TinyML for Fault Detection in Rotating Machinery: A Post-Training Quantization Assessment on the MAFAULDA Dataset

Cândido Alfredo Carvalho de Lucena Filho[1], **Felipe Valencia de Almeida**[1]

[1] Polytechnique School of the Universidade de São Paulo – São Paulo, SP – Brazil

{candidoalfredo,fvalencia}@usp.br

Abstract. *Unplanned downtime in rotating machinery is a major cost driver in industrial operations, and early fault detection through vibration analysis can prevent it. Running these classifiers directly on microcontrollers (TinyML) reduces inference latency and enables faster on-device response, but it is still unclear how different model architectures and quantization schemes behave under strict memory and flash constraints. This paper evaluates the feasibility of embedding rotating-machinery fault machine learning classifiers on low-cost microcontrollers. We compare MLP, 1D-CNN, LSTM and Random Forest on the MAFAULDA dataset, using 132 features per window and stratified group k-fold cross-validation to prevent leakage across windows from the same file. Three post-training quantization schemes (FP32, FP16, INT8) are evaluated against the requirements of the ESP32-S3, STM32F4, and ESP32-WROOM. The INT8 MLP was the most efficient model conforming to all three platforms (37 KB, F1=0.98), reducing size by 67% over its FP32 counterpart with negligible accuracy loss. The 1D-CNN lost accuracy under INT8 (F1 from 0.94 to 0.75), and the unrolled LSTM grew 3× rather than shrinking, results that highlight the need to jointly evaluate the model and quantization, not in isolation.*

1. Introduction

Rotating machinery such as pumps, compressors, and electric motors represents a large portion of industrial equipment in operation. Monitoring these complex systems through vibration analysis represents an effective way to predict faults before they lower efficiency due to unplanned downtime [Randall and Antoni 2011, Lei et al. 2020]. More recently, data-driven classification trained on vibration signals has demonstrated strong performance on distinguishing among multiple fault types [Souza et al. 2021].

Traditionally, these classifiers run on servers or cloud-based platforms, introducing latency and reliance on network connectivity. TinyML addresses this limitation by enabling ML models to run on microcontrollers with a few kilobytes of RAM, allowing inference to occur at the sensor level [Warden and Situnayake 2019, David et al. 2021]. However, microcontrollers operate under hard constraints on flash (1–4 MB) and SRAM (128–520 KB), which means that a workstation model may not fit the target hardware.

Post-training quantization (PTQ) reduces the size of the model by converting weights and activations from 32-bit floats to 16-bit floats or 8-bit integers, often achieving a compression factor of 2–4× with little accuracy loss [Jacob et al. 2018,

Gholami et al. 2021]. However, in practice, the balance between compression and performance depends on the architecture and the calibration dataset, meaning that not all models will benefit equally.

Although TinyML-based fault detection has received increased attention over the last few years [Martinez-Rau et al. 2024, Thota et al. 2025], most studies focus on a single model and hardware. Few of them compare how different architectures behave in PTQ against real microcontroller constraints. In addition, issues related to data leakage that originate from overlapping windows in cross-validation are often overlooked [Kapoor and Narayanan 2023].

This paper addresses these limitations by training four classification approaches: Multi-Layer Perceptron (MLP), 1D Convolutional Neural Network (1D-CNN), Long Short-Term Memory (LSTM), and Random Forest (RF), using the MAFAULDA rotating-machine fault dataset [Marins et al. 2018]. We evaluated three PTQ schemes (FP32, FP16, INT8) and assessed the resulting models against the specifications of three commercial microcontrollers. To avoid data leakage, we employed cross-validation using stratified group k-fold splitting to ensure that windows from the same recording never appear in both training and test sets.

The contributions of this paper are: (i) a comparative benchmark of four architectures on MAFAULDA with group-aware cross-validation to avoid data leakage; (ii) an evaluation of post-training quantization strategies covering FP32, FP16, and INT8 with stratified calibration; and (iii) a deploy conformity analysis relating each quantized model to three target platforms (ESP32-S3, STM32F4, ESP32-WROOM), identifying the MLP as the only architecture viable across all three, with INT8 yielding the smallest footprint.

2. Related Work

Machine learning for fault detection. Lei et al. [Lei et al. 2020] review the evolution from traditional feature-engineering approaches to end-to-end deep learning methods in rotating-machinery fault diagnostics. Earlier techniques relied on extracting statistical and spectral features and feed them to classifiers such as Random Forests [Breiman 2001]. More recent studies covering 1D-CNNs applied to raw vibration signals have reported accuracies above 99%. However, these results may be inflated due to cross-validation setups that do not prevent data leakage between overlapping windows [Kapoor and Narayanan 2023].

The MAFAULDA dataset. Marins et al. [Marins et al. 2018] introduced the Machinery Fault Database (MAFAULDA), collected using a SpectraQuest Machinery Fault Simulator at Universidade Federal do Rio de Janeiro. The dataset contains measurement from 8 channels at 50 kHz, covering 6 operational conditions with different severity levels. Khan et al. [Khan et al. 2021] combined synthetic data augmentation and CNNs with MAFAULDA, reporting classification accuracies above 99%, but with standard k-fold that does not prevent same-file windows from appearing in both training and testing partitions.

TinyML and quantization. TensorFlow Lite Micro [David et al. 2021] enables the deployment of Keras models on microcontrollers through a flat-buffer format and lightweight interpreter. Jacob et al. (2018) demonstrated that INT8 quantization can reduce model size by approximately 4× while maintaining under 1% accuracy drop on

image benchmarks, though Gholami et al. (2021) noted that the impact of quantization depends on activation distributions and architecture. In TinyML fault detection, Thota et al. (2025) deployed a 1D-CNN for bearing classification reporting 99% accuracy with 29 KB RAM, while Martinez-Rau et al. (2024) proposed a microcontroller-based anomaly detection approach.

None of these previous works compares systematically multiple architectures under multiple quantization schemes against multiple platform constraints in a leakage-free evaluation setting. This study aims to address this gap.

3. Methodology

3.1. Dataset and Feature Extraction

We used the MAFAULDA dataset [Marins et al. 2018], which contains 1951 CSV files distributed across 6 fault categories. Each file has 8 channels sampled at 50 kHz, with approximately 250,000 samples per channel. In this study, only the six accelerometer channels (three orthogonal axes, axial, radial, and tangential, per bearing) are considered, while the microphone and tachometer signals are excluded from the analysis. Each recording is segmented into windows of 2048 samples with 50% overlap, generating 97,550 windows in total. For each window, we extracted 22 statistical and spectral features per accelerometer channel, resulting in a 132-dimensional feature vector.

Table 1 summarizes the class distribution. The dataset is imbalanced, with overhang bearing failures accounting for approximately 26% of all windows, while normal operation accounts for only 2.5%.

Table 1. MAFAULDA class distribution after windowing.

Class	Windows	%
Normal	2,450	2.5
Horizontal misal.	10,000	10.3
Vertical misal.	10,150	10.4
Imbalance	24,350	25.0
Underhang bearing	24,950	25.6
Overhang bearing	25,650	26.3
Total	97,550	100.0

3.2. Model Architectures

We evaluated four classifiers representing different machine learning paradigms:

MLP. A fully-connected network with two hidden layers of 128 and 64 units (ReLU), batch normalization, dropout (0.3), and a softmax output layer with 6 units. Input: the 132-dimensional feature vector. Total parameters: 28,326.

1D-CNN. Two convolutional blocks (32 and 64 filters of size 3, ReLU, batch normalization, max-pooling) followed by global average pooling, a 64-unit dense layer, and softmax. Input: the feature vector reshaped to $(132, 1)$. Total parameters: 14,214.

LSTM. A single LSTM layer with 16 units (`unroll=True` for TFLite compatibility), batch normalization, dropout (0.3), a 32-unit dense layer, and softmax. Input: $(132, 1)$. Total parameters: 1,958.

Note that both the 1D-CNN and LSTM receive the 132-dimensional *engineered feature vector*, not the raw temporal signal; consequently, neither architecture can fully exploit its inductive bias (spatial locality for convolutions, sequential dependencies for recurrence).

Random Forest. An ensemble of 200 trees with no maximum depth, using Gini impurity and the square root of the number of features at each split [Breiman 2001]. Included as a non-neural baseline for reference; its large serialized size makes it impractical for microcontroller deployment.

All neural network models were trained using the Adam optimizer together with sparse categorical cross-entropy loss. Early stopping with a patience value of five epochs was employed based on validation loss to reduce overfitting and prevent unnecessary training iterations.

3.3. Cross-Validation Strategy

A common challenge in windowed-based vibration analysis is data leakage: when windows from the same recording file appear in both training and test folds, the model can exploit the overlap between adjacent windows rather than learning generalizable patterns. This approach can artificially inflate accuracy by 5–15 percentage points [Kapoor and Narayanan 2023].

To avoid this problem, we used *Stratified Group K-Fold* cross-validation with $k = 5$, where the group variable is the source file. This guaranteed that all windows from a given file are assigned to the same fold. We verified this with an assertion: if the intersection of training-set file IDs and test-set file IDs is non-empty, execution halts. In addition, class stratification preserved representative label distributions across all folds.

3.4. Post-Training Quantization

After completing cross-validation, the model corresponding to the best-performing fold for each neural-network architecture is exported and converted to TensorFlow Lite using three different post-training quantization (PTQ) configurations. We selected the best fold because TFLite conversion requires a single concrete model; the low baseline variance (Table 3) suggests consistent quantization behavior across folds.

FP32 (baseline). The model was converted with no optimization. Weights and activations remained in 32-bit floating point.

FP16 (float16). Weights were stored in 16-bit floating point. The converter flag `OPTIMIZE_FOR_SIZE` is enabled with `supported_types = [tf.float16]`. At inference level, weights are upcast to FP32 on the fly.

INT8 (full integer). Weights and activations were quantized to 8-bit integers. This required a *representative dataset* for calibration. We construct this dataset by sampling 1,000 examples using stratified sampling (equal count per class) to avoid biasing the quantization range toward the majority class.

The Random Forest model was serialized with joblib and assessed without quantization, since TFLite does not support tree ensembles.

3.5. Platform Conformity Assessment

We defined conformity as the simultaneous satisfaction of three conditions on a target platform: (i) the model fits in flash, (ii) the inference arena fits in SRAM, and (iii) the F1-score is at least 0.90. We adopted F1-macro rather than accuracy because the dataset is imbalanced (Table 1): a model that ignores minority classes can still achieve high accuracy, whereas F1-macro penalizes poor recall on any class equally. The 0.90 threshold reflects a practical deployment requirement: below this level, more than 10% of fault events are missed or incorrectly classified, which is unacceptable for condition-monitoring systems where undetected faults can lead to equipment damage. Table 2 lists the target platforms and their constraints.

Table 2. Target microcontroller platforms.

Platform	SRAM (KB)	Flash (MB)
ESP32-S3	520 (320 usable)	4
STM32F4	128	1
ESP32-WROOM	520 (280 usable)	4

For TFLite models, the arena size (peak memory during inference) is evaluated by the interpreter. For Random Forest model, both flash and RAM requirements equal the serialized model size, since the entire tree structure reside in memory at inference time.

4. Results and Discussion

4.1. Baseline Classification

Table 3 presents the 5-fold cross-validation results using Stratified Group K-Fold. The Random Forest achieved the highest F1-macro (0.994), followed closely by the MLP (0.981). The 1D-CNN reaches 0.856, while the LSTM trails at 0.560, consistent with the fact that neither architecture can exploit its inductive bias on engineered feature vectors (Section 3.2). These results are lower than the $>$99% accuracies reported in prior MAFAULDA studies [Khan et al. 2021, Marins et al. 2018]; this gap is explained by our use of group-aware cross-validation, which prevents the model from exploiting overlap between adjacent windows of the same file.

Table 3. Baseline 5-fold Stratified Group K-Fold results.

Model	Params	Acc.	F1-macro	Precision	Recall
MLP	28,326	$0.981_{\pm 0.010}$	$0.981_{\pm 0.011}$	$0.982_{\pm 0.008}$	$0.982_{\pm 0.013}$
1D-CNN	14,214	$0.897_{\pm 0.047}$	$0.856_{\pm 0.073}$	$0.876_{\pm 0.073}$	$0.854_{\pm 0.068}$
LSTM	1,958	$0.701_{\pm 0.088}$	$0.560_{\pm 0.112}$	$0.614_{\pm 0.097}$	$0.570_{\pm 0.100}$
RF	—	$0.997_{\pm 0.002}$	$0.994_{\pm 0.006}$	$0.997_{\pm 0.002}$	$0.991_{\pm 0.009}$

It is worth noting that the RF achieved 1.000 in all folds for F1-macro before we switched from standard Stratified K-Fold to Stratified Group K-Fold. The drop to 0.994 confirms that the earlier result was inflated by data leakage across overlapping windows.

4.2. Quantization Effects

Table 4 summarizes the impact of PTQ on each model, the key observations are as follows.

Table 4. Post-training quantization results (test set).

Model	Quant.	F1	Size (KB)	Arena (KB)	Red. (%)
MLP	FP32	0.985	111.0	109.1	—
	FP16	0.985	58.8	162.9	47.0
	INT8	0.984	36.7	27.9	66.9
1D-CNN	FP32	0.945	65.0	278.9	—
	FP16	0.945	40.0	306.3	38.4
	INT8	0.749	29.4	70.3	54.7
LSTM	FP32	0.576	375.5	248.5	—
	FP16	0.576	373.1	252.2	0.6
	INT8	0.515	1134.7	62.3	−202.2
RF	Joblib	0.999	23,918	23,918	—

MLP: robust to quantization. F1 drops from 0.985 (FP32) to 0.984 (INT8), while model size shrinks from 111 KB to 37 KB (−67%) and the arena from 109 KB to 28 KB. Fully-connected layers have smooth activation distributions that are well-obtained by uniform INT8 quantization [Gholami et al. 2021].

1D-CNN: relevant INT8 degradation. F1 is preserved under FP16 (0.945) but drops to 0.749 under INT8. Convolutional activations tend to have long-tailed distributions that lose information when clipped to 8-bit ranges. Without stratified calibration, F1 dropped to 0.487, confirming that the calibration dataset composition is critical for INT8.

LSTM: quantization increases the model size. The LSTM under INT8 grew from 375 KB to 1,135 KB, a 3× increase. The `unroll=True` flag expands the LSTM into 132 static copies of the recurrence block; under INT8, each copy obtains its own quantization parameters and `QUANTIZE`/`DEQUANTIZE` operators, whose overhead exceeds the savings from reduced precision. FP16 also barely reduces size (373 vs. 375 KB) because the converter shares weight tensors across the unrolled blocks.

Random Forest: accurate but not embeddable. The RF achieves F1 = 0.999 but requires 23.9 MB, roughly 650× the MLP INT8. Tree ensembles cannot be quantized with TFLite, making deployment on any of the three platforms infeasible.

4.3. Platform Conformity

Figure 1 and Table 5 show which model–quantization pairs conform to each target platform. A model is marked as conformable (✓) only if it fits simultaneously in flash and SRAM, and achieves F1 $\geq$ 0.90.

The MLP was the only architecture that conformed to all three platforms at any quantization level. Among its variants, INT8 offers the smallest footprint (37 KB model, 28 KB arena), leaving space for firmware, drivers, and communication stacks. The MLP FP16 fails on STM32F4 because the FP16 weights are upcast to FP32 at inference, pushing the arena to 163 KB, above the 128 KB SRAM limit.

The 1D-CNN FP32 fits in the ESP32-S3 and ESP32-WROOM, but its INT8 variant is excluded by the F1 $<$ 0.90 threshold despite fitting in memory. No LSTM or RF configuration meets all three requirements on any platform.

Table 5. Platform conformity assessment. ✓ = viable (flash OK, RAM OK, F1 ≥ 0.90).

Model	Quant.	ESP32-S3	STM32F4	ESP32-WROOM
MLP	FP32	✓	✓	✓
	FP16	✓	—	✓
	INT8	✓	✓	✓
1D-CNN	FP32	✓	—	✓
	FP16	✓	—	—
	INT8	—	—	—
LSTM	all	—	—	—
RF	Joblib	—	—	—

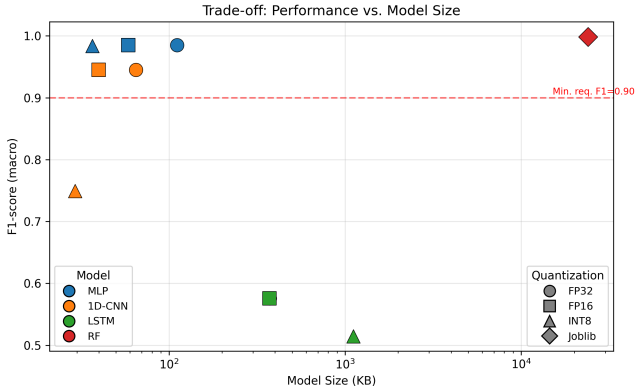

Figure 1. F1-score versus model size for all model–quantization pairs. The dashed line marks F1 = 0.90; top-left models are ideal for deployment.

4.4. Discussion

Three lessons stand out. First, *quantization is not universally beneficial*: its impact depends on architecture-specific factors that cannot be predicted from parameter count alone. Second, *data leakage inflates results*: our RF dropped from F1 = 1.000 to 0.994 after switching to group-aware splitting. Third, *stratified calibration matters*: the 1D-CNN's F1 jumped from 0.487 to 0.749 with class-balanced calibration.

5. Conclusion

We compared four classifiers (MLP, 1D-CNN, LSTM, RF) for rotating-machinery fault detection under three PTQ schemes and three microcontroller platforms. Using MAFAULDA with leakage-free cross-validation, the MLP was the only architecture conforming to all targets, and its INT8 variant provides the most efficient deployment option (37 KB, F1 = 0.98). The study showed that INT8 quantization can degrade or increase model size depending on architecture, and that stratified calibration is essential for imbalanced fault datasets.

Future work will focus on deploying the INT8 MLP on real ESP32-S3 hardware to measure inference latency, energy consumption, and accuracy with live sensor data. This on-device validation is essential to confirm that quantized accuracy holds under real-world noise and to establish a practical TinyML pipeline for industrial predictive mainte-

nance. We also plan to explore platform-specific optimizations and lightweight classical alternatives as deployment baselines.

The source code and experimental *notebooks* for full reproducibility are available at: `https://github.com/candidoalfilho/MAFAULDA_TINY_ML_ANALYSIS`.

References

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Wang, T., Warden, P., and Rhodes, R. (2021). TensorFlow Lite Micro: Embedded machine learning for TinyML systems. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 3, pages 800–811.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. (2021). A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713.

Kapoor, S. and Narayanan, A. (2023). Leakage and the reproducibility crisis in machine-learning-based science. *Patterns*, 4(9):100804.

Khan, A., Hwang, H., and Kim, H. S. (2021). Synthetic data augmentation and deep learning for the fault diagnosis of rotating machines. *Mathematics*, 9(18):2336.

Lei, Y., Yang, B., Jiang, X., Jia, F., Li, N., and Nandi, A. K. (2020). Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mechanical Systems and Signal Processing*, 138:106587.

Marins, M. A., Ribeiro, F. M. L., Netto, S. L., and da Silva, E. A. B. (2018). Improved similarity-based modeling for the classification of rotating-machine failures. *Journal of the Franklin Institute*, 355(4):1913–1930.

Martinez-Rau, L. S., Zhang, Y., Oelmann, B., and Bader, S. (2024). TinyML anomaly detection for industrial machines with periodic duty cycles. In *Proceedings of the IEEE Sensors Applications Symposium (SAS)*.

Randall, R. B. and Antoni, J. (2011). Rolling element bearing diagnostics—a tutorial. *Mechanical Systems and Signal Processing*, 25(2):485–520.

Souza, R. M., Nascimento, E. G. S., Miranda, U. A., Silva, W. J. D., and Lepikson, H. A. (2021). Deep learning for diagnosis and classification of faults in industrial rotating machinery. *Computers & Industrial Engineering*, 153:107060.

Thota, Y. R., Afshar, M., Boden, S., Dunlap, B., Akin, B., and Nikoubin, T. (2025). TinyML enabled real-time bearing fault classification in motors using vibration signals. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI)*.

Warden, P. and Situnayake, D. (2019). *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media.