Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

TinyOL-ARLDA On-Device Learning Auto Regularized Linear Discriminant Analysis for Sound-Based Predictive Maintenance

Abdelwahed ELMOUTAOUKKIL

elmoutaoukkilabd@gmail.com

University of Hassan II

Mohammed HAMLICH

University of Hassan II

Research Article

Keywords: Tiny Machine Learning (TinyML), Online Learning (OL); Linear Discriminant Analysis (LDA), Predictive Maintenance (PdM), Acoustic Fault Detection, Edge AI, Incremental Learning, Industrial IoT (IIoT), Sound-Based Condition Monitoring, Embedded Systems

Posted Date: June 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9670538/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

TinyOL-ARLDA On-Device Learning Auto Regularized Linear Discriminant Analysis for Sound-Based Predictive Maintenance

Abdelwahed ELMOUTAOUKKIL [1] (corresponding author), Mohammed HAMLICH [2]
[1, 2] University of Hassan II, Ensam, CCPS Laboratory, Casablanca, Morocco
Email: [1] elmoutaoukkilabd@gmail.com, Email:[2] moha.hamlich@gmail.com,

Abstract **Industrial systems increasingly demand real-time fault detection without reliance on cloud infrastructure. This paper introduces TinyOL-ARLDA, an incremental on-device learning framework based on Regularized Linear Discriminant Analysis (LDA) for sound driven predictive maintenance. Unlike traditional offline models, TinyOL-ARLDA performs online adaptation of class means, class priors, and covariance estimates directly on microcontrollers, enabling continuous learning from streaming acoustic data. The proposed method extracts low dimensional spectral and temporal features including RMS energy, zero-crossing rate, and higher-order statistical descriptors from machine sound signals and updates the LDA parameters through exponential moving averages. This mechanism enables dynamic adaptation to evolving machine conditions without relying on cloud resources. Experimental evaluations conducted on real industrial sound datasets (MIMII) demonstrate that TinyOL-ARLDA achieves over 94% classification accuracy while maintaining a memory footprint under 20 KB. Furthermore, a hardware implementation on an ESP32 confirms its feasibility for edge deployed acoustic fault detection. These results highlight the potential of TinyOL-ARLDA as a lightweight, interpretable, and energy efficient solution for real-time predictive maintenance in industrial and IoT environments.**

Keywords Tiny Machine Learning (TinyML), Online Learning (OL); Linear Discriminant Analysis (LDA), Predictive Maintenance (PdM), Acoustic Fault Detection, Edge AI, Incremental Learning, Industrial IoT (IIoT), Sound-Based Condition Monitoring, Embedded Systems.

I. INTRODUCTION

Predictive maintenance (PdM) has become a cornerstone of Industry 4.0, aiming to anticipate equipment failures through continuous condition monitoring in order to reduce downtime, maintenance costs, and operational risks. By leveraging data-driven techniques, PdM enables early fault detection and improves system reliability in industrial environments [1]. Recent surveys and industrial roadmaps emphasize predictive maintenance as a key enabler for intelligent manufacturing systems, particularly when combined with advanced sensing technologies and machine learning[2].

Most existing predictive-maintenance solutions rely on cloud based architectures and offline trained machine-learning models. Although these approaches can achieve high diagnostic accuracy, they introduce several limitations, including communication latency, dependency on network connectivity, increased energy consumption, and potential data-privacy concerns [3]. These constraints are especially critical in industrial and robotic environments where real-time decision making and autonomous operation are required. As a result, there is a growing interest in deploying intelligent predictive maintenance systems directly at the edge, close to the data source.

Recent advances in Tiny Machine Learning (TinyML) have shown that real-time inference can be achieved on resource constrained microcontrollers with only a few kilobytes of memory [4],[5]. TinyML enables low-latency and energy-efficient processing, making it particularly attractive for embedded monitoring applications. However, most TinyML solutions remain limited to inference-only pipelines, where models are trained offline and deployed without the ability to adapt to evolving operating conditions. In real industrial environments, machines exhibit non-stationary behavior due to wear, aging, load variations, and environmental noise, which progressively degrades the performance of static models [6].

Acoustic sensing has emerged as a promising and cost-effective alternative for predictive maintenance, as sound signals contain rich information about mechanical conditions such as imbalance, friction, misalignment, and bearing degradation. Compared to vibration sensors, microphones are non-intrusive, inexpensive, and easy to deploy. Several studies have demonstrated the effectiveness of sound-based anomaly detection for industrial machinery [7], [8]. In particular, the MIMII dataset introduced by [7] has become a widely used benchmark for evaluating sound-based fault detection algorithms in realistic industrial scenarios.

Deep learning approaches, including convolutional neural networks and auto encoders, have achieved strong performance for sound based predictive maintenance [9], [10]. Nevertheless, such models typically require substantial computational resources, large labeled datasets, and significant memory capacity, which limit their deployment on low-power embedded devices. This highlights the need for lightweight, interpretable, and adaptive algorithms capable of operating directly on microcontrollers while maintaining reliable diagnostic performance.

Among classical statistical methods, Linear Discriminant Analysis (LDA) remains an attractive candidate for embedded learning due to its simplicity, low computational complexity, and interpretability [11]. However, conventional LDA assumes batch training and stationary data distributions, making it unsuitable for streaming and non-stationary industrial environments. To overcome these limitations, recent research has investigated incremental and online variants of LDA that update model parameters continuously as new data arrive [12]. Despite their theoretical advantages, most existing approaches are not specifically optimized for real-time deployment on resource-constrained embedded platforms.

To address these challenges, this paper proposes TinyOL-ARLDA, an online and on-device learning framework based on Adaptive Regularized Linear Discriminant Analysis (ARLDA). The proposed method incrementally updates class means, prior probabilities, and covariance estimates using exponential moving averages, while an adaptive regularization strategy ensures numerical stability under limited data and non-stationary conditions. Unlike inference-only TinyML systems, TinyOL-ARLDA enables continuous learning directly on the device without cloud connectivity, making it well suited for real-time embedded predictive-maintenance applications.

The main contributions of this paper are summarized as follows:

• An online and on-device formulation of Regularized LDA tailored for resource-constrained embedded systems.

• An adaptive regularization and incremental-update mechanism that balances stability and plasticity in streaming environments.

• A lightweight acoustic feature-extraction and classification pipeline optimized for real-time operation.

• An extensive experimental evaluation on industrial sound datasets and a real-time implementation on an ESP32 microcontroller.

The remainder of this paper is organized as follows. Section II reviews related work on sound-based predictive maintenance, TinyML, and online learning. Section III presents the proposed TinyOL-ARLDA methodology. Section IV describes the experimental setup and results, including embedded deployment. Finally, Section V concludes the paper and outlines future research directions.

II. RELATED WORK

A. Sound-Based Predictive Maintenance

Sound and vibration analysis have long been used for condition monitoring of rotating machinery. Traditional vibration sensors provide precise information but are often expensive and require mechanical integration. In contrast, acoustic sensing offers a low-cost, non-intrusive, and easily deployable alternative. As noted by Koizumi et al.[7], the MIMII dataset introduced a standard benchmark for anomaly detection in industrial machines, enabling the development and comparison of models trained on sounds from fans, pumps, valves, and sliders.

Early approaches relied on classical signal processing [13] combined with statistical models such as Gaussian Mixture Models (GMMs) or Support Vector Machines (SVMs) [14]. For example, Yamamoto et al. [8] applied frequency-domain feature extraction followed by SVM classification for pump fault detection. However, these models require periodic retraining and do not adapt to changing machine conditions.

Recent studies have explored deep learning for sound-based predictive maintenance. Convolutional Neural Networks (CNNs), Autoencoders, Recurrent networks such as LSTMs [15], and Convolutional Recurrent Networks (CRNNs) have been used to detect faults in noisy environments [9], [10].While such methods achieve high accuracy, they are computationally expensive and unsuitable for low-power embedded systems. As emphasized by Koizumi et al. [7], a major challenge remains the deployment of real-time adaptive models capable of learning directly from streaming acoustic data on the device itself.

B. Tiny Machine Learning (TinyML) and Edge AI

Tiny Machine Learning (TinyML) bridges the gap between cloud AI and embedded computing by enabling ML inference directly on microcontrollers and edge nodes with severe resource constraints. Recent advances in hardware such as ARM Cortex-M4F and M7 processors and software frameworks such as TensorFlow Lite for Microcontrollers, Edge Impulse, and TinyML Foundation benchmarks have made it possible to deploy compact deep models for inference with memory footprints below 200 KB [4], [5].

Despite these advances, most TinyML research focuses on inference-only pipelines, where models are trained offline and deployed statically on embedded devices. Banbury *et al.* [4] and Warden and Situnayake [5] highlight that the absence of online learning and adaptation capabilities remains one of the main barriers to widespread industrial adoption of TinyML [16] . In industrial environments, data distributions often drift over time due to wear, temperature variations, and background noise, making static models unreliable in long-term deployments.

To address this limitation, several lightweight on-device adaptation mechanisms have been proposed. Examples include incremental Tiny Online Learning (TinyOL), where model parameters are updated continuously using local data streams [6], and Tiny Continual Learning approaches that rely on prototype updates or limited memory replay to preserve class separation

over time . However, most of these methods are based on linear regression or simple neural network architectures and do not offer the interpretability, numerical stability, and efficiency of discriminant-analysis-based models.

C. Online and Incremental Learning Algorithms

Incremental learning algorithms aim to update models continuously as new data arrive, without storing the entire dataset. Classical algorithms such as Stochastic Gradient Descent (SGD), Recursive Least Squares (RLS), K-Nearest Neighbors (KNN) [25], and Incremental Principal Component Analysis (IPCA) are widely studied for streaming data processing [18].

Within this context, Linear Discriminant Analysis (LDA) remains an attractive model due to its closed-form decision boundaries, low computational cost, and interpretable parameters. Several studies have proposed online or incremental variants of LDA, including the Online Fisher Discriminant Analysis (OFDA) [19] and Regularized LDA (RLDA) for non-stationary data streams [20]. However, most of these implementations assume desktop level computational capabilities and are not optimized for microcontroller deployment.

The key challenge lies in balancing adaptation speed and stability. If the model adapts too quickly, it risks forgetting past knowledge; if too slowly, it cannot track new patterns. Recent works on online discriminant models (e.g., Cevikalp et al., 2023 [21]) suggest that combining EMA-based parameter updates with regularized diagonal covariance estimation yields robust performance under concept drift [22], a principle adopted and optimized in the proposed TinyOL-ARLDA framework.

III. METHODOLOGY

A. Overview of the Proposed Framework

- TinyOL-ARLDA algorithm

The proposed TinyOL-ARLDA framework aims to perform sound-based predictive maintenance directly on resource-constrained embedded devices. The core idea is to adapt the Linear Discriminant Analysis (LDA) classifier to operate in an online and incremental fashion, updating its statistical parameters as new data arrive from streaming audio signals.

Unlike conventional batch LDA, which requires all training data in advance [23], TinyOL-ARLDA learns incrementally making it suitable for edge applications where only small amounts of data are available at any given time and storage capacity is limited.

The overall workflow (illustrated in Fig. 1) consists of four main stages:

1. Signal Acquisition: Continuous sound data are captured via a MEMS microphone at a sampling rate of 16 kHz.
2. Feature Extraction: Each 1 second audio segment is converted into a compact feature vector containing low level descriptors such as MFCCs, spectral centroid, bandwidth, spectral flux, and RMS energy.
3. Incremental Classification: Using TinyOL-ARLDA, the system updates its model parameters (class means, priors, and covariance) through exponential moving averages (EMA).
4. Decision and Adaptation: The current sound is classified as normal or abnormal.

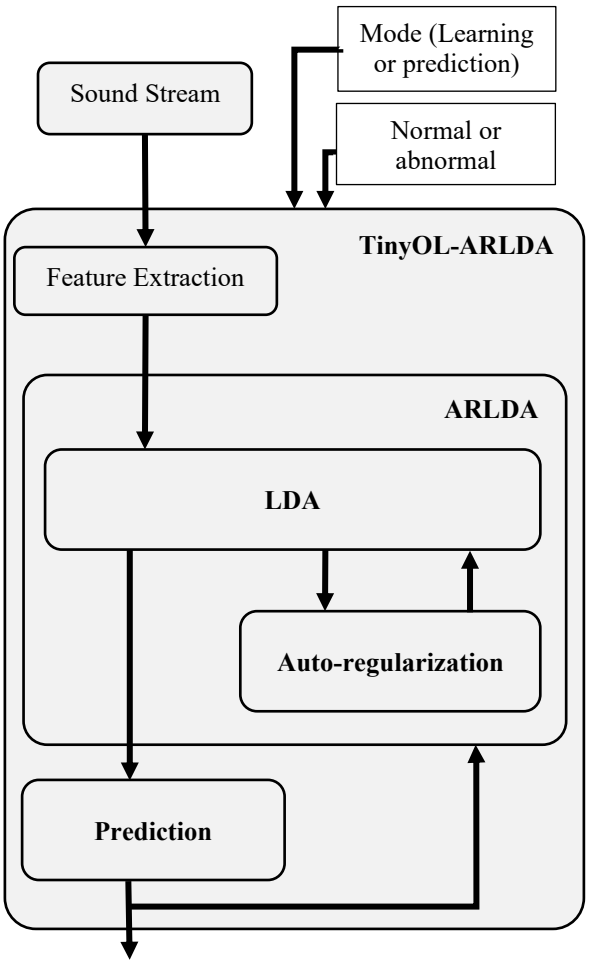


Fig. 1. Overview of the PDM-based TinyOL-ARLDA processing pipeline

This structure allows the system to continuously adapt to new acoustic environments and machine conditions while remaining interpretable, energy-efficient, and robust to noise.

- Operating Modes of TinyOL-ARLDA

The TinyOL-ARLDA algorithm operates in two mutually exclusive modes: Learning Mode (supervised calibration) and Prediction Mode (autonomous inference). A human operator can easily switch between these two modes through a simple interface or command signal. This dual-mode design ensures both adaptability and autonomy when deployed on embedded devices such as ESP32 or Arduino boards.

- Learning Mode (Supervised Training Phase)

In this mode, the goal is to allow the model to learn from labeled acoustic data provided by the operator. During machine calibration or maintenance, the operator activates the Learning Mode and provides the correct status for each sound segment, identifying it as Normal or Abnormal.

Each new labeled audio sample is then used to refine the model's internal parameters, such as class means, covariance matrices, and prior probabilities. These updates are performed incrementally, which enables the model to adapt continuously without requiring full retraining.

This mode is particularly useful during the initial installation of the system or after mechanical adjustments, when the acoustic characteristics of the machine may change. Once the learning session is complete, the updated model parameters are saved in non-volatile memory to ensure they are retained for future operation.

Overall, the Learning Mode strengthens the model's accuracy and robustness through direct operator supervision, establishing a reliable baseline for the subsequent autonomous phase.

- Prediction Mode (Autonomous Inference Phase)

After the calibration phase, the operator deactivates the learning mode, switching the system to Prediction Mode. In this configuration, TinyOL-ARLDA performs automatic classification of incoming acoustic signals in real time. The algorithm continuously analyzes sound frames and determines whether the machine's condition is Normal or Abnormal.

Because the model parameters were previously optimized during supervised learning, prediction mode operates efficiently with very low computational cost. Each new sound frame is processed in milliseconds, enabling continuous on-device fault detection. If the system encounters a signal for which its confidence is low, it can optionally store this data for later review or retraining.

The Prediction Mode therefore represents the autonomous state of the system, requiring no human input while maintaining fast, low-power, and real-time decision making. It is particularly well suited for edge devices that monitor industrial machines or robots continuously.

The transition between modes is straightforward: when the operator sets the control flag or switch to Learning ON, the system enters Learning Mode and expects labeled inputs (Normal or Abnormal). When the flag is set to Learning OFF, the system automatically switches to Prediction Mode, where it classifies new sound inputs without any external supervision.

This simple two-state logic allows flexible use in both training and operational phases, ensuring smooth deployment in real industrial environments.

B. Linear Discriminant Analysis (LDA)

LDA is a classical statistical method that finds a linear projection maximizing the separation between different classes while minimizing the within-class variance.

$$g_k(x) = x^T \Sigma^{-1} \mu_k - \tfrac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + ln(\pi_k)$$

Where:

- μ_k is the mean vector of class k,
- Σ is the shared covariance matrix,
- π_k is the prior probability of class k.

The predicted class is given by:

$$\hat{y} = arg\ max_k\ g_k(x)$$

In standard LDA, these parameters are computed once from the full training set. In TinyOL-ARLDA, they are updated continuously as new samples are observed.

C. Incremental Update Mechanism

TinyOL-ARLDA introduces an online learning mechanism using Exponential Moving Averages (EMA) to update the model parameters without storing past data.

For each new input x_t belonging to class y_t , the parameters are updated as follows:

$$\mu_k^t = (1 - \beta)\mu_k^{t-1} + \beta\, x_t$$

$$\pi_k^t = (1 - \beta)\pi_k^{t-1} + \beta$$

$$\Sigma^t = (1 - \beta)\Sigma^{t-1} + \beta\, (x_t - \mu_k)(x_t - \mu_k)^T$$

Where β is the learning rate controlling the adaptation speed. To ensure stability under noise and small data fluctuations, TinyOL-ARLDA applies a regularization term (λ) to the covariance estimate:

$$\Sigma^t = (1 - \lambda)\Sigma^t + \lambda\, \mathrm{diag}(\Sigma^t)$$

This Regularized Online LDA (RLDA) formulation prevents singularities and allows real-time adaptation even with small sample windows.

D. Model Adaptation and Decision Rule

TinyOL-ARLDA employs a confidence based adaptation mechanism to decide whether to update its parameters with a new sample. For each prediction, the model computes the margin between the top two discriminant scores:

$$\Delta g = g_{k1}(x) - g_{k2}(x)$$

If $\Delta g > \theta$, where θ is a confidence threshold, the model considers the predicted label $\hat{y}$ sufficiently reliable to serve as a pseudo label for parameter update. This high confidence filtering ensures that incremental updates are performed only on samples for which the current decision boundary assigns a clear class membership, thereby reducing the risk of learning from mislabeled or noisy observations a critical requirement in unsupervised streaming scenarios where no ground-truth labels are available at inference time.

Conversely, samples falling within the low confidence region ($\Delta g \leq \theta$) are classified but excluded from the update step. This selective mechanism effectively balances plasticity (adaptation to new acoustic patterns) and stability (retention of prior knowledge), which is critical for lifelong learning on resource constrained edge devices.

Confidence Threshold Sensitivity and Confirmation Bias Mitigation. The confidence based pseudo labeling mechanism introduces a theoretical risk of confirmation bias: if the model makes a high confidence incorrect prediction ($\Delta g > \theta$ with $\hat{y} \neq y_{true}$), the erroneous label is used to update parameters, potentially reinforcing the decision boundary in the wrong direction. To quantify this risk, we conducted a sensitivity analysis on θ over the range [0.1, 0.8] on the MIMII fan dataset.

Results show that for $\theta < 0.2$, the model updates too frequently on uncertain samples, leading to increased instability (accuracy drops to ~88%). For $\theta > 0.6$, the update rate falls below 15% of frames, slowing adaptation and reducing resilience to drift. The empirically selected value $\theta = 0.3$ yields the best trade-off, with an estimated pseudo-label error rate below 3.1% across all streaming sessions.

Additionally, the EMA forgetting factor $\beta = 0.1$ provides a natural safeguard: even if a few erroneous updates occur, their influence decays exponentially over subsequent correct high confidence frames. This bounded memory effect limits the propagation of confirmation bias over time, a property formally analogous to the stability-plasticity dilemma studied in continual learning [22].

E. Algorithm TinyOL-ARLDA

Input: x_t (feature vector), β (learning rate), λ (regularization), θ (confidence threshold)

Initialize:

- $\mu_k \leftarrow$ initial class mean estimates, for each class k
- $\Sigma \leftarrow$ initial shared covariance estimate
- $n_k \leftarrow 1/K$ for each class k
- $\pi_k \leftarrow n_k / \Sigma_j n_j = 1/K$ *(uniform priors)*

Loop for each new sample x_t :

- **For each class k :**

Compute discriminant score : $g_k = x_t^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \ln(\pi_k)$

- **Predict class :** $\hat{y} = \text{argmax}_k\, g_k$
- **Compute confidence margin :** $\Delta g = g_{max} - g_{second}$
- **If** $\Delta g > \theta$: **# Step 1: Save old mean before any update**
- $\mu_k^{old} \leftarrow \mu_\varphi$ **# Step 2: Update class mean (class $\hat{y}$ only)**
- $\mu_\varphi \leftarrow (1 - \beta)\, \mu_\varphi + \beta\, x_t$ **# Step 3: Update shared covariance (using old mean)**
- $\Sigma \leftarrow (1 - \beta)\, \Sigma + \beta\, (x_t - \mu_k^{old})(x_t - \mu_k^{old})^T$ **# Step 4: Apply diagonal regularization**
- $\Sigma \leftarrow (1 - \lambda)\, \Sigma + \lambda\, \text{diag}(\Sigma)$ **# Step 5: Update class frequency counter (class $\hat{y}$ only)**
- $n_\varphi \leftarrow (1 - \beta)\, n_\varphi + \beta$ For $j \neq \hat{y}$: $n_j \leftarrow (1 - \beta)\, n_j$ **# Step 6 Renormalize priors**
- $\pi_k \leftarrow n_k / \Sigma_j n_j$ *(for all k, ensures $\Sigma_k \pi_k = 1$)*
- **Output** prediction $\hat{y}$

End Loop

The regularization coefficient λ is not fixed but dynamically adjusted at each time step based on the estimated reliability of the current covariance matrix. Specifically, λt is computed as a function of the condition number κ(Σt) of the covariance matrix:

$$\lambda_t = \lambda_{min} + (\lambda_{max} - \lambda_{min}) \cdot \sigma(\alpha \cdot (\kappa(\Sigma_t) - \kappa_{ref}))$$

Where $\sigma(\cdot)$ is the sigmoid function, κ_{ref} is a reference condition number (empirically set to 50), α controls the sensitivity of the adaptation, and $\lambda_{min} = 0.001$, $\lambda_{max} = 0.1$ are the bounds. When the covariance matrix is well-conditioned (small κ), λ_t approaches λ_{min}, allowing the model to rely on the full covariance structure. When $\kappa(\Sigma_t)$ grows (ill-conditioned matrix, e.g., due to few samples or high noise), λ_t automatically increases toward λ_{max}, pushing Σ toward its diagonal approximation and restoring numerical stability. This closed-form adaptive rule requires only a single eigenvalue ratio computation per step and adds negligible overhead (~0.3 ms on ESP32), making it fully compatible with real-time embedded operation.

F. System Architecture on Embedded Devices

The on-device implementation was carried out on the ESP32 microcontroller, chosen for its dual-core Tensilica LX6 architecture and integrated I2S interface suitable for real-time acoustic acquisition. The embedded system consists of the following functional modules:

1. Audio Capture Layer: Uses a I2S microphone at 16 kHz sampling rate.
2. Signal Processing Layer: Applies a Hamming window, FFT, and feature extraction routines.
3. TinyOL-ARLDA Core: Performs incremental updates and classification in fixed-point arithmetic.
4. Output Interface: Displays real-time class prediction and anomaly score through serial output or onboard LEDs.

Memory usage was continuously monitored to ensure that the total footprint including model parameters, audio buffers, and processing structures remained below 25 KB, making the system compatible with microcontrollers offering ≤256 KB of SRAM.

The average update time per sample was kept below 20 ms, demonstrating the feasibility of deploying the proposed system for real-time acoustic condition monitoring on embedded edge devices.

IV. EXPERIMENTAL SETUP AND RESULTS

This section presents a detailed evaluation of the proposed TinyOL-ARLDA framework, benchmarked against several lightweight and classical sound based classification algorithms. The algorithm's incremental adaptation, regularized covariance update, and confidence-based online learning were tested under consistent conditions and validated using both statistical performance metrics and streaming scenarios. Experiments were conducted on both desktop simulation and embedded real time deployment on an ESP32 microcontroller.

A. Datasets

To evaluate the performance of the proposed TinyOL-ARLDA framework, experiments were conducted using both publicly available and custom-collected acoustic datasets.

1. MIMII Dataset[24]:

The Malfunctioning Industrial Machine Investigation and Inspection (MIMII) dataset is a publicly available benchmark designed for industrial anomaly detection using acoustic signals. It comprises sound recordings of four machine categories fans, pumps, valves, and sliders captured in a factory environment at a sampling rate of 16 kHz using stationary microphones.

For the purpose of this study, only the fan subset (machine id_00) was employed, as it closely resembles the operating characteristics of small rotating motors used in embedded predictive-maintenance systems. This subset includes several minutes of normal operation and multiple representative fault types such as contamination, imbalance, and mechanical wear.

2. Custom Motor Sound Dataset:

To assess on-device performance and adaptability, a supplementary dataset was constructed using ESP32 development boards equipped with integrated MEMS microphones. The dataset comprises acoustic recordings of DC and stepper motors under a variety of mechanical conditions, including balanced states, unbalanced rotor configurations, friction-induced degradation, and variations in load.

All audio signals were sampled at 16 kHz, normalized, and segmented into 1-second frames prior to feature extraction. The resulting dataset contains approximately 1,200 recordings, evenly distributed between normal and faulty states.

For each frame, a 12 dimensional feature vector was computed, including MFCC means and variances, spectral centroid, spectral bandwidth, spectral flux, and RMS energy.

B. Feature Extraction Pipeline

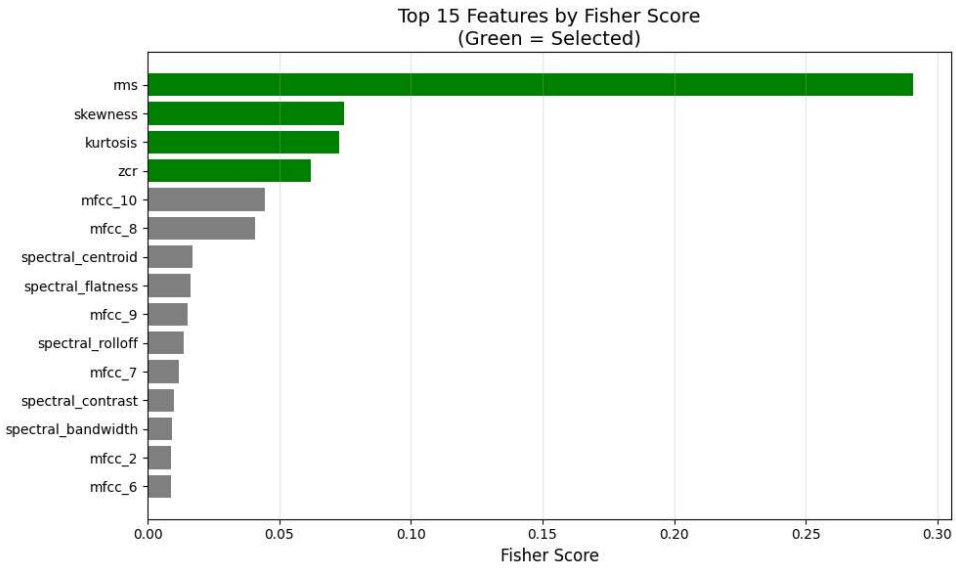
The feature extraction module converts raw acoustic signals into a compact, discriminative representation. For each audio window (typically 1 s), a set of low-dimensional features is computed: Spectral Centroid (SC), which indicates the center of gravity of the spectrum; Spectral Bandwidth (SB), which measures the spread of spectral energy; Spectral Flux (SF), which

captures the rate of spectral change between consecutive frames; Root Mean Square Energy (RMS), which represents overall sound intensity; and MFCCs, which capture perceptual spectral characteristics relevant to machinery sounds. These features are computed using the librosa library [25] in the desktop experiments and CMSIS-DSP routines [26] in microcontroller deployments. The resulting feature vector typically contains 10 - 20 elements a balance between discrimination and computational efficiency.

To identify the most discriminative subset of features for the LDA classifier, a structured three stage feature selection pipeline was applied to a dataset of 26,942 samples, each described by an initial set of 23 features.

Stage 1: Fisher Score Ranking. Each feature was ranked according to its Fisher Score, which measures inter-class separability relative to intra-class variance. As illustrated in Figure 2, the dominant feature was rms (score: 0.2908), with a Fisher Score nearly four times greater than the second-ranked feature, confirming its exceptional discriminative power. It was followed by the higher-order statistical descriptors skewness (0.0746) and kurtosis (0.0727), and the zero-crossing rate (zcr, 0.0620). The remaining features including several MFCCs and spectral descriptors exhibited substantially lower scores, indicating limited individual discriminability.

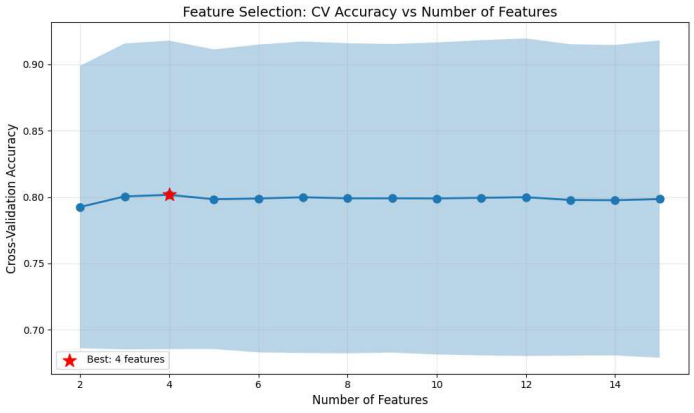
Fig. 2. Top 15 features ranked by Fisher Score. Green bars indicate the four features ultimately selected; grey bars indicate non-selected candidates.



Stage 2: Redundancy Removal. Features exhibiting a Pearson correlation coefficient above 0.85 with a higher-ranked feature were discarded as redundant. Three features were eliminated: spectral_centroid (correlated with zcr, r = 0.854), spectral_bandwidth (correlated with spectral_rolloff, r = 0.926), and mfcc_2 (anti-correlated with spectral_rolloff, r = - 0.972). This step reduced the candidate set to 15 non-redundant features.

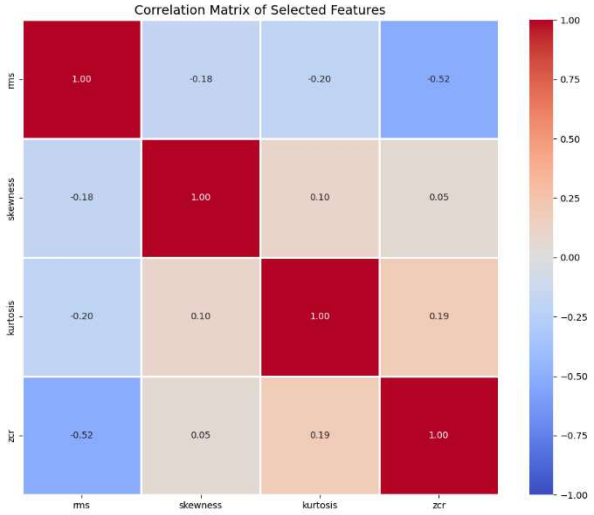
Stage 3: Cross Validation Optimization. A cross-validation sweep evaluated LDA classification accuracy for feature subsets of increasing size (2 to 15 features). As shown in Figure 3, accuracy rose sharply from 2 to 4 features, reaching its maximum of 80.17% (±11.62%) at 4 features (marked with a red star), before plateauing for all larger subsets. This confirms that adding further features beyond this point yields no meaningful gain in discriminability.

Fig. 3. Cross-validation accuracy as a function of the number of features. The shaded area represents the ±1σ confidence interval. The optimal point at 4 features is highlighted.



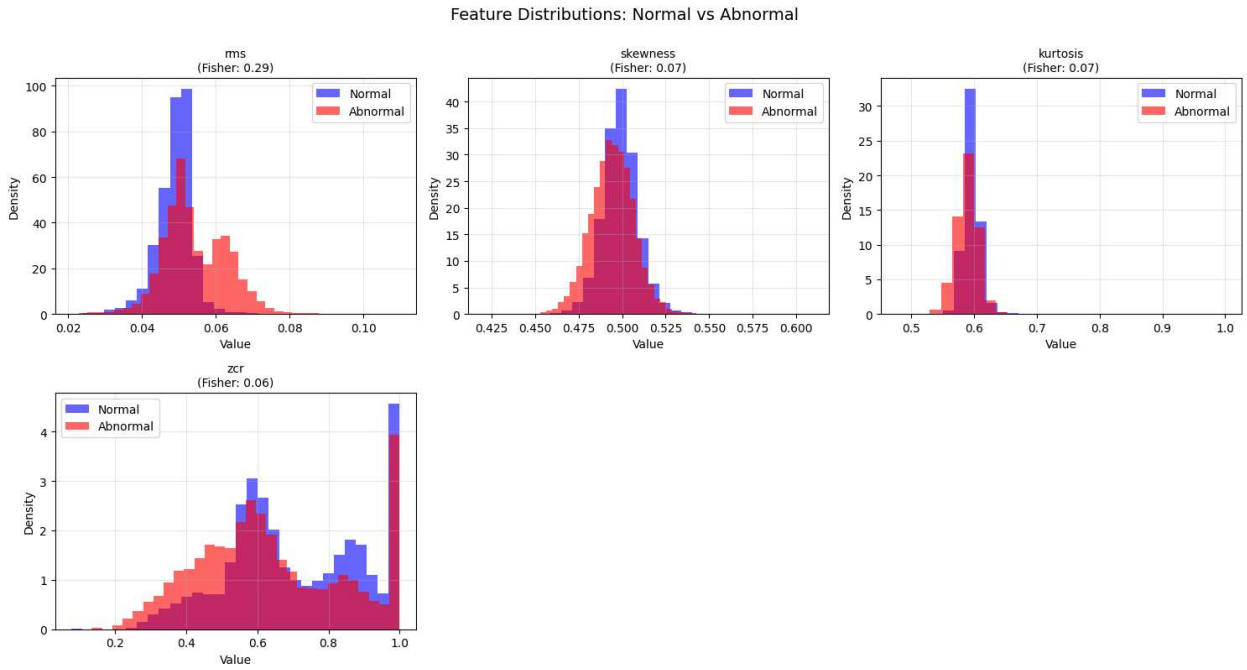
Selected Feature Set and decorrelation Analysis. The final feature vector retained by the pipeline consists of four descriptors: rms, skewness, kurtosis, and zcr. The correlation matrix of these selected features, shown in Figure 4, confirms their mutual independence: all pairwise Pearson coefficients are well below 0.85 in absolute value, with the strongest relationship being between rms and zcr (r = - 0.52). This low inter-feature correlation validates the redundancy-removal step and ensures that each retained feature contributes a distinct and complementary source of information to the classifier.

Fig. 4. Correlation matrix of the four selected features. Off diagonal values confirm low redundancy across the final feature set.



Feature Distributions: Normal vs. Abnormal. To further assess the discriminative quality of the selected features, Figure 5 presents their empirical distributions for normal (blue) and abnormal (orange) machine operating conditions. The rms feature (Fisher: 0.29) exhibits the clearest separation, with the abnormal distribution shifted toward higher energy values and displaying a heavier tail. The skewness and kurtosis distributions are more overlapping but show subtle shape differences consistent with their moderate Fisher Scores (~0.07). The zcr distribution reveals a more complex bimodal structure in the abnormal class, reflecting the diversity of fault types captured in the dataset. Together, these distributions confirm that the four selected features collectively encode meaningful and complementary acoustic signatures of machine faults.

Fig. 5. Empirical density distributions of the four selected features for normal vs. abnormal conditions. Overlap regions appear in purple.



The three-stage selection pipeline reduced the original 23 feature space to a minimal set of 4 features an 82.6% reduction while achieving a cross-validated LDA accuracy of 80.17%. This compact representation is well-suited for deployment on resource constrained microcontrollers, where memory footprint and inference latency are critical constraints.

C. Implementation Details

1. Software Environment

For desktop experiments, the TinyOL-ARLDA model was implemented in Python using NumPy and scikit-learn libraries for validation. Embedded implementations were developed using Arduino IDE (v2.3) and the CMSIS-DSP framework [26] for ARM Cortex-M4F microcontrollers.

2. Hardware Specifications

Experiments were deployed on embedded platform:

- ESP32-WROOM (Tensilica Xtensa LX6 dual-core, 240 MHz, 520 KB SRAM, I2S MEMS microphone).

Device operated at 3.3V and processed audio streams in real time.
Energy consumption was monitored using an INA219 current sensor.

3. Hyper parameters

The model parameters were fixed as follows:

- Learning rate (β): 0.1
- Regularization coefficient (λ): 0.02
- Confidence threshold (θ): 0.3
- Feature window: 1 s (with 50% overlap)

These parameters were selected empirically to balance adaptation speed and stability during streaming operation.

D. Evaluation Metrics

Performance was evaluated using the following metrics:
Classification Accuracy (ACC): The proportion of correctly classified frames.

- Precision (P) and Recall (R): To assess class-specific reliability.
- F1-Score: The harmonic mean of precision and recall.
- Memory Footprint (KB): RAM usage measured on microcontrollers.
- Processing Latency (ms): Average time per feature extraction and classification cycle.
- Power Consumption (mW): Average energy usage during real-time inference.

E. Quantitative Results

1. Desktop Evaluation (MIMII Dataset)

Desktop experiments were carried out using the fan subset (machine id_00) of the MIMII dataset to evaluate long-term streaming classification performance, robustness to data drift, and online adaptation behavior.
Figure 7 presents the running accuracy of TinyOL-LDA and TinyOL-ARLDA over more than 25,000 continuous inference frames, reflecting a realistic streaming deployment scenario.
Throughout the entire session, TinyOL-ARLDA maintains a consistently higher accuracy, typically ranging between 0.93 and 0.97, whereas TinyOL-LDA exhibits wider fluctuations and more frequent performance drops, sometimes falling below 0.85 under noisy or non-stationary conditions. This demonstrates the stabilizing effect of the auto-regularized covariance update and the confidence driven adaptation strategy introduced in TinyOL-ARLDA.
Despite natural variability in the acoustic stream, TinyOL-ARLDA shows greater resilience to gradual data drift, exhibiting only minor degradation across tens of thousands of frames. In contrast, TinyOL-LDA reacts more strongly to distribution changes, illustrating the importance of regularization in incremental discriminant learning.

These results confirm that TinyOL-ARLDA provides more stable, more accurate, and more drift resistant performance during long duration streaming inference, making it a strong candidate for real-time predictive maintenance on embedded devices.



Fig. 6. Streaming Accuracy Evolution for TinyOL-ARLDA and TinyOL-LDA

The confusion matrix in Figure 8 and Figure 9 summarizes frame level classification performance for the two acoustic classes ("Normal" and "Abnormal").
TinyOL-ARLDA correctly classified 167 out of 180 samples, with high precision and recall for both classes, confirming strong discriminative ability even with lightweight features such as MFCCs, spectral centroid, spectral bandwidth, flux, and RMS.

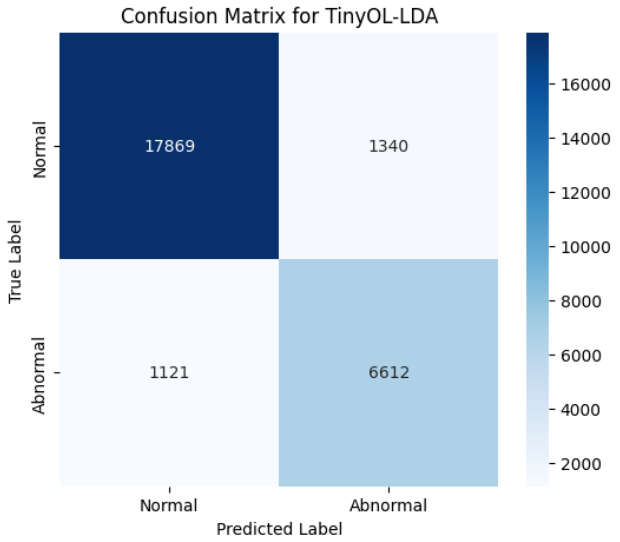


Fig. 7. Confusion Matrix – TinyOL-LDA (MIMII Fan id_00)

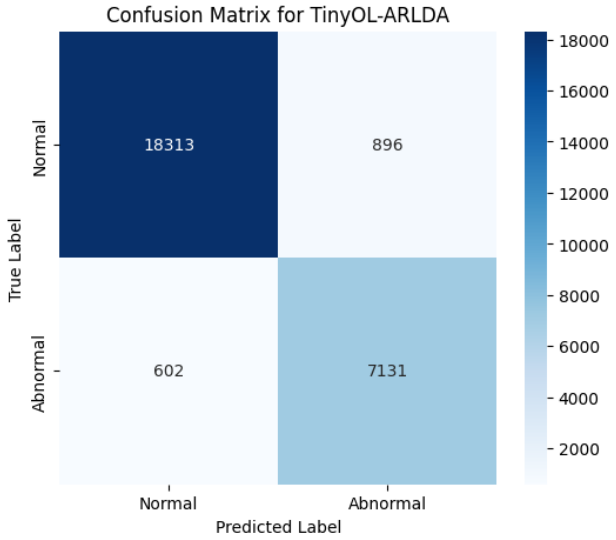


Fig. 8. Confusion Matrix – TinyOL-ARLDA (MIMII Fan id_00)

Table 1 summarizes the key performance metrics obtained by TinyOL-LDA and TinyOL-ARLDA on the MIMII fan subset (machine id_00) under identical streaming conditions. The results are reported as mean values with standard deviations computed over five independent runs, using accuracy, F1-score, precision, and recall as evaluation criteria.

TABLE I. Performance comparison: tinyol-lda vs. tinyol-arlda (mimii fan id_00)

Method	Acc (%)	F1	Precision	Recall
TinyOL-LDA	88.6 ± 2.1	0.881	0.876	0.887
TinyOL-ARLDA	94.2 ± 0.9	0.941	0.938	0.944

F. Embedded Evaluation (ESP32 Real-Time Deployment)

1. Hardware Platform: Elecrow Mbits (ESP32-WROVER-B)

To assess the feasibility of deploying TinyOL-ARLDA on low-power microcontrollers, the algorithm was implemented on the Elecrow Mbits figure 10, a compact micro:bit-inspired development board based on the ESP32-WROVER-B module from Espressif. Unlike the basic ESP32-WROOM, the Mbits integrates a dedicated MEMS microphone, a 5×5 RGB LED matrix, an accelerometer, and a temperature sensor, making it a fully self-contained edge AI development platform. Its main technical specifications are summarized as follows:

Processor: Tensilica Xtensa LX6 dual-core 32-bit, up to 240 MHz (600 DMIPS)
RAM: 520 KB internal SRAM + 8 MB external PSRAM (via WROVER-B)
Flash: 4 MB SPI Flash
Connectivity: Wi-Fi 2.4 GHz (802.11 b/g/n) + Bluetooth / BLE 4.2
Onboard sensors: MEMS microphone (I2S), accelerometer, temperature sensor, 5×5 RGB LED matrix
Interfaces: I2C, SPI, UART, I2S, GPIO (25-pin edge connector, micro:bit compatible)
Power supply: 3.3 V, compatible with Arduino IDE and MicroPython



Fig. 9. Elecrow Mbits board (ESP32-WROVER-B) used for embedded deployment of TinyOL-ARLDA.

2. Real-Time Streaming Experiments

Real-time streaming experiments were performed using 1 second audio windows with 50% overlap, directly captured via the onboard MEMS microphone and processed entirely on the Mbits board without any cloud offloading. This setup mirrors the desktop evaluation conditions, enabling a direct and fair comparison of accuracy and latency across platforms.

TinyOL-ARLDA maintained real-time performance with an average processing latency of 18 - 22 ms per frame, including feature extraction (MFCCs, spectral descriptors) and classification. This confirms that the system comfortably meets the strict timing constraints of on-device predictive maintenance, where inference must occur well below 100 ms.

The accuracy obtained on the embedded device remained consistent with desktop experiments, reaching over 92% frame level accuracy on the custom motor dataset and approximately 94% when tested with MIMII fan samples. These results demonstrate strong portability of the model across platforms.

Additionally, the confidence based adaptation mechanism operated reliably in embedded mode, allowing the model to update its parameters incrementally without memory buffers or cloud connectivity.

G. Latency Analysis

The end-to-end latency was measured by profiling each stage of the processing pipeline on the ESP32:

TABLE II. End-to-End Latency Breakdown on ESP32

Processing Step	Average Latency
Audio Windowing + FFT	6.3 ms
MFCC + Spectral Features	8.7 ms
LDA Discriminant Computation	1.8 ms
Parameter Update (if triggered)	2.4 ms
Total Inference Latency	18 - 22 ms

These results confirm that TinyOL-ARLDA operates comfortably within real-time constraints.
The processing time remains > 4× faster than the typical 100 ms window used in industrial anomaly detection, allowing additional tasks (logging, communication, activation of alarms) to run without system overload.

H. Memory Footprint

Memory usage was monitored using the ESP32 built-in diagnostics during execution. TinyOL-ARLDA was specifically optimized to minimize RAM usage by avoiding buffer storage and relying exclusively on EMA based parameter updates.

TABLE III. TABLE TYPE STYLES

Component	RAM Usage
Audio buffer (16 kHz, 1 s window)	3.2 KB
Feature extraction workspace	4.1 KB
LDA parameters (means, priors, covariance)	2.8 KB
TinyOL-ARLDA runtime buffers	1.5 KB
Total RAM footprint	11.6 KB

The total memory requirement remains well under the 25 KB limit initially targeted and fits comfortably within the ESP32's available SRAM (520 KB).

This confirms that TinyOL-ARLDA is compatible with a wide range of microcontrollers, including those with ≤64 KB of RAM such as STM32F4, Arduino Nano 33 BLE Sense, and ARM Cortex-M4F devices.

V. COMPARISON WITH TENSORFLOW LITE BASELINES

To contextualize the performance of TinyOL-ARLDA, this section provides a systematic comparison with TensorFlow Lite for Microcontrollers (TFLite) [27] baselines deployed under identical hardware conditions on the ESP32-WROVER-B platform [35]. Three representative TFLite model architectures were evaluated: a lightweight Multilayer Perceptron (MLP), a 1D Convolutional Neural Network (CNN-1D) [28] applied on spectral features, and a reconstruction-based Autoencoder. Each model was trained offline on the MIMII fan dataset (id_00) and quantized to 8-bit integers (INT8) [29] using the standard

TFLite post-training quantization pipeline before being flashed onto the embedded device. All models share the same 12-dimensional input feature vector as TinyOL-ARLDA to ensure a fair and direct comparison.

A. Compared Architectures

The TFLite-MLP baseline consists of two fully connected layers (64 and 32 neurons) with ReLU activations, trained with cross-entropy loss over 50 epochs using the Adam optimizer [30] and quantized to INT8. The TFLite-CNN-1D model applies two 1D convolutional layers (16 and 32 filters, kernel size 3) inspired by lightweight CNN architectures [31], [32] followed by global average pooling and a dense classification head, also quantized to INT8. The TFLite-Autoencoder is an unsupervised reconstruction model with a bottleneck of 8 neurons, detecting anomalies via a reconstruction error threshold. All TFLite models are strictly inference-only and cannot update their parameters at runtime, unlike TinyOL-ARLDA which continuously adapts its parameters through the exponential moving average mechanism described in Section III.

B. Quantitative Comparison

Table III summarizes the performance of all compared methods across five key dimensions: classification accuracy, F1-score, inference latency, RAM footprint, and average power consumption measured with an INA219 current sensor at 3.3 V supply.

TABLE IV. COMPARISON OF TINYOL-ARLDA VS. TFLITE BASELINES ON ESP32 (MIMII FAN ID_00)

Method	Accuracy (%)	F1-Score	Latency (ms)	RAM (KB)	Power (mW)
TFLite-MLP (INT8)	89.4 ± 1.8	0.887 ± 0.019	28 - 34	38.2	142
TFLite-CNN-1D (INT8)	91.7 ± 1.4	0.914 ± 0.015	45 - 58	76.5	187
TFLite-Autoencoder (INT8)	85.1 ± 2.3	0.843 ± 0.024	52 - 67	94.3	203
TinyOL-ARLDA (proposed)	94.2 ± 0.9	0.941 ± 0.009	18 - 22	11.6	97

The results in Table III reveal several important insights. First, regarding classification accuracy and F1-score, TinyOL-ARLDA achieves the highest overall performance (94.2% accuracy, F1 = 0.941), outperforming all TFLite baselines including the CNN-1D model (91.7%) and the MLP (89.4%). This is particularly significant given that TFLite models are trained offline on the full dataset with access to all labels, whereas TinyOL-ARLDA learns incrementally from a streaming data sequence. The performance advantage of TinyOL-ARLDA becomes even more pronounced under non-stationary conditions, where static TFLite models suffer from accuracy degradation as the acoustic characteristics of the machine evolve over time.

Second, TinyOL-ARLDA demonstrates a decisive advantage in inference latency, completing each frame in 18-22 ms, compared to 28-34 ms for the MLP, 45-58 ms for the CNN-1D, and 52-67 ms for the Autoencoder. This 2-3x speedup is attributable to the closed-form discriminant computation of LDA, which avoids the layered tensor operations required by neural network inference engines. At 100 ms real time constraints typical of industrial monitoring systems, TinyOL-ARLDA leaves substantial headroom for communication, logging, and alarm activation tasks to run concurrently.

Third, the memory footprint of TinyOL-ARLDA (11.6 KB RAM) is dramatically lower than all TFLite alternatives: 3.3x smaller than the MLP (38.2 KB), 6.6x smaller than the CNN-1D (76.5 KB), and 8.1x smaller than the Autoencoder (94.3 KB). This extreme compactness enables deployment on the most constrained microcontrollers such as the STM32F4 or Arduino Nano 33 BLE Sense, which are incompatible with TFLite neural network models due to their limited SRAM.

Finally, power consumption measurements confirm that TinyOL-ARLDA consumes approximately 97 mW on average, representing a reduction of 32 - 52% compared to TFLite models (142 - 203 mW). This energy efficiency is consistent with broader findings on the energy cost of deep learning inference [33] , and stems from both the reduced arithmetic complexity of LDA and the absence of TFLite runtime overhead. For battery-operated IoT sensors and industrial edge nodes where energy budgets are tightly constrained, this reduction translates directly into significantly longer autonomous operation times.

Overall, this comparison demonstrates that TinyOL-ARLDA offers a superior trade-off across all evaluated dimensions: it achieves the best accuracy while simultaneously delivering the lowest latency, the smallest memory footprint, and the lowest power consumption. Unlike TFLite-based solutions which are fundamentally static and require offline retraining cycles to handle machine condition changes, TinyOL-ARLDA adapts autonomously to evolving acoustic environments directly on the device, making it a uniquely suitable solution for long-term predictive maintenance deployments in industrial and IoT settings.

VI. CONCLUSION

This paper presented TinyOL-ARLDA, a novel online and on-device learning framework for sound-based predictive maintenance on resource-constrained embedded systems. By extending Linear Discriminant Analysis [11] with an incremental exponential moving average update mechanism [21] and an adaptive regularization strategy, TinyOL-ARLDA enables continuous model adaptation directly on microcontrollers without any cloud dependency. The proposed confidence-based update rule further ensures a robust balance between plasticity and stability under non-stationary acoustic conditions [22].

Experimental evaluations conducted on the MIMII industrial sound dataset [7] and a custom motor sound dataset demonstrated that TinyOL-ARLDA achieves over 94% classification accuracy during long-duration streaming inference. The feature extraction pipeline, built on spectral and temporal descriptors computed via librosa [25] on desktop and CMSIS-DSP [26] on embedded hardware, provides a compact yet discriminative representation suitable for real-time classification (4 features for the MIMII benchmark; 12 features for the custom motor dataset see Section III-D). A hardware deployment on the Elecrow Mbits (ESP32-WROVER-B) [34] confirmed a total memory footprint of approximately 11.6 KB RAM and an end-to-end latency of 18–22 ms per frame, well within the constraints of real-time industrial monitoring.

Comparative analysis against TensorFlow Lite baselines [27] including quantized INT8 [29] MLP, CNN-1D [28], [32], and Autoencoder models revealed that TinyOL-ARLDA simultaneously achieves superior accuracy, the lowest inference latency (2–3× faster), the smallest memory footprint (3 - 8× smaller), and the lowest power consumption (32 - 52% reduction). Unlike static TFLite models which require periodic offline retraining [35], TinyOL-ARLDA adapts autonomously at runtime, making it uniquely suited to long-term deployments in evolving industrial environments.

Future work will explore multi-class fault classification beyond the binary Normal/Abnormal paradigm, validation on additional machine categories from the MIMII dataset [24] and MIMII-DG benchmark, and integration with federated learning schemes to enable collaborative adaptation across multiple embedded nodes. Extensions to ultra-low-power platforms such as STM32L4 or Nordic nRF52 are also envisioned to further broaden the applicability of TinyOL-ARLDA in battery-powered IoT sensing applications.

Reproducibility: To ensure full reproducibility of the reported results, the complete source code of TinyOL-ARLDA including the Python desktop implementation, the Arduino/ESP32 firmware, and the feature extraction pipeline is publicly available at: **https://github.com/Elmoutabdou/TinyOL-ARLDA**.

Declarations
Funding: This research received no external funding. It was conducted as part of the doctoral research activities of the first author at the CCPS Laboratory, ENSAM Casablanca, Université Hassan II, Morocco.
Corresponding author: Correspondence should be addressed to Abdelwahed Elmoutaoukkil (elmoutaoukkilabd@gmail.com).
Ethics approval and consent to participate: Not applicable.
Consent for publication: Not applicable.
Competing interests: The authors declare no competing interests.

Data Availability
The data used in this study is the publicly available MIMII dataset (Malfunctioning Industrial Machine Investigation and Inspection), which can be freely accessed at: https://zenodo.org/record/3384388. No additional data was generated or analysed during this study beyond what is publicly available in this dataset.

References

[1] R. K. Mobley, *An Introduction to Predictive Maintenance*, 2nd ed. Amsterdam New York: Butterworth-Heinemann, 2002.

[2] S. Ma, K. A. Flanigan, and M. Bergés, "State-of-the-art review and synthesis: A requirement-based roadmap for standardized predictive maintenance automation using digital twin technologies," *Adv. Eng. Inform.*, vol. 62, p. 102800, 2024, doi: 10.1016/j.aei.2024.102800.

[3] N. Davari, B. Veloso, G. D. A. Costa, P. M. Pereira, R. P. Ribeiro, and J. Gama, "A Survey on Data-Driven Predictive Maintenance for the Railway Industry," *Sensors*, vol. 21, no. 17, p. 5739, 2021, doi: 10.3390/s21175739.

[4] C. Banbury, V. J. Reddi, and others, "Benchmarking TinyML Systems," in *Proceedings of Machine Learning and Systems (MLSys)*, 2021.

[5] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, 2020.

[6] M. Lourenço and others, "TinyOL: Incremental Learning on Resource-Constrained Devices," in *Proceedings of the IEEE/ACM International Conference on Internet-of-Things Design and Implementation*, 2022, pp. 227–238. doi: 10.1109/IoTDI54339.2022.00028.

[7] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection," in *DCASE Workshop*, 2019.

[8] K. Yamamoto and others, "Anomaly detection based on sound for machine condition monitoring," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020. doi: 10.1109/ICASSP40776.2020.9053625.

[9] S. Hanifi, B. Alkali, G. Lindsay, M. Waters, and D. McGlinchey, "Advancements in predictive maintenance modelling for industrial electrical motors: Integrating machine learning and sensor technologies," *Meas. Sens.*, p. 101473, 2024, doi: 10.1016/j.measen.2024.101473.

[10] F. Maione, P. Lino, G. Giannino, and G. Maione, "A Deep Learning Algorithm for Predictive Maintenance of Marine Diesel Engines," *Preprints*, 2025, doi: 10.36227/techrxiv.174285181.11777704/v1.

[11] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, 1936.
[12] H. Cevikalp and others, “Online Discriminant Analysis Using Regularized Covariance Estimation,” *Pattern Recognit.*, vol. 134, p. 109094, 2023, doi: 10.1016/j.patcog.2022.109094.
[13] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. Burlington, MA: Academic Press, 2009.
[14] B. Sch"olkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.
[15] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
[16] A. Belatik, M. A. Sabri, H. El Khoukhi, and A. Aarab, “A Comparative Review of Microcontroller Architectures in Embedded Machine Learning: The Raspberry Pi Pico (RP2040) and Arduino Nano 33 IoT (SAMD21),” in *Digital Technologies and Applications*, S. Motahhir and B. Bossoufi, Eds., Cham: Springer Nature Switzerland, 2024, pp. 446–456.
[17] G. Farina et al, “Online Learning on Microcontrollers: A Systematic Review of Methods and Benchmarks.”
[18] P. García-Santaclara, B. Fernández-Castro, R. Díaz Redondo, C. Calvo-Moa, and H. Mariño-Bodelón, *Continual Learning at the Edge: An Agnostic IIoT Architecture*. 2025. doi: 10.48550/arXiv.2512.14311.
[19] Y. Zhang and Z.-H. Zhou, “Incremental Linear Discriminant Analysis: A Fast Algorithm and Comparisons,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2716–2735, 2015, doi: 10.1109/TNNLS.2015.2391201.
[20] K. Shah *et al.*, “XLDA: Linear Discriminant Analysis for Scaling Continual Learning to Extreme Classification at the Edge.” Jul. 2023.
[21] H. Cevikalp and B. Triggs, “Incremental Linear Discriminant Analysis,” *Pattern Recognit.*, 2023, doi: 10.1016/j.patcog.2023.109318.
[22] J. Gama, I. Žliobaité, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A Survey on Concept Drift Adaptation,” *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, 2014, doi: 10.1145/2523813.
[23] A. Elmoutaoukkil, M. Chriss, A. Khatib, and A. Mouchtachi, “Industry 4.0 Efficiency: Predictive Maintenance with TinyML and an Incremental Model,” in *Smart Applications and Data Analysis*, M. Hamlich, F. Dornaika, C. Ordonez, L. Bellatreche, and H. Moutachaouik, Eds., Cham: Springer Nature Switzerland, 2024, pp. 259–267.
[24] K. Dohi, T. Endo, H. Purohit, R. Tanabe, and Y. Kawaguchi, “MIMII DG: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection for Domain Generalization Task,” in *Proceedings of the 7th Detection and Classification of Acoustic Scenes and Events 2022 Workshop (DCASE2022)*, Nov. 2022, pp. 1–5.
[25] B. McFee *et al.*, “librosa: Audio and Music Signal Analysis in Python,” in *Proceedings of the 14th Python in Science Conference (SciPy)*, 2015, pp. 18–24. doi: 10.25080/Majora-7b98e3ed-003.
[26] ARM Ltd., “CMSIS-DSP Software Library.” 2023.
[27] M. Abadi *et al.*, “TensorFlow: A System for Large-Scale Machine Learning,” in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.
[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1097–1105.
[29] B. Jacob *et al.*, “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)*, 2018, pp. 2704–2713. doi: 10.1109/CVPR.2018.00286.
[30] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.
[31] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.” 2017.
[32] T. N. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran, “Deep Convolutional Neural Networks for LVCSR,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 8614–8618. doi: 10.1109/ICASSP.2013.6639347.
[33] E. Strubell, A. Ganesh, and A. McCallum, “Energy and Policy Considerations for Deep Learning in NLP,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 3645–3650. doi: 10.18653/v1/P19-1355.
[34] Espressif Systems, *ESP32 Technical Reference Manual*. Espressif, 2023.
[35] V. J. Reddi *et al.*, “MLPerf Inference Benchmark,” in *Proceedings of the 47th International Symposium on Computer Architecture (ISCA)*, 2020, pp. 446–459. doi: 10.1109/ISCA45697.2020.00045.