LOPdM: A Low-power On-device Predictive Maintenance System Based on Self-powered Sensing and TinyML

Zijie Chen, Yiming Gao, and Junrui Liang, Senior Member, IEEE

Abstract—Predictive maintenance (PdM) has emerged as a prominent strategy that can recognize the current state and predict the future trend of machines. It helps prevent disastrous breakdowns. Such systems were mostly realized based on artificial intelligence (AI) models that run on resource-rich and power-hungry servers. To meet the ultralow-power, low-cost, and on-device-inferencing demands, in this paper, we introduce a self-contained low-power on-device predictive maintenance (LOPdM) system based on the cutting-edge self-powered sensor (SPS) and tiny machine learning (TinyML) techniques. A rich dataset is collected with an SPS in a simulated vibration environment. The collected data is analyzed using six established AI models. Under an ultra-short data length, small data number, and low sampling rate condition, the random forest (RF) and the deep neural network (DNN) stand out with up to 99% precision. The trained model is then deployed on an embedded system for in-situ inferring and condition-based PdM. Power measurement is carefully conducted to compare the power consumption using an inertial measurement unit (IMU) or an SPS, respectively. It shows that the SPS-based system can save up to 66.8% of energy. An all-in-one prototype is assembled and utilized in field tests. It makes a high accuracy in malfunctions identification. As an interdisciplinary study, the development of LOPdM provides valuable guidance for future ubiquitous AI applications.

Index Terms—predictive maintenance, self-powered sensing, Internet of Things, tiny machine learning.

ABBREVIATIONS

ADC analog-to-digital converter
AI artificial intelligence
AloT artificial intelligence of things
BLE Bluetooth low-energy
DL deep learning
DNN deep neural network
DT decision tree
FFT fast Fourier transform
IMU inertial measurement unit
IoT Internet of Things
KNN k-nearest neighbor
LR logistic regression
MCU microcontroller
ML machine learning
PdM predictive maintenance
ReLU rectified linear unit
RF random forest
SoC system on chip
SPS self-powered sensor
SVM support vector machine
TinyML tiny machine learning

I. INTRODUCTION

With the advancement of IoT technologies, a manufacturing digitization and automation process, called Industry 4.0, is thriving [2]. Ubiquitous connectivity and seamless data exchange are realized among people, machines, and products through the Internet [3], [4]. Efficient maintenance is essential to reduce the high costs resulting from unpredictable machine downtime and defective products [5]. PdM is a cutting-edge strategy utilized to forecast severe malfunctions in factory machines. It ensures the smooth operation of production lines [6], [7]. Many devices have a latency period from potential failure to performance failure. The latter is easier to detect. But, as claimed in a previous study, 99% of equipment failures are preceded by some correlated indications [8]. The most fundamental part of a PdM system is to identify the current state of the equipment in real time. Once possible machine anomalies are detected, diagnostic tasks and maintenance actions are triggered to improve the quality and performance of the productive process. ML techniques have recently emerged to reinforce the performance of PdM [9], [10]. Many AloT approaches rely on resource-rich servers due to the limited computing and storing ability of IoT devices [11]. Fig. 1(a) shows the traditional AI-based real-time PdM system, where all data are transmitted to the server. The centralized data processing causes high energy consumption and an unwanted data privacy issue. Recently, researchers have been paying more attention to providing AI functionalities at the edge devices [12]. However, these studies are mainly based on smartphones, field programmable gate array, raspberry pi(s), or personal computers, which are relatively high-cost and high-power compared with MCU [13]. TinyML, a booming branch of state-of-the-art ML techniques, enables the low-cost MCU to run on-device ML models at a milliwatt-level power consumption without real-time support of large servers [14], [15]. The essential technique of TinyML is to compress and
simplify the conventional ML model and deploy it on the end device where data originates [16], [17]. Fig. 1(b) illustrates the TinyML-based PdM system, where the data is inferred locally. An alarm is triggered only if an anomaly is detected. It minimizes the maintenance effort and data transmitting cost. Furthermore, end-side inferring reduces potential data security problems.

IMUs are typical choices among many sensors in vibration signal analyzers. However, IMUs sometimes consume even more energy than MCUs to support their internal conditioning circuit [18]. Its power requirement limits the further development of long-lasting and ubiquitous sensing units [19]. Many energy harvesting solutions were proposed to convert otherwise wasted ambient energy into usable electricity [20]. Before making the whole system self-powered, which is a more demanding task, many studies use the kinetic energy harvesting module as an SPS for providing higher energy efficiency while achieving the desired performance [21], [22]. The parametric relation between the generated voltage from SPS and the specific motions is utilized to extract the motion information. Ref. [19] propose a novel human activity recognition system called HARKE, which collects data from a kinetic energy harvester and achieves basic activity recognition with improved machine learning methods. Compared with the IMU-based designs, SPS-based systems can save power in the sensing part.

**Motivation:** Although some studies in the PdM field have already included the SPS or TinyML techniques recently, efforts toward a more comprehensive and robust co-design methodology are still necessary. For the self-powered sensing part, most studies focus on sensor design and data analysis. The data collection and analysis processes were usually separated. We need a practical real-time monitoring system for immediate PdM. For the TinyML part, several studies leverage a solid AI model that performs well on their dataset. Detailed model comparison and benchmark were missing. In addition, a comprehensive system deployment pipeline, window size and sampling rate analysis, energy consumption measurement, and system-level low-power design are also necessary for a practical PdM application. Therefore, there is still much work to do for integrating self-powered sensing and endpoint intelligence toward self-contained and on-device PdM. In this study, we propose a novel low-power on-device PdM system based on an SPS and TinyML, which is called LOPdM.

With an emphasis on high energy efficiency and low cost, LOPdM utilizes a lightweight piezoelectric cantilever as a sensor and leverages TinyML technology to realize end-side intelligence right at where the data stream originates. By proposing LOPdM, this paper has made three featured contributions:

1) LOPdM is an on-device PdM system based on TinyML. It uses the SPS as a data source. The low-power and low-cost characteristics further promote pervasive PdM.
2) A detailed evaluation of different machine learning methods toward PdM has been provided. RF and DNN stand out regarding accuracy, window size, and required data number. The TinyML deployment pipeline for traditional machine learning and deep learning of the proposed system has been comprehensively introduced.
3) A comprehensive power measurement has been conducted to analyze the power-saving contribution using self-powered sensing. An all-in-one prototype has been fabricated and tested in a field test. It shows good feasibility and excellent performance.

The rest of this article is organized as follows. Section II discusses the related work on TinyML and PdM. Section III provides an overview of the proposed system. Section IV gives the self-powered sensing principle introduction and data analysis. Section V analyzes the performance of different ML models. Section VI presents the development and deployment process of the proposed TinyML-based system. Section VII introduces the experimental evaluation, including energy measurement and field test. Finally, Section VIII draws conclusions.

**II. RELATED WORK**

**A. TinyML**

The number of end devices based on MCUs in the wild environment far exceeds that of traditional cloud and mobile systems. Although smartphones have become more affordable, their relatively high cost, compared with MCUs, remains a barrier to the genuinely ubiquitous AoI. TinyML is a burgeoning field at the intersection of embedded systems, machine learning, and performance engineering [14]. It empowers ultra-low power devices like MCUs to infer original data at the endpoint with a ML algorithm. On-device inferring relieves the omnipresent dependency on remote servers, benefiting low-power operation, economics, low latency, and privacy protection. It is worth noting that TinyML includes all ML methods, not just DL. The efficient TinyML technology brings more intelligence to battery-powered and always-on endpoint devices. It brings revolution to real-time data acquisition and processing [33]. Anomaly detection, as one of the mainstream applications of TinyML, is a technique leveraged for identifying malfunctions in factory machines. It triggers a corresponding maintenance call, thus avoiding the unexpected breakdown of production lines [34].
Vibration information operation time and efficiency [9]. Therefore, equipment maintenance plays a vital role in the manufacturing industry, ranging from 15% to 60% B. PdM are needed to further advance the technology. In addition, more application-specific studies based on TinyML compression and deployment of ML models are desired. In order to compare different TinyML-based embedded systems and widely adopted datasets to train ultra-low-power models are of concern. More theoretical analysis and quantification of the compression and deployment of ML models are desired. In addition, more application-specific studies based on TinyML are needed to further advance the technology.

TensorFlow Lite Micro is the state-of-the-art inference framework from Google [35]. It is designed for straightforward and portable deployment of neural networks. The flexibility based on the interpreter makes it compatible with many processors, such as ARM Cortex-M Series and ESP32 [36]. Several impressive compiler frameworks for TinyML systems have also been proposed, such as Graph Lowering, STM32Cube.AI, TinyEngine, and uTensor.

Although TinyML is still at a very early stage and facing many challenges, plenty of opportunities exist. The ML model is required to be trained offline before deploying on the MCUs. It might be alleviated by the on-device learning technique [37]. The lack of comprehensive benchmarking tools to compare different TinyML-based embedded systems and widely adopted datasets to train ultra-low-power models are of concern. More theoretical analysis and quantification of the compression and deployment of ML models are desired. In addition, more application-specific studies based on TinyML are needed to further advance the technology.

### B. PdM

Maintenance takes a significant portion of operational costs in the manufacturing industry, ranging from 15% to 60% [38]. Therefore, equipment maintenance plays a vital role in identifying and solving machine anomalies. It affects the operation time and efficiency [9].

There are mainly three types of maintenance:

1) Run-to-Failure. It is a reactive maintenance approach carried out only when the equipment stops working. It adds a direct cost to the process [39].

2) Preventive Maintenance. It is performed periodically based on a planned schedule, which leads to unnecessary action costs [40].

3) Predictive Maintenance. PdM enables the repairs to be triggered only when needed or just before, avoiding production shutdowns and minimizing maintenance costs [41].

Given the better safety and lower maintenance cost, PdM outperforms the former two methods. Traditional PdM systems are based on physical models, which quantitatively characterize the behaviors of failure modes derived from first principles [42]. The faults mechanisms should be well understood. It might bring obstacles when the machine model can only be partially obtained [10].

Recently, ML methods, especially DL techniques, have attracted increasing interest from academia and industry for their high accuracy in anomaly detection and prediction. Gongora et al. [43] proposed a failure detection method for a three-phase induction motor based on DNNs using the sampled stator’s current in a motor as input data. In [23], a DNN is trained with IMU data for tool wear monitoring and PdM. In addition, some studies use SPS as a substitute for IMU to lower power consumption. The authors of [28] applied a convolutional neural network to predict a photovoltaic panel’s daily electrical power curve according to the power curves of its neighboring panels. In [29], a piezoelectric micro-electromechanical system based vibration sensor was designed for surface roughness prediction using an ML algorithm.

Furthermore, some studies utilize the transfer learning technique to realize cross-machine fault prognosis under limited data, leading to a more general and practical PdM system. In [25], a deep transfer network with multi-kernel dynamic

### Fig. 2. Architecture of the proposed LOPdM system.

<table>
<thead>
<tr>
<th>Literature</th>
<th>SPS</th>
<th>ML</th>
<th>TinyML</th>
<th>IoT</th>
<th>System-level</th>
<th>Prototyped</th>
<th>Transfer learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hess et al. [25]</td>
<td>-</td>
<td>√</td>
<td>-</td>
<td>-</td>
<td>√</td>
<td>√</td>
<td>-</td>
</tr>
<tr>
<td>Eciton [24]</td>
<td>-</td>
<td>√</td>
<td>-</td>
<td>-</td>
<td>√</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Lv et al. [25]</td>
<td>-</td>
<td>√</td>
<td>-</td>
<td>√</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Han et al. [26]</td>
<td>-</td>
<td>√</td>
<td>-</td>
<td>√</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sagar et al. [22]</td>
<td>√</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>√</td>
<td>-</td>
</tr>
<tr>
<td>Askari et al. [27]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Huuhfianen et al. [28]</td>
<td>√</td>
<td>√</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Trivedi et al. [29]</td>
<td>√</td>
<td>√</td>
<td>-</td>
<td>√</td>
<td>-</td>
<td>√</td>
<td>-</td>
</tr>
<tr>
<td>Zhou et al. [30]</td>
<td>-</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Antonini et al. [31]</td>
<td>-</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kamal et al. [32]</td>
<td>-</td>
<td>√</td>
<td>√</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>This work</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

TABLE I

Comparison among existing technologies and this work.
distribution adaptation is introduced to address the cross-machine fault diagnosis problem. Moreover, the authors in [26] introduced a domain generalization-based hybrid diagnosis network for deploying to rotating machinery under unseen working conditions.

Instead of utilizing the server for data analysis, recent PdM studies based on the TinyML technique are proposed for low-power and privacy-protection demands. For example, in [30], an intelligent rail vehicle running states monitoring system based on TinyML and IMU is introduced. Moreover, an unsupervised anomaly detection algorithm is run locally for autonomously learning the behaviors of the monitored asset.

Table I lists the comparison of this proposed LOPdM design with other existing representative works in literature. To the best of the authors’ knowledge, a PdM system based on TinyML and SPS remains unexplored. Such a system can further satisfy the ultra-low-power and low-cost demands in extreme industrial environments.

III. SYSTEM OVERVIEW

Fig. 2 illustrates the architecture of LOPdM. The equipment typically operates under stable conditions. Any anomalies will cause changes in the machine’s vibration information. To achieve real-time monitoring, a lightweight piezoelectric cantilever, functioning as a self-powered vibration sensor, is used instead of using a commercial IMU. The end-side device is controlled by a low-cost MCU. The end device is then installed on the vibrating equipment for real-time monitoring. The vibration information is reflected by the voltage signal generated from the self-powered piezoelectric sensor.

To save energy, real-world data acquired under different machine states are collected with an ultra-low sampling rate and used for pre-training a TinyML model. According to the Nyquist sampling theorem, the sampling frequency should be at least twice the bandwidth of the feature signal. By consulting with experts, specific abnormal information can be obtained in advance. It allows the system to identify not only the occurrence of the anomaly, but also the specific abnormal situation. Once the compressed model is deployed on an MCU, the end device performs intelligent evaluation. The machine state is sensed and pre-processed before an on-device inference is conducted. The inferring result is then transmitted to a remote receiver with low power consumption within one second. Based on application requirements, the MCU may send an alarm only when a suspicious condition is detected. Unlike conventional IMU-based approaches, LOPdM samples only one transducer that is active all the time, regardless of the presence or absence of a battery supply. In addition, LOPdM minimizes continuous data transmission, which consumes a considerable portion of power in an IoT system. With high accuracy, short latency, and minimal development cost, LOPdM achieves real-time edge machine diagnosis and relieves the tight dependence on the cloud server.

IV. SELF-POWERED SENSING

Industrial systems generate a large amount of data daily. They provide valuable insights into the manufacturing process. Sensing is a critical method for acquiring this data for processing and analysis. Yet, most existing vibration sensing techniques rely on IMUs, which require additional power supplies. SPS is a promising technique for ultra-low-power requirements. Transducers can serve as sensors based on the specific relationship between the environment and the energy generated. However, compared to a well-calibrated integrated IMU, an SPS may have lower accuracy. Therefore, a detailed compensation solution is required.

A. SPS Principle

An IMU is an electromechanical device used to sense vibrations or movements. The most common IMU-based method for sensing vibrations uses a three-axis digital accelerometer to collect motion data, which is then transmitted via an I²C bus. However, both of these processes require a stable power supply. Reducing the sampling rate of the accelerometer or working intermittently can save some power. A self-powered sensing approach that generates one-dimensional vibration data without the need for an extra power supply can eliminate power consumption by reducing the auxiliary circuit.

The environment around us contains numerous sources of kinetic energy that can be converted into electric power using three main transduction mechanisms, i.e., the piezoelectric, electromagnetic, and electrostatic ones.

A piezoelectric transducer can be considered an SPS. Fig. 3 shows a piezoelectric cantilever that generates an alternating voltage according to the vibration condition. One end of the
cantilever beam is fixed at the vibrating structure, which is usually referred to as the base. The other end is free and mounted with a tip mass magnifying the strain. The beam is excited by the base vibration of a machine. A piezoelectric patch is glued at the beam near the fixed end. It subsequently generates an alternating voltage output as the beam oscillates back and forth around the equilibrium position. At the open-circuit condition, the piezoelectric voltage \( v_p(t) \) is proportional to the beam deflection \( x(t) \) with a ratio of \( \alpha \), i.e.,

\[
v_p(t) = \alpha x(t).
\]

In this study, the SPS directly connects to the ADC of the MCU for data acquisition.

B. Data Acquisition

 Maintenance issues vary drastically in different application scenarios. The anomaly information might be unique in each problem [44]. There is currently no available public SPS-based dataset for simulation reference. In this study, we used a vibration platform to simulate a working machine and collected multiple sets of data by referring to a previous study which also conducted a vibration analysis [45]. In real-world industrial applications, using wireless sensor nodes is the most convenient method for distributed data acquisition. Wireless sensor network enables the large-scale deployment of sensors. A low sampling rate is usually adopted for battery-powered sensors to save energy during data sampling. Thus, in this study, we use the data sampled at an ultra-low rate for vibration classification. Owing to the TinyML technology, the poorly sampled data can still achieve high accuracy in vibration pattern recognition.

The experimental setup for data acquisition is shown in Fig. 4. Two low-cost cantilevered piezoelectric sensors are installed on the vibrating base, which is controlled by a computer-based vibration controller (computer #1). As the platform oscillates, an oscilloscope measures the voltage of one generator. The voltage from the other sensor is sampled by an MCU (ESP32, Espressif Inc.) at rates of 10 Hz, 20 Hz, and 40 Hz, respectively, with 12-bit ADC resolution. The digital values are sent to another computer (computer #2) for recording. An 8 Hz vibration is regarded as a normal state, while a 0 Hz one is regarded as an idle state. Three unwanted vibration signals at 10.972 Hz, 12 Hz, and 14.656 Hz are considered as three specific abnormal conditions by an MCU (ESP32, Espressif Inc.) at rates of 10 Hz, 20 Hz, and 40 Hz, respectively, with 12-bit ADC resolution. The sensed value often has some ripple due to the factors such as nonlinearity and electromagnetic interference. Since the ADC module of most MCUs can only read positive voltage, an additional voltage offset circuit is needed if it is necessary to read the negative value. Extra power consumption will be caused by a voltage offset circuit. For an extremely low-cost and low-power purpose, the values with negative polarity are directly neglected in the sampling process in this study. The ADC results in 4 seconds are shown in Fig. 5(c). The piezoelectric open-circuit voltage is sampled at a rate of 20 Hz, a little less than two folds of the vibration frequency, which violates the commonly referred Nyquist sampling criterion. Obvious distortions are observed by discarding the negative values and sampling at such a low sampling rate. Fig. 5(d) shows the time-domain data of 32 sampled points and the corresponding unipolar frequency-domain values after FFT. The most significant frequency component locates at about 9 Hz, which has some variation to the vibration frequency of 10.972 Hz. For energy-saving purposes, less sensing data for the calculation is preferred. Thus, we further decrease the sampled data and check its corresponding frequency-domain picture. Fig. 5(e) shows two sets of the time-domain data of 16 sampled points and the corresponding unipolar frequency-domain values after FFT. From the frequency-domain picture, the model-based FFT method can well describe the most significant frequency component around the same 9 Hz, even the sample points are reduced to half and the phase varies. We further reduce the sampling points to eight in each set. Fig. 5(f) shows the time-domain and frequency-domain pictures of four sets of data. As the data points get much fewer, it is difficult to tell the dominant frequency component from the frequency-domain pictures of all sets of data. The example shows that a model-based expert system might have a blind or incapable area when the data size is reduced below a certain level. However, in such an extremely poor case with a limited size dataset, we can still use the lightweight TinyML technique to fulfill the pattern classification tasks toward anomaly detection.

V. ANALYSIS USING ML METHODS

In this study, we apply six classical supervised ML algorithms, including DT, RF, SVM, LR, KNN, and DNN, on the SPS dataset for testing their performance in classification. RF and DNN stand out among them. The results indicate that optimized ML methods can effectively compensate for the distortion introduced by the low-cost SPS and easy sampling process.

A. ML Principle

ML is a data-driven method focusing on building systems that learn and progress from the dataset. Fig. 6(a) shows the simple architecture of RF, one of the classical ML methods.

C. Blind Area of Model-based Analysis

Fig. 5(a) shows the excitation acceleration signal of the vibration environment at 10.972 Hz. Fig. 5(b) shows the voltage signal generated by a piezoelectric SPS under this excitation. A simple piezoelectric sensor is less accurate than the commercial one. The sensed value often has some ripple due to the factors such as nonlinearity and electromagnetic interference. Since the ADC module of most MCUs can only read positive voltage, an additional voltage offset circuit is needed if it is necessary to read the negative value. Extra power consumption will be caused by a voltage offset circuit. For an extremely low-cost and low-power purpose, the values with negative polarity are directly neglected in the sampling process in this study. The ADC results in 4 seconds are shown in Fig. 5(c). The piezoelectric open-circuit voltage is sampled at a rate of 20 Hz, a little less than two folds of the vibration frequency, which violates the commonly referred Nyquist sampling criterion. Obvious distortions are observed by discarding the negative values and sampling at such a low sampling rate. Fig. 5(d) shows the time-domain data of 32 sampled points and the corresponding unipolar frequency-domain values after FFT. The most significant frequency component locates at about 9 Hz, which has some variation to the vibration frequency of 10.972 Hz. For energy-saving purposes, less sensing data for the calculation is preferred. Thus, we further decrease the sampled data and check its corresponding frequency-domain picture. Fig. 5(e) shows two sets of the time-domain data of 16 sampled points and the corresponding unipolar frequency-domain values after FFT. From the frequency-domain picture, the model-based FFT method can well describe the most significant frequency component around the same 9 Hz, even the sample points are reduced to half and the phase varies. We further reduce the sampling points to eight in each set. Fig. 5(f) shows the time-domain and frequency-domain pictures of four sets of data. As the data points get much fewer, it is difficult to tell the dominant frequency component from the frequency-domain pictures of all sets of data. The example shows that a model-based expert system might have a blind or incapable area when the data size is reduced below a certain level. However, in such an extremely poor case with a limited size dataset, we can still use the lightweight TinyML technique to fulfill the pattern classification tasks toward anomaly detection.
Fig. 5. Sampled data illustration. (a) Excitation signal of the vibration environment under 10.972 Hz. (b) Voltage data generated by an SPS. (c) Digitized values after ADC at a 20 Hz sampling rate. (d) Time-domain data of 32 sampled points and the corresponding unipolar frequency-domain values after FFT. (e) Two sets of time-domain data of 16 sampled points and the corresponding unipolar frequency-domain values. (f) Four sets of time-domain data of 8 sampled points and the corresponding unipolar frequency-domain values. It is difficult to explicitly tell the similarities by comparing these sets of 8-point data.

DL, a subset of ML, is one of the most popular topics in academia and industry nowadays. It can automatically discover and learn effective features from raw data [12]. Fig. 6(b) shows a simple DNN architecture, which includes three parts: input layer, hidden layer, and output layer. Every layer contains many neurons, and the trainable weights connect the neurons. The input layer comprises the input data features. The weighted sums from all hidden layers propagate to the output layer, which produces the final outputs of the network [46]. The neurons of hidden and output layers are generally combined with a summation and a non-linear function. The output of a neuron can be derived as follows

$$y_j = f \left( \sum_i \omega_{ij} x_{ij} + b_j \right),$$

where $\omega_{ij}$, $x_{ij}$, $y_j$, $b_j$, and $f(\cdot)$ are the weights, input activations, output activations, bias term, and the non-linear function of the $j$ neuron, respectively. The non-linear function is also called the activation function.

The essence of training is to continuously optimize the weights using the optimizer until the loss function is minimized to a satisfactory level or the designated training epoch is reached. The process of running programs with these trained

Fig. 6. RF and DNN architectures. (a) RF. (b) DNN.

First, the number of decision trees, $n$, needs to be decided. For every tree, the given data is sampled randomly with replacement from the training dataset based on bootstrap sampling. During training, the features for every tree are selected stochastically to ensure that each decision tree (evaluator) is unique and independent. RF takes into account the predicted classes of multiple evaluators and determines the result based on voting strategies, such as majority voting.
weights is referred to as inference. Hyperparameters, which are set before the learning process, directly affect how well a model is trained. Therefore, analyzing the impact of different hyperparameters and selecting the best-performing ones is a common way to optimize a DNN model.

B. Comparison of Different AI Methods

Given the challenges of the distorted SPS signal, we aim to utilize classical ML models to analyze the data. In addition, we intend to select the most suitable model for the target system among DT, RF, SVM, LR, KNN, and DNN from different perspectives. For comparability, all the following analysis is based on a triple classification problem for identifying 0 Hz, 8 Hz, and 10.972 Hz from the SPS vibration signal, which is sampled at a rate of 20 Hz. Traditional ML models are based on the default settings of the Scikit-learn library. The results are based on 5-fold cross-validation. LR’s regularization parameter and penalty are 1.0 and L2; DT’s max depth, minimal samples split, and criterion are none, 2, and Gini; SVM’s regularization parameter and kernel function is 1.0 and Rbf; KNN’s neighbor number and distance metric are 5 and Minkowski; RF’s tree number, max depth, minimal samples split, and criterion are 13, none, 2, and gini. DNN is configured with 2 hidden layers with a neuron distribution of (16, 8). The activation function is ReLU. The output function is Softmax. The optimizer is Adam. The loss function is the cross-entropy function. The batch size is 16. The learning rate is 0.1. All raw SPS data is normalized before training.

Fig. 7(a) shows the accuracy of different models for varying data lengths from 1 to 20. Except for the LR algorithm, all models reach a high accuracy when the data length is only 8, corresponding to a window size of 0.4 seconds at a sampling rate of 20 Hz. Therefore, AI methods can satisfy high performance in identifying SPS signals with an ultra-small window size, which benefits in small latency, computational complexity, and memory requirement. So the remainder of the analysis in this paper is presented using a window size of 0.4 seconds. In addition, Fig. 7(b) illustrates the exact accuracy of models when the data length is 8. RF and DNN exhibit slightly better performance than other models.

Fig. 8(a) displays the accuracy of different models for the required data set number of each class, ranging from 10 to 250. The result demonstrates RF and DNN can reach an accuracy higher than 98% with only 150 sets of data for each class, while DT, SVM, KNN can reach an accuracy higher than 91%. Smaller amounts of data required for well-performing models result in lower costs for building and developing the system. Fig. 8(b) illustrates the specific accuracy of RF and DNN for required data samples ranging from 80 to 150. Thus, the remainder of the analysis in this paper is presented using only 150 sets of data, which are divided into training and testing sets in the ratio of 8 : 2.

For simple ADC sampling, a small increase in sampling rate does not result in a large increase in energy consumption [19]. Considering the trade-off between accuracy and power consumption, a 20 Hz sampling rate is preferred and utilized in the remainder of this paper.

[Figures and tables are not included in this text representation.]
C. Hyperparameter Comparison

After selecting RF and DNN through analysis of different data perspectives, it is time to optimize their hyperparameters. RF comprises many decision trees. The max depth parameter limits the number of splits performed in a decision tree. It determines the number of layers that each decision tree can reach. The larger the max depth, the better RF performs, but a lower computational efficiency and a greater possibility of over-fitting. Fig. 10(a) shows the accuracy of RF for max depth ranging from 6 to 20. The result indicates that a max depth of 10 is enough. So the remainder of the RF results with pruning skill is presented using this value. In addition, RF does not always perform significantly better as the number of decision trees increases. Doubling the tree number may sometimes be pointless [47]. Fig. 10(b) shows the accuracy of RF for tree numbers ranging from 1 to 20. We can see that RF performs best when the tree number is 9, reaching an accuracy of 99.56%. The RF performance is not reduced with pruning. Moreover, the RF with the gini criterion performs better than the one with the entropy criterion.

For DNN, there exist many hyperparameters to consider, such as activation function, batch size, optimizer, the number of hidden layers, etc. Fig. 11(a) shows the accuracy of DNN for different activation functions, including Sigmoid, ReLU, ReLU6, Leaky ReLU, Swish, and Hard Swish. DNN can converge rapidly with all activation functions except for the Sigmoid function. In this study, DNN performs best with the ReLU6 function, which is defined as follows

\[ y = \min [\max (0, x), 6], \tag{3} \]

where \( x \) is the input and \( y \) is the output. The batch size defines the number of data propagated for training at each iteration. A larger batch size leads to faster network convergence, better memory utilization, and fewer training shocks, but longer iteration times and a higher probability of getting stuck in a locally optimal solution. Fig. 11(b) shows the accuracy of DNN for different batch sizes. Based on the aforementioned trade-off, a batch size of 32 is preferred. Fig. 11(c) shows the accuracy of DNN for different optimizers, including Stochastic gradient descent, Momentum, Adagrad, RMSprop, and Adam. The result of this study proves that the DNN with Adam optimizer performs best. In addition, from Fig. 11, we can see that an epoch of 150 is sufficient. It means that the training process is very fast. The cost of developing LOPdM is low. Using the above optimal hyperparameters, we further discuss the effect of the number of hidden layers. DNN can possess better learning and inferring ability as the number of layers increases. Table II lists the performance of DNN models with different hidden layers and neuron distributions. The number of parameters directly affects the model’s size, latency, and practicality. We can see that the number of parameters can reach a higher value, although the number of hidden layers is smaller. So, the selection of neurons is also essential. The results show that DNN can reach 100% accuracy with only 331 trainable parameters and a hidden layer structure of (16, 8, 4).

D. Remarks

After analyzing different ML models, we found that both RF and DNN can achieve ultra-high accuracy in triple classifications, given optimal hyperparameters and small window size, dataset, and sampling rate requirements. To further explore the performance of RF and DNN in quadruple and quintuple classifications, we conducted additional experiments. Fig. 12 illustrates the performance of optimized RF and DNN in triple, quadruple, and quintuple classifications. These results demonstrate that the models can achieve high accuracy for multiple classification problems, which confirms the performance of LOPdM. The key hyperparameters of RF and DNN for different cases are also provided in Fig. 12.
VI. TINYML DEVELOPMENT AND DEPLOYMENT

TinyML equips edge IoT nodes with a certain level of intelligence, substantially reducing the computing burden on cloud servers and the power consumption for data communication. When AI models are well designed, the next step is model compression and deployment. Owing to some advanced ML frameworks and mature developing tools, compressing and deploying a model on an MCU is now handy. With energy-aware programming and system-level designs, the energy consumption for intermittent computing on low-cost IoT devices can be further reduced.

A. Pipeline of TinyML

Fig. 13 illustrates the TinyML pipeline of the proposed system, including end device sampling, model generation, model deployment, and application development. An SPS captures vibration information from the equipment. To reduce power consumption, a lower MCU sampling rate is preferred in data acquisition. The real-world data is then processed using the techniques such as data normalization, missing value imputation, or feature selection.

Both traditional ML algorithms and deep learning models can be deployed on MCUs. For traditional ML, the model is built using the open-source Scikit-learn library and trained with processed data. The model is then converted using advanced inference frameworks like micromlgen. For DL, a DNN is built using TensorFlow (Google Inc.) and trained with processed data. It is noteworthy that a smaller model size translates to lower inference time [48]. Once trained, the model is compressed using TensorFlow Lite, an advanced inference framework designed for running ML algorithms on resource-constrained end devices [35]. Techniques like parameter quantization or pruning are used to achieve compression. The development cost is lower when less data and smaller training epochs are required.

The lightweight models, once converted into an embedded system-compatible format, can be easily deployed to TinyML-supported MCUs for on-device monitoring of real-time SPS signals. Owing to the complete and open embedded ecosystem, different applications can be easily customized.

B. On-device Condition-based PdM

Before locally inferring real-time signals, the MCU often needs to pre-process the data to extract features in both time and frequency domains, in order to improve the detection performance. The vibration time-series information usually undergoes FFT, which consumes a considerable amount of electrical energy. In this study, we prove that normalizing the raw data is sufficient before sending it to the TinyML model. The simple pre-processing further reduces power consumption for each operation cycle.

The conceptual current profile of the proposed on-device low-power monitoring system is shown in Fig. 14. The system can be implemented based on most 32-bit Cortex-M or Xtensa LX6 MCUs. The entire process can generally be divided into four states: sleeping, sampling, data pre-processing, and inferring. During normal operation, real-world data is sensed within a specific time interval. The original data is then processed and inferred on the device within an extremely short time, typically one second. Studies have demonstrated that intermittent operation significantly reduces power consumption [20]. Therefore, the MCU is in sleep mode most of the time to save energy. It is woken up by a timer and works intermittently. For most MCUs, the operation current is in the μA and mA levels for sleeping and active modes, respectively.

MCU has different sleeping modes to meet different demands. Generally, deep sleep and light sleep are the two most popular modes. The current in deep-sleep mode (denoted as $I_{\text{deep}}$) is much lower than that in light-sleep mode (denoted as $I_{\text{light}}$). However, most MCUs need to be initialized after being woken up from deep sleep. Light sleep has no such initialization when being woken up. Thus, there is a trade-off between sleep duration (denoted as $t_{\text{sleep}}$) and energy consumption in initialization (denoted as $E_{\text{init}}$) when the supply voltage $V$ is stable. The guideline for selecting the sleep mode is derived as follows

$$
\text{Sleep mode} = \begin{cases} 
\text{Deep sleep}, & t_{\text{sleep}} \geq \frac{E_{\text{init}}}{V(I_{\text{light}} - I_{\text{deep}})}, \\
\text{Light sleep}, & t_{\text{sleep}} < \frac{E_{\text{init}}}{V(I_{\text{light}} - I_{\text{deep}})}
\end{cases}
$$

For instance, since the $E_{\text{init}}$ of the MCU (ESP32, Espressif Inc.) is relatively high, light-sleep mode is preferred when a short monitoring interval is required.

After the on-device inference, the MCU can transmit a warning message through Bluetooth low-energy (BLE) or low-power WiFi channel once an anomaly is detected, such that to realize condition-based PdM. Therefore, transmission is not needed most of the time. End-side intelligent devices achieve good performance with low latency and high energy efficiency by using the TinyML and SPS techniques. The power consumption can be further reduced by using the ultra-low-power SoC or energy-aware programming technology [21].
Fig. 13. The development and deployment process of the proposed TinyML-based system toward lightweight PdM demand.

Fig. 14. Conceptual current profile illustrating the intermittent operation of the on-device low-power monitoring.

Fig. 15. Experimental setup for energy measurement.

Fig. 16. Measured current profile of one round operation.

VII. EXPERIMENTAL EVALUATION

To comprehensively evaluate and quantify the power-saving capability of LOPdM, we conduct detailed energy consumption measurements in a laboratory environment. Subsequently, the all-in-one prototype is assembled and used for validating the performance of LOPdM in the field test.

A. Power Measurement

Fig. 15 shows the experimental setup for measuring the energy consumption in either the SPS-based or IMU-based system. Two MCUs (ESP32, based on a 240 MHz Tensilica Xtensa LX6 processor with 512 KB RAM, 2 MB flash, Espressif Inc.) of the same model number are chosen for two groups of experiments. A lightweight off-the-shelf piezoelectric-film sensor based on polyvinylidene fluoride is utilized as an SPS. A commonly used IMU (MPU6050, TDK Ltd.) is used in the comparative study. The IMU is sampled at 100 Hz and communicates through the I2C bus with the MCU. For a fair comparison, the IMU is initialized to only sample acceleration in a single direction. It means only 1-axial acceleration data is generated to save power. In addition, no third-party library is used for programming since it might introduce extra power consumption. The MCU is connected with a current sensor in series and powered by a 3.7 V lithium battery. Then, the current is measured by a current analyzer (CX3322A, Keysight Inc.) connected across the current sensor. Table III presents a comparison of the deployed RF and DNN, where RF has a relatively bigger model size but smaller energy consumption and inferring latency. Both the RF and DNN models meet the memory, computing, and power limitations of a common MCU. It demonstrates the rapid and economic characteristics of TinyML and ubiquitous AI.

Fig. 16 illustrates the measured current diagram of the MCU, which runs intermittently. The MCU has been programmed to execute delay functions that separate different operational events in order to measure specific power consumption clearly. The energy consumption statistics of a round of SPS sampling, IMU sampling, data normalization, model inferring, and FFT are listed in Table IV. From the table, it is evident that the duration of FFT is longer and the energy cost is higher than those of TinyML inferring.
This suggests that TinyML outperforms FFT in terms of performance and power consumption in this proposed system. In real-world applications, the MCU goes into sleep mode before the next sampling operation. The current from the MCU (ESP32, Espressif Inc.) in light sleep is approximately 0.8 mA on average. Thus, the energy for sleeping during eight-time sampling at a 20 Hz sampling rate is $800 \mu A \times 3.7 V \times (8 - 1) \times 0.05 \text{ sec} = 1036 \mu J$. The amounts of energy consumption in data sampling are $333.5 \mu J \times 8 + 1036 \mu J = 3704 \mu J$ and $20.02 \mu J \times 8 + 1036 \mu J = 1196.16 \mu J$ for IMU-based and SPS-based systems, respectively. The total energy consumption of the system based on light sleep can be calculated as the sum of the energy for data sampling, pre-processing, and inferring. According to the energy consumption of every signal atomic operation listed in Table IV, for a round of operation with the IMU-based system, the total consumed energy is $3704 \mu J + 16.64 \mu J + 31.2 \mu J = 3751.84 \mu J$. For the SPS-based case, it is $1196.16 \mu J + 16.64 \mu J + 31.2 \mu J = 1244.00 \mu J$. The comparison of these two solutions is summarized in Table V. It demonstrates that the proposed SPS + TinyML system can save 66.8% of energy for carrying out the same AI-based detection task. The results indicate that SPS is superior to most commonly used accelerometers in terms of energy saving. The SPS-based system can be even more energy-efficient when using other lower-power MCUs.

B. Prototype and Field Test

A compact prototype is assembled for better conducting the field test, as shown in Fig. 17(a). It consists of a housing shell, a low-power SoC (ESP32, Espressif Inc.), an SPS, and a lithium battery. The housing shell is made out of Polylactic acid or polylactide material using a 3D printer. The size of the prototype is $5 \times 7 \times 2.1 \text{ cm}^3$. Its weight is 37 g. The real-time signals generated by the SPS are sampled and inferred by the SoC. When the prototype is mounted on a machine, the information on the machine status can be transmitted to nearby IoT nodes via BLE wireless packets.

Fig. 17(b) shows the experimental setup for the field test. All MCU settings follow the above analysis. A customized app is developed on a mobile phone to receive and display the BLE messages sent by the endpoint device. A brushless motor mounted with the prototype is controlled by the digital power supply. The different supplied voltages cause the corresponding vibrations through the motor. We set five different real-world vibration states by changing the supplied voltage to the motor. It is considered as an idle state at 0 V, a normal state at 8 V, and three specific similar abnormal states at 9 V, 10 V, and 11 V, respectively. Based on the aforementioned data analysis, a small dataset of 100 seconds is sufficient for each class. For robust performance, we chose a sampling rate of 40 Hz, which means the window size is only 0.2 seconds. The original sampled data of five cases are transmitted wirelessly to a personal computer for data pre-processing and model training. A DNN model is generated with the same setting provided in personal computer for data pre-processing and model training.
performance of LOPdM.

The MCU does not send any signals when the machine is in a normal state. Wireless packets carrying specific information are sent by the MCU when a specific malfunction occurs. The TinyML model has a near-perfect recognition ability for different machine states. When unknown anomalies occur, the MCU will have a relatively low confidence rate. It means that the MCU can also accurately distinguish the presence of unknown anomalies by simply taking a threshold setting. To increase the stability and reliability of the system, we set the MCU to report only if the same abnormal information is analyzed twice in a row. Within a 20 m radius around the receiver, we found no data loss through BLE transmission. In addition to using mobile phones as receivers, low-cost terminals such as MCU or Raspberry Pi can also be used. During the experiment, receivers do not have downtime problems due to the stable power supply. Together with the domain knowledge about the system’s failure mechanism, specific anomaly signals deduced by experts can make the system more comprehensive and robust.

The proposed LOPdM is a general method that can be applied to detecting irregularities and preventing severe events in other application vibration scenarios, such as aerogenerators, industrial robots, launch vehicles, and bridges. When extensively deployed, the performance can be further improved by the cutting-edge AI techniques like transfer learning, end-edge-cloud computing, on-device learning, etc. The on-device, energy-efficient, and low-cost characteristics of LOPdM provide a promising prospect for pervasive sensing and AIoT applications.

VIII. CONCLUSION

This paper proposed a low-power on-device predictive maintenance system, called LOPdM, based on SPS and TinyML techniques. TinyML equips the end device with intelligence for perceiving its situation in real time with high accuracy and low power consumption. To further reduce power consumption, a piezoelectric SPS was utilized to replace the commonly used IMU in conventional designs. A rich dataset based on the SPS was collected with a vibration platform and analyzed using six well-known ML models, including LR, SVM, DT, KNN, RF, and DNN. RF and DNN were the most capable out of the six for identifying raw SPS signals with up to 99% accuracy when only using a data length of 8, a data number of 150, and a sampling rate of 20 Hz (about twice of the signal frequency). The hyperparameters of these two models were also finely tuned. In addition, the MCU operates intermittently to extend battery life. After a comprehensive design from sensing, analyzing, intermittent operation mode, etc., the LOPdM system can reliably fulfill an energy-efficient condition-based PdM. Several groups of experiment were conducted to validate and quantify the feasibility, performance, and energy consumption of LOPdM. The results showed that the system achieves excellent performance in identifying anomalies while saving 66.8% of energy compared with its conventional IMU counterpart. This design procedure provides a valuable insight for future studies of pervasive sensing and AIoT.

REFERENCES


