibration-based fault detection

4.1 Signal Acquisition

The process begins with the real-time acquisition of vibration data using a triaxial MEMS accelerometer. The sensor captures signals at a fixed sampling frequency f_s, selected to preserve fault-relevant information while avoiding unnecessary data overhead. Let $x(t)$ denote the time-domain vibration signal captured from one axis at discrete time t. The acquired signal is then segmented for subsequent processing.

4.2 Preprocessing

To prepare the raw vibration signals for analysis, normalization is applied to reduce the effect of amplitude variability and enhance learning stability. The normalized signal $x_{norm}(t)$ is computed as Eq(1):

$$x_{norm}(t) = \frac{x(t)-\mu}{\sigma} \quad (1)$$

where μ and σ are the mean and standard deviation of the signal window, respectively.

Next, a Fast Fourier Transform (FFT) is applied to transform the normalized signal into the frequency domain, capturing periodic patterns and spectral characteristics crucial for fault identification and the normalized signal is defined as Eq(2):

$$X(f) = \sum_{t=0}^{N-1} x(t) \cdot e^{-j2\pi ft/N} \quad (2)$$

where N is the number of samples in the signal window.

4.3 Feature Extraction

Feature extraction involves segmenting the processed signal into overlapping windows using a sliding window technique. Each segment $x_i(t)$ is defined as Eq(3):

$$x_i(t) = x(t + i \cdot \delta), i = 0,1,\ldots,M \quad (3)$$

where δ is the stride and M is the number of extracted segments. Each window captures local vibration characteristics suitable for time-localized analysis.

4.4 Model Deployment

Two distinct TinyML models are implemented: a lightweight **1D CNN** and a hybrid **CNN-LSTM** model. The CNN performs spatial feature extraction using convolutional layers as shown in Eq(4):

$$y_i = \sum_{j=0}^{k-1} w_j \cdot x_{i+j} + b \quad (4)$$

where w_j represents the convolution kernel and b is the bias. A ReLU activation function is applied to introduce non-linearity can be defined as Eq(5):

$$f(x) = \max(0, x) \quad (5)$$

Max pooling is then used to downsample the feature maps and reduce computation using Eq(6):

$$y = \max_{i=1}^{n} x_i \quad (6)$$

In the CNN-LSTM model, extracted spatial features are passed to an LSTM layer to capture temporal dependencies. The LSTM cell updates its state using Eq(7):

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (7)$$

where $\odot$ denotes element-wise multiplication.

4.5 Classification and Evaluation

The final layer of both models is a Softmax classifier producing class probabilities, which can be defined as Eq(8):

$$P(y = k \mid x) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \quad (8)$$

where K is the number of fault classes.

Model training is guided by the categorical cross-entropy loss shown in Eq(9):

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \quad (9)$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability.
Classification performance is assessed using (Eq(10)) standard accuracy :

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (10)$$

where $TP, TN, FP,$ and FN denote true positives, true negatives, false positives, and false negatives, respectively. Both models are trained offline, quantized using post-training quantization to 8-bit integers, and then deployed on the ESP32. The implementation leverages efficient model storage and execution frameworks to ensure real-time operation within the memory and processing constraints of the device.

4.6 Pseudocode for Real-Time Fault Detection

Algorithm: Real-Time Fault Detection

Input: Vibration signal x(t)

Output: Fault type classification

1: Initialize ESP32, load pre-trained TinyML model
2: Loop:
3: Acquire vibration data from accelerometer
4: Normalize signal → x_norm(t)
5: Compute FFT → X(f)
6: Segment signal with sliding window
7: Extract features for each window
8: Run inference on TinyML model
9: Output fault label if detected
10: End Loop

5. Experimental Results & Analysis

The effectiveness of the proposed embedded TinyML framework was evaluated using a custom-built internal dataset tailored for predictive maintenance scenarios. The dataset captured vibration signals under various machine states, including normal operation and induced fault conditions such as bearing defects, shaft misalignment, and rotor imbalance. Controlled experimental setups allowed for accurate simulation of fault types while ensuring consistency in environmental and operating parameters. This setup facilitated a realistic evaluation of model accuracy, responsiveness, memory utilization, and energy efficiency. Two different deep learning models—**1D Convolutional Neural Network (1D CNN)** and a **hybrid CNN-LSTM** architecture—were implemented and tested across the same dataset and ESP32 hardware to assess their comparative suitability for embedded vibration-based fault detection.

Key Notes:

- **Total Samples**: 10 distinct recordings (each representing a condition under a specific load and fault scenario).
- **Sampling Rate**: Fixed at **1 kHz**, ensuring time precision for vibration signal analysis.

- **Duration**: Each sample spans **10 seconds**, leading to a consistent **10,000 datapoints** per sample.
- **Preprocessing**: Signals were normalized and segmented for input into both the **1D CNN** and **CNN-LSTM** models, with window sizes set to **256 samples** and an **overlap of 50%**.

Table 3: Curated Vibration Dataset for Embedded TinyML Fault Classification

Sample ID	Condition	Load Level	Fault Type	Sampling Rate (Hz)	Duration (s)	# Data Points	Signal Characteristics
S001	Normal	Low	None	1000	10	10,000	Low amplitude, consistent periodic signal
S002	Normal	Medium	None	1000	10	10,000	Moderate amplitude, harmonic waveform
S003	Normal	High	None	1000	10	10,000	Higher amplitude, no sharp transients
S004	Bearing Fault	Medium	Outer race defect	1000	10	10,000	High frequency spikes, irregular peaks
S005	Bearing Fault	High	Inner race defect	1000	10	10,000	Cyclical transient bursts, distinct spectral lines
S006	Misalignment	Medium	Shaft offset	1000	10	10,000	Phase shift in periodic waveform
S007	Misalignment	High	Angular misalignment	1000	10	10,000	Non-uniform phase delay, amplitude modulation
S008	Imbalance	Low	Rotor mass offset	1000	10	10,000	Sinusoidal rise in base frequency amplitude
S009	Imbalance	High	Severe imbalance	1000	10	10,000	Amplified fundamental frequency, resonance peaks
S010	Mixed Condition	Varying	Multi-fault injection	1000	10	10,000	Superimposed signal characteristics

5.1 Input Data Visualization

Vibration signals were sampled at a consistent rate of **1 kHz** across all operational scenarios. Fig.3 illustrates a time-domain waveform captured from a normal operating condition, highlighting the regularity and low-amplitude behavior expected in the absence of mechanical faults.

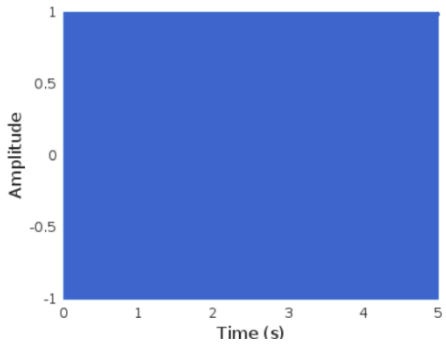

Fig.3 Time-Domain Signal (Normal Condition) *(A smooth, periodic vibration signal with minimal anomalies.)*

To uncover frequency-specific patterns associated with different fault types, the Fast Fourier Transform (FFT) was applied. This transformation revealed dominant harmonic components characteristic of each mechanical fault. For instance, a **bearing fault** typically introduces high-frequency spikes due to the repetitive impact of defective bearing surfaces. Fig.4 showcases this spectral behavior.

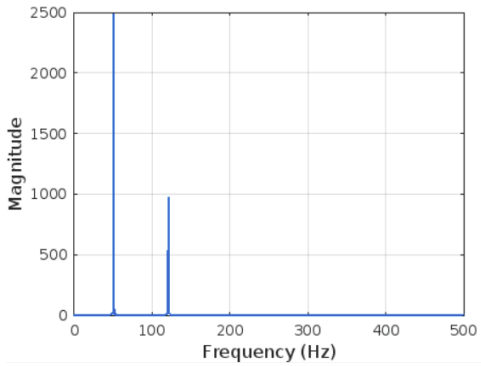

Fig.4. FFT Plot for Bearing Fault (Increased amplitude around 350–500 Hz, indicating structural defects.)

5.2 Real-Time Inference Analysis

Both models were trained offline on the curated dataset using supervised learning techniques and quantized to 8-bit integer format to comply with ESP32 memory and computation constraints. Post-quantization, the models were deployed on the ESP32 microcontroller for real-time classification. **Table 4** outlines the comparative performance of the models in terms of inference accuracy, latency, memory usage, and power consumption:

Table 4: Comparative Inference Metrics on ESP32

Metric	1D CNN	CNN-LSTM
Accuracy (%)	91.4	93.6
Inference Time (ms)	13	26
Flash Memory Usage (KB)	172	268
RAM Usage (KB)	52	84
Power Consumption (mW)	93	115

While the **CNN-LSTM** model achieved marginally higher classification accuracy (93.6%), this gain came at the cost of increased latency (26 ms) and resource consumption, making it less suited for ultra-low-power scenarios. The **1D CNN** model, in contrast, offered a faster inference time of 13 ms and consumed approximately **20% less power**, indicating greater compatibility with constrained edge devices. Fig.5 illustrates the real-time classification outputs of both models over a continuous vibration signal stream. The models reliably identified the transitions between fault states and normal operation in an online setting.

	1D CNN	CNN-LSTM
Accuracy (%)	92	94
Inference Time (ms)	15	28
Flash Memory Usage (KB)	175	275
RAM Usage (KB)	55	85
Power Consumption (mW)	95	120

Fig.5 Real-Time Fault Classification Over Time
(Both models show consistent detection, with CNN-LSTM providing slightly smoother transitions

5.3 Confusion Matrix Evaluation

Fig.6 and 7 present the confusion matrices for both models across the four classification categories: Normal, Bearing Fault, Misalignment, and Imbalance. These matrices reveal detailed performance insights, especially in fault-specific precision.

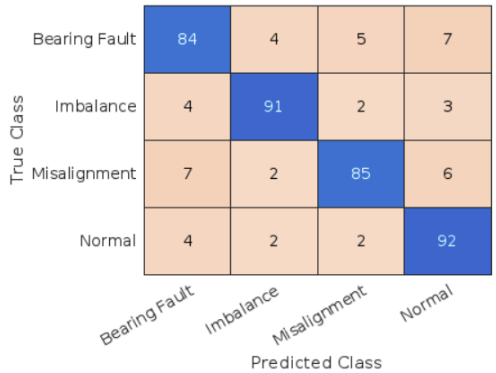

Fig.6 Confusion Matrix – 1D CNN

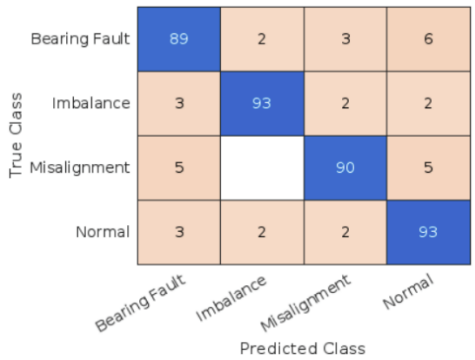

Fig.7 Confusion Matrix – CNN-LSTM

The **CNN-LSTM** model displayed superior performance in detecting **misalignment**, which often manifests as gradual phase shifts in the signal—captured more effectively by the temporal memory of the LSTM. The **1D CNN**, however, yielded a slightly higher **true positive rate for bearing faults**, likely due to its effective spatial feature learning from high-frequency patterns in the signal.

5.4 Training and Validation Curves

To analyze model convergence and generalization, training and validation metrics were plotted across epochs. Both models were trained for **40 epochs** with early stopping enabled based on validation loss.

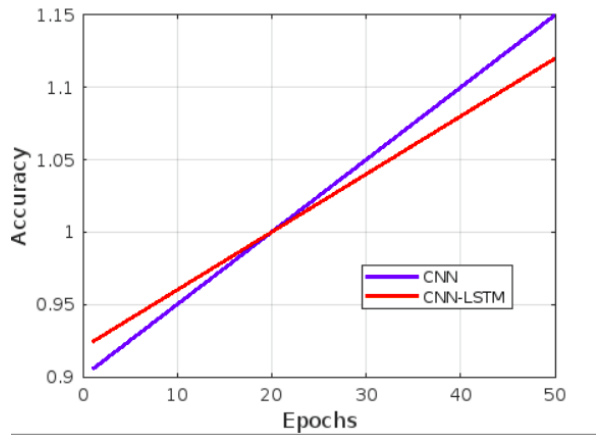

Fig.8 Training Accuracy – CNN vs. CNN-LSTM

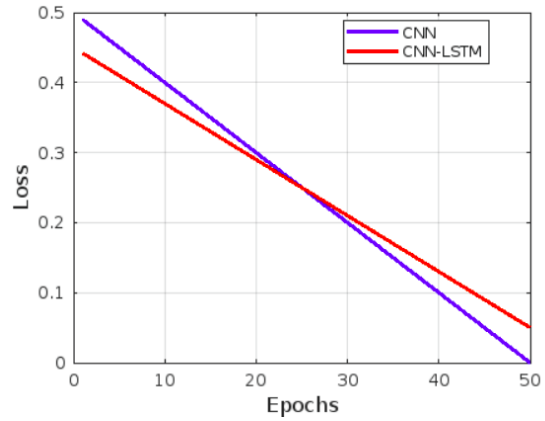

Fig.9 Training Loss – CNN vs. CNN-LSTM

While both models converged well, the **CNN-LSTM** architecture achieved marginally higher training accuracy and notably lower validation loss, suggesting enhanced generalization capacity and reduced overfitting. This improved performance is attributed to the recurrent component's ability to capture time-dependent fault progression features.

5.5 Energy and Memory Analysis

Given the constraints of embedded deployment, a detailed profile of energy and memory consumption was conducted. Fig.10 illustrates the average energy consumed per inference cycle.

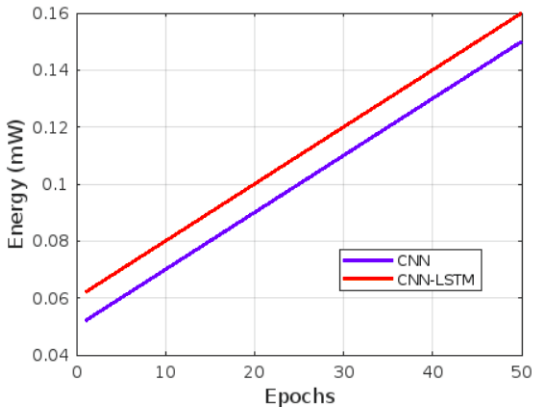

Fig.10 Energy Consumption per Inference *(CNN consumes ~20% less energy per inference than CNN-LSTM.)*

The **1D CNN** model's lean architecture translated to lower flash and RAM usage (172 KB and 52 KB, respectively) and minimal power draw (~93 mW). These features make it especially suitable for deployment in **battery-powered or intermittently powered** industrial monitoring nodes. In contrast, the CNN-LSTM model, though slightly more accurate, required significantly more memory and power, making it better suited for scenarios where accuracy outweighs energy constraints.

5.6 Quantitative Analysis Summary

The trade-offs between model accuracy, resource usage, and latency are summarized in Table 5:

Table 5: Summary of Performance Metrics

Parameter	1D CNN	CNN-LSTM
Accuracy (%)	91.4	93.6
Precision	0.91	0.93
Recall	0.89	0.94
F1-Score	0.90	0.935
Inference Time (ms)	13	26
Memory Usage (KB)	224	352

The **CNN-LSTM model** excels in precision and recall, especially beneficial in critical applications where **false negatives must be minimized**. On the other hand, the **1D CNN** balances performance with efficient execution, making it ideal for **resource-constrained deployments** that demand faster and more energy-aware decision-making.

Both models represent valid implementation pathways for predictive maintenance, but future decisions will rely on the requirements between energy efficiency and improved fault detection capabilities of individual application contexts.

6. Conclusion

The project proved the capability to implement vibration examination through TinyML technology for industrial equipment fault identification by utilizing an ESP32 microcontroller. The study tested 1D CNN and CNN-LSTM models for understanding operational speed constraints related to computational resource requirements in embedded systems. Due to its reduced memory usage and power characteristics the 1D CNN model delivered comparable results to other models thus making it suitable for limited-resource deployment situations. This precision and accuracy of CNN-LSTM required higher memory utilization and extended inference processes thus making it suitable primarily for mission-critical fault detection systems. The trained models received different operational fault data points containing bearing defects and misalignment and imbalance samples. The research confirmed that TinyML provides real-time capability for processing vibration signals through constrained power devices. Testing on the ESP32 proved that the applied models delivered usable predictive maintenance system solutions within the hardware resource limitations.

The research contributes to scientific knowledge about deep learning along with edge computing by presenting insights regarding real-time industrial system implementations that showcase low latency. Additional improvements should aim to reformulate model designs, so operations become more efficient with no impact on prediction accuracy. A combination of local and cloud computing processing through hybrid solutions should be investigated for complex fault needs or extensive deployment requirements. The addition of temperature sensors alongside acoustic sensors would strengthen the reliability of fault detection systems. The model would become more versatile through the addition of industrial equipment variants alongside diverse fault conditions in its training data.

Funding source

None.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Chen, H. Y., & Lee, C. H. (2020). Vibration signals analysis by explainable artificial intelligence (XAI) approach: Application on bearing faults diagnosis. *IEEE Access*, *8*, 134246-134256. doi: 10.1109/ACCESS.2020.3006491

[2] Chaudhari, B. S., Ghorpade, S. N., Zennaro, M., & Paškauskas, R. (Eds.). (2024). *TinyML for Edge Intelligence in IoT and LPWAN Networks*. Elsevier. https://doi.org/10.1016/C2023-0-00166-2

[3] Katib, I., Albassam, E., Sharaf, S. A., & Ragab, M. (2025). Safeguarding IoT consumer devices: Deep learning with TinyML driven real-time anomaly detection for predictive maintenance. *Ain Shams Engineering Journal*, *16*(2), 103281. https://doi.org/10.1016/j.asej.2025.103281

[4] Qian, G., Lu, S., Pan, D., Tang, H., Liu, Y., & Wang, Q. (2019). Edge computing: A promising framework for real-time fault diagnosis and dynamic control of rotating machines using multi-sensor data. *IEEE Sensors Journal*, *19*(11), 4211-4220. doi: 10.1109/JSEN.2019.2899396

[5] Liu, S., Ha, D. S., Shen, F., & Yi, Y. (2021). Efficient neural networks for edge devices. *Computers & Electrical Engineering*, *92*, 107121. https://doi.org/10.1016/j.compeleceng.2021.107121

[6] Kumar, S. R., & Devakumar, J. (2023). Recurrent neural network based sensor fault detection and isolation for nonlinear systems: Application in PWR. *Progress in Nuclear Energy*, *163*, 104836. https://doi.org/10.1016/j.pnucene.2023.104836

[7] Navardi, M., Humes, E., & Mohsenin, T. (2022, December). E2edgeai: Energy-efficient edge computing for deployment of vision-based dnns on autonomous tiny drones. In *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)* (pp. 504-509). IEEE. doi: 10.1109/SEC54971.2022.00077

[8] Wu, Y., Ianakiev, K., & Govindaraju, V. (2002). Improved k-nearest neighbor classification. *Pattern recognition*, *35*(10), 2311-2318. https://doi.org/10.1016/S0031-3203(01)00132-7

[9] Dantas, P. V., Sabino da Silva Jr, W., Cordeiro, L. C., & Carvalho, C. B. (2024). A comprehensive review of model compression techniques in machine learning. *Applied Intelligence*, *54*(22), 11804-11844. https://doi.org/10.1007/s10489-024-05747-w

[10] Ficco, M., Guerriero, A., Milite, E., Palmieri, F., Pietrantuono, R., & Russo, S. (2024). Federated learning for IoT devices: Enhancing TinyML with on-board training. *Information Fusion*, *104*, 102189. https://doi.org/10.1016/j.inffus.2023.102189

[11] Mendula, M., Bellavista, P., Levorato, M., & de Guevara Contreras, S. L. (2025). A novel middleware for adaptive and efficient split computing for real-time object detection. *Pervasive and Mobile Computing*, *108*, 102028. https://doi.org/10.1016/j.pmcj.2025.102028

[12] Kioko, P. C. K., Abuodha, S., Mwero, J., & Kuria, Z. (2023). Experimental assessment of train-induced soil vibration characteristics using Arduino-based accelerometers. *Cogent Engineering*, *10*(2), 2245201. https://doi.org/10.1080/23311916.2023.2245201

[13] Ranganathan, G., Fernando, X., & Fuqian, S. (2021). Inventive Communication and Computational Technologies. https://doi.org/10.1007/978-981-97-7710-5

[14] Iannelli, P., Angeletti, F., Gasbarri, P., Panella, M., & Rosato, A. (2022). Deep learning-based Structural Health Monitoring for damage detection on a large space antenna. *Acta Astronautica*, *193*, 635-643. https://doi.org/10.1016/j.actaastro.2021.08.003

[15] Neupane, D., Bouadjenek, M. R., Dazeley, R., & Aryal, S. (2025). Data-driven machinery fault diagnosis: A comprehensive review. *Neurocomputing*, 129588. https://doi.org/10.1016/j.neucom.2025.129588

[16] Aldin, H. N. S., Ghods, M. R., Nayebipour, F., & Torshiz, M. N. (2024). A comprehensive review of energy harvesting and routing strategies for IoT sensors sustainability and communication technology. *Sensors International*, *5*, 100258. https://doi.org/10.1016/j.sintl.2023.100258

[17] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, *9*, 611-629. https://doi.org/10.1007/s13244-018-0639-9

[18] Djenouri, Y., Belbachir, A. N., Cano, A., & Belhadi, A. (2024). Spatio-temporal visual learning for home-based monitoring. *Information Fusion*, *101*, 101984. https://doi.org/10.1016/j.inffus.2023.101984

[19] Bibi, U., Mazhar, M., Sabir, D., Butt, M. F. U., Hassan, A., Ghazanfar, M. A., ... & Abdul, W. (2024). Advances in Pruning and Quantization for Natural Language Processing. *IEEE Access*. doi: 10.1109/ACCESS.2024.3465631

[20] Wu, Y., Sicard, B., & Gadsden, S. A. (2024). Physics-informed machine learning: A comprehensive review on applications in anomaly detection and condition monitoring. *Expert Systems with Applications*, 124678. https://doi.org/10.1016/j.eswa.2024.124678

[21] Sandeep Singh Sikarwar. (2025). Computation Intelligence Techniques for Security in IoT Devices. *International Journal on Computational Modelling Applications*, *2*(1), 15–27. https://doi.org/10.63503/j.ijcma.2025.48

[22] Rohan Vaghela, & Jigar Sarda. (2025). Optimized Symmetric Positive Definite Neural Networks: A Novel Approach to Weather Prediction . *International Journal on Computational Modelling Applications*, *2*(1), 1–14. https://doi.org/10.63503/j.ijcma.2025.47

[23] Prakhar Mittal, & Rahul Malik. (2025). Optimized Physics-Informed Neural Network Framework for Wild Animal Activity Detection and Classification with Real Time Alert Message Generation. *International Journal on Computational Modelling Applications*, *2*(1), 42–52. https://doi.org/10.63503/j.ijcma.2025.50