IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 January 2026, accepted 11 February 2026, date of publication 20 February 2026, date of current version 2 March 2026.

*Digital Object Identifier 10.1109/ACCESS.2026.3666778*

RESEARCH ARTICLE

# A Low-Resource Hardware Design for Bearing Fault Detection Using Support Vector Machines

**JAN TIMMER, ARDAVAN ELAHI, (Member, IEEE), PHILIPP LEHNINGER, (Member, IEEE), AND AXEL JANTSCH, (Fellow, IEEE)**

Faculty of Electrical Engineering and Information Technology, Vienna University of Technology (TU Wien), 1040 Vienna, Austria

Corresponding author: Ardavan Elahi (ardavan.elahi@tuwien.ac.at)

This work was supported by the i-EDGE project, which has received funding from the European Union (grant number 101092018), the Swiss State Secretariat for Education, Research and Innovation (SERI) and U.K. Research and Innovation (UKRI) under the U.K. government's Horizon Europe funding guarantee (grant numbers 10061130 and 10063023).

**ABSTRACT** Bearing fault detection is critical for ensuring the reliability of industrial machinery, as mechanical failures can lead to significant downtime and maintenance costs. This paper presents a hardware-efficient accelerator for bearing fault detection based on Support Vector Machines (SVMs). Unlike neural network–based accelerators, which are often over-parameterized and demand substantial hardware resources, the proposed design focuses on compactness and hardware efficiency. By combining carefully selected time-domain features with quantization and normalization techniques, we developed a hardware-efficient architecture suitable for both FPGA and ASIC implementation. Evaluated on the Case Western Reserve University (CWRU) bearing dataset, the accelerator achieves 96.93% accuracy in the single-variate and 98.42% in the multi-variates implementation. The FPGA implementation and the ASIC synthesized in TSMC 65-nm demonstrate significantly lower hardware requirements compared to state-of-the-art designs, making the proposed approach highly suitable for deployment in resource-constrained environments.

**INDEX TERMS** ASIC, FPGA, bearing fault detection, hardware accelerator, machine learning, support vector machines (SVM).

## I. INTRODUCTION

Rotating machinery is a critical component of industrial systems, powering applications from manufacturing plants to energy production. The reliability of such machinery strongly depends on the integrity of its bearings, which are essential for ensuring smooth operation. Bearing faults are among the most common causes of machine breakdowns, accounting for approximately 40% of failures in large machines and up to 90% in smaller machines [1]. These failures can lead to costly downtime, reduced efficiency, and unplanned maintenance. Consequently, early and accurate detection of bearing faults has become a critical research area in the broader context of predictive maintenance and Industry 4.0.

The associate editor coordinating the review of this manuscript and approving it for publication was Guillermo Valencia-Palomo.

Traditional condition-monitoring systems rely on analyzing vibration or current signals using statistical or signal-processing methods. More recently, machine learning (ML) and deep learning (DL) approaches have demonstrated impressive diagnostic capabilities [2], [3]. Convolutional Neural Networks (CNNs) and related architectures can automatically learn features from raw data and have achieved high diagnostic accuracy. However, these models are often hardware-inefficient, requiring large numbers of parameters and extensive computational resources. This makes them challenging to deploy in resource-constrained embedded environments, where low power and compact design are essential [4], [5].

To address these challenges, there is growing interest in hardware accelerators, which bring computation closer to the sensor and reduce latency, power, and cost [6]. Hardware

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

accelerators can be realized on FPGAs for rapid prototyping and flexibility, or as ASICs for highly optimized deployments. While most state-of-the-art hardware accelerators for bearing fault detection are based on neural networks, these approaches often trade efficiency for marginal performance gains. In contrast, classical machine learning algorithms (CMLAs) such as Support Vector Machines (SVMs) offer simpler and more interpretable models that are inherently more hardware-friendly. Although they require explicit feature extraction, they have the potential to deliver competitive accuracy with far lower hardware cost.

Motivated by these insights, this paper proposes a hardware-efficient accelerator for bearing fault detection based on SVMs and lightweight feature extraction. By carefully selecting a minimal subset of time-domain features and applying quantization and normalization, we design an implementation that achieves high accuracy while drastically reducing resource usage. We show that bearing fault detection accelerators can achieve competitive accuracy at significantly lower hardware cost by leveraging classical ML methods instead of over-parameterized neural networks.
The main contributions of this work are as follows:

1) We investigate hardware-efficient feature selection for bearing fault detection, identifying a minimal subset that balances accuracy and resource usage.
2) We design a minimal and compact solution based on SVM with integrated normalization and evaluate it using the CWRU data set, achieving 96. 93% precision in the single-variate and 98.42% in the multi-variate case.
3) We propose a hardware-efficient architecture, implemented and validated on a Xilinx Zynq-7000 FPGA, and synthesized as an ASIC design using the TSMC 65-nm standard-cell library. The results demonstrate significant hardware savings compared to state-of-the-art neural network–based accelerators. The proposed SVM-based accelerator achieves major resource savings, requiring 6× fewer LUTs and 8× fewer flip-flops on FPGA compared to the most hardware efficient prior design, and achieving 2.5×–2.7× lower area than state-of-the-art ASIC implementations, while maintaining competitive accuracy.

The remainder of this paper is organized as follows. Section II reviews related work on accelerator designs for bearing fault detection, highlighting both FPGA and ASIC-based approaches. Section III presents the proposed methodology, including the model design and hardware architecture of our accelerator. Section IV reports the results, evaluating both the accuracy and hardware resources of the proposed implementation. Finally, Section V concludes the paper.

## II. RELATED WORKS

The purpose of this section is to provide an overview of the problem space and summarize related work in designing hardware accelerators for bearing fault detection. We begin by introducing the datasets commonly used in this domain. We then review the state-of-the-art hardware accelerators, focusing on implementations on FPGAs and ASICs, which aim to achieve high accuracy while maintaining strict constraints on area, power, and throughput. By examining these works, we highlight the strengths and limitations of existing approaches and identify opportunities for further optimization, which motivate the design choices of our proposed accelerator.

**TABLE 1. Label and fault types in CWRU dataset (12 kHz) [7].**

| Label | Diameter (inch) | Type of fault | Description |
|---|---|---|---|
| 0 | 0.007 | Ball Fault | B007_0 to B007_3 |
| 1 | 0.014 | Ball Fault | B014_0 to B014_3 |
| 2 | 0.021 | Ball Fault | B021_0 to B021_3 |
| 3 | 0.007 | Inner Race Fault | IR007_0 to IR007_3 |
| 4 | 0.014 | Inner Race Fault | IR014_0 to IR014_3 |
| 5 | 0.021 | Inner Race Fault | IR021_0 to IR021_3 |
| 6 | 0 | Normal Healthy Data | Normal_0 to Normal_3 |
| 7 | 0.007 | Outer Race Fault | OR007@6_0 to OR007@6_3, OR007@3_0 to OR007@3_3, OR007@12_0 to OR007@12_3 |
| 8 | 0.014 | Outer Race Fault | OR014@6_0 to OR014@6_3 |
| 9 | 0.021 | Outer Race Fault | OR021@6_0 to OR021@6_3, OR021@3_0 to OR021@3_3, OR021@12_0 to OR021@12_3 |

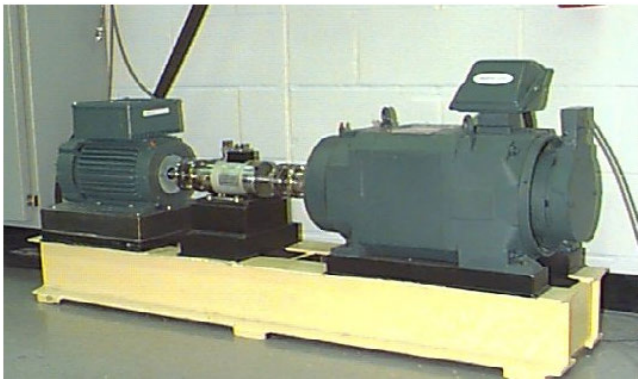

**FIGURE 1. Test set up for CWRU bearing dataset [7].**

### A. DATASETS

As mentioned earlier, mechanical failures of bearings account for a major part of failures in machinery. From these failures the need for bearing fault detection arises.

For this purpose, a dataset obtained from the Case Western Reserve University (CWRU) [7] Bearing Data Center was utilized. The experiments that produced this dataset involved a 2 hp Reliance Electric motor, with acceleration data collected from locations near the motor bearings. The bearings were intentionally seeded with faults using electro-discharge

machining (EDM), providing a range of fault conditions for the analysis. Fig. 1 shows the test setup.

The CWRU bearing dataset consist of data measured at 12kHz and at 48kHz. Since the 12kHz dataset has more data points, we will be using this set for this work. The dataset is divided into 10 classes as described in Table 1.

The CWRU bearing dataset provides a robust benchmark for evaluating the bearing fault detection algorithm. This dataset offers extensive bearing fault data, making it ideal for testing fault detection and classification. It ensures comprehensive testing across diverse scenarios, demonstrating the algorithm's reliability and robustness.

Another popular dataset is provided by the Mechanical Engineering Construction and Drive Technology Research Center at Paderborn University, commonly referred to as the PU dataset [8]. It contains vibration measurements of bearings with faults located on the inner race, outer race, and in some cases a combination of both. The dataset distinguishes between artificially introduced damage, created through techniques such as electrical discharge machining and drilling, and realistic faults, which were generated through accelerated life testing under high radial loads and improper lubrication [9]. In this work, we select the CWRU dataset because it is widely regarded as one of the most comprehensive bearing datasets. Compared to alternatives such as the PU dataset, it contains more fault classes, diverse load conditions, and multiple fault diameters. Although our experiments rely solely on CWRU, this is consistent with most prior work, as CWRU offers ten fault classes and well-structured operating scenarios. These characteristics make it the most widely adopted benchmark for evaluating bearing-fault detection methods.

### B. ACCELERATORS FOR BEARING FAULT DETECTION

Efforts to bring bearing fault detection to the edge devices focus on achieving high accuracy, throughput, and low power consumption in resource-constrained environments. Recent research primarily explores deep neural networks on FPGAs and ASICs.

Liao et al. [4] proposed BearingPGA-Net, a lightweight CNN for bearing fault diagnosis trained via decoupled knowledge distillation and deployed on FPGA hardware. In [5] a hardware-efficient convolutional autoencoder (AE) is proposed for bearing fault detection, where the weights are binarized while the activations are only partially binarized to balance accuracy and resource usage. The design was mapped to FPGA and synthesized in TSMC CMOS nodes, achieving 138 $\mu$W/MHz power and 0.49 mm$^2$ area, making it suitable for on-die integration with MEMS sensors.

Authors in [10] designed a 2-D hierarchical CNN (HCNN) accelerator in 40-nm CMOS for bearing fault diagnosis using the CWRU dataset. With only 29k parameters and real-time operation at 100 MHz, the post-layout design achieved 95.31% accuracy and 65.6 mW power. Chung et al. [9] presented a 1-D CNN hardware accelerator for real-time bearing fault diagnosis on FPGA. By combining down-sampling and quaternary quantization, their design reduced memory usage by 89% while achieving 96.37% accuracy and real-time performance at 100 MHz with a response time of 0.28 s.

In [11], a lightweight and multiplier-free neural network accelerator is proposed for FPGA-based bearing fault detection, comprising only 8.69k parameters. By applying incremental network quantization and fixed-point operations, the design reduced memory usage by 63.5%, enabling real-time inference at 48 k samples/s with 342 mW power and 95.12% accuracy, making it well suited for predictive maintenance in industrial settings. Also, Fu et al. [12] proposed ADEPOS, an approximate computing approach for bearing fault detection that adaptively varies network size and precision to save energy. Implemented in a fabricated UMC 65-nm CMOS chip, the system achieved an 8.95× energy reduction over its lifetime on the NASA bearing dataset while maintaining full detection accuracy.

Zhang et al. [13] implemented a 1-D CNN accelerator with depthwise separable convolution (DSC) on an FPGA for bearing fault detection using the PU dataset. By quantization of the DSC, the design reduced parameters to 22 KB and achieved 96.12% accuracy with 527 mW power at 50 MHz.

De Vita et al. [14] proposed a hardware-friendly analog SVM classifier with on-chip learning capability, implemented in TSMC 90-nm CMOS. Post-layout simulations showed 72 $\mu$W power at 0.6 V and accuracy within 1.4% of the software baseline, demonstrating a trade-off of area efficiency for full autonomy.

While neural networks dominate the current design space, classical machine learning algorithms (CMLAs) remain underexplored, presenting opportunities for further optimization in edge-based fault detection systems.

### C. CLASSICAL MACHINE LEARNING ALGORITHMS FOR BEARING FAULT DETECTION

In this part, we review the CMLA algorithms that were used for bearing fault detection.

In [15], Support Vector Machines (SVMs), K-Nearest Neighbor (KNN), and Naive Bayes are compared alongside two neural networks. SVMs achieve the highest accuracy among CMLAs but are prone to overfitting. Similarly, [16] evaluates SVM, KNN, Decision Tree (DT), and LSTM using the CWRU dataset, where SVM outperforms the others.

K-Means clustering in [17] achieves 99.921% accuracy on the CWRU dataset using 3 time-domain and 5 frequency-domain features.

Overall, SVMs consistently achieve high accuracy in bearing fault detection; however, they rely on explicit feature extraction, in contrast to neural networks, which can automatically learn features from raw data.

### D. CONCLUSION

The landscape of hardware accelerators for bearing fault detection is highly diverse. Our review reveals that deep neural networks (DNNs) proposed in the literature vary

substantially in size and complexity, often leading to large differences in hardware cost. In many cases, the performance gains from increasing network depth or parameter count are marginal compared to the significant increase in resource usage. This indicates that careful architectural design, rather than brute-force scaling of model size, is crucial to achieving true hardware efficiency.

Given this observation, it becomes essential to also consider classical machine learning algorithms (CMLAs), such as support vector machines, which inherently demand fewer resources. However, little attention has been devoted to the development of CMLA-based hardware accelerators for bearing fault detection, despite their potential advantages. While these methods require explicit feature extraction, their reduced complexity and smaller hardware footprint make them promising candidates for efficient accelerator design. In this work, we therefore investigate the use of support vector machines as a CMLA approach for accelerating bearing fault detection.

## III. METHODOLOGY

In this section, we seek to answer the question, 'How can an extremely hardware-efficient accelerator for bearing fault detection be designed with an SVM?'. We have divided the methodology into several subsections to answer that question. We first describe the model development process using the CWRU dataset [7], including the SVM formulation, feature extraction, and training. We then present the FPGA and ASIC architectures and explain how the complete system is implemented in hardware.

### A. SUPPORT VECTOR MACHINE

Support Vector Machines were first introduced in [18]. The core idea behind a support vector machine is to map a line or, in higher dimensions, a hyperplane between the different classes which maximizes the margin between two classes at a time as seen in Fig. 2.

Given a trained Support Vector Machine model, the inference process for a new input vector $\mathbf{x}$ involves calculating the sign of the decision function, which is defined as follows:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \tag{1}$$

where:

- $N$ is the number of support vectors.
- $y_i \in \{-1, 1\}$ represents the class labels of the support vectors.
- $\alpha_i$ are the learned weights in the dual problem representation.
- $K(\mathbf{x}_i, \mathbf{x})$ is the kernel function used to compute the inner product in the transformed feature space.
- $b$ is the bias term.

The sign of $f(\mathbf{x})$ determines the class of $\mathbf{x}$:

$$\text{Class of } \mathbf{x} = \begin{cases} 1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & \text{if } f(\mathbf{x}) < 0 \end{cases} \tag{2}$$

The kernel function $K$ allows the SVM to perform classification in a higher-dimensional space without explicitly mapping the input features to that space, a technique known as the kernel trick.

When using a linear kernel, the inference process in a Support Vector Machine involves calculating the dot product directly between the input vectors. The decision function for a new input vector $\mathbf{x}$ can be represented as:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b) \tag{3}$$

where:

- $\mathbf{w}$ is the weight vector, which is a linear combination of the support vectors, scaled by their corresponding labels and dual coefficients.
- $b$ is the bias term.

The decision function directly computes the weighted sum of the input features, adjusted by the bias, and the sign of this sum determines the class:

$$\text{Class of } \mathbf{x} = \begin{cases} 1 & \text{if } f(\mathbf{x}) \geq 0 \\ -1 & \text{if } f(\mathbf{x}) < 0 \end{cases} \tag{4}$$

The linear kernel function simplifies to $K(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i^T \mathbf{x}$, which is the standard dot product of vectors $\mathbf{x}_i$ and $\mathbf{x}$. This approach provides a straightforward geometric interpretation of the decision boundary as a hyperplane in the feature space.

The linear kernel is the least computationally intensive kernel in an SVM, primarily because it eliminates the need for complex transformations of the input data. Unlike non-linear kernels, such as the polynomial or radial basis function (RBF) kernels, which map input data into higher-dimensional spaces to find a more separable boundary, the linear kernel directly operates in the original feature space. This direct approach significantly reduces the computational overhead, as it bypasses the need for calculating and storing additional dimensions or performing non-linear transformations.

The simplicity of the linear kernel also results in fewer mathematical operations during both training and inference. Specifically, it minimizes the number of multiplications and additions required, as the decision function involves a straightforward dot product between the weight vector and the input features, followed by a bias adjustment. This reduction in computational complexity translates into faster processing times, making the linear kernel particularly suitable for real-time applications and scenarios where computational resources are limited.

SVM's were originally designed for binary classification, requiring voting structures like One-vs-One (OvO) and One-vs-All (OvA) to handle multi-class problems. OvO offers key advantages over OvA, particularly in terms of classification performance. By simplifying each binary classifier to distinguish between just two classes, OvO generally

produces more accurate decision boundaries, especially in non-linearly separable or imbalanced datasets. While OvO requires more classifiers, $K(K-1)/2$ where $K$ equals the number of classes compared to $K$ classifiers for OvA, each is smaller and faster to train, reducing overall complexity and improving reliability. Consequently, OvO is preferred when high accuracy is crucial, justifying its choice in this paper.

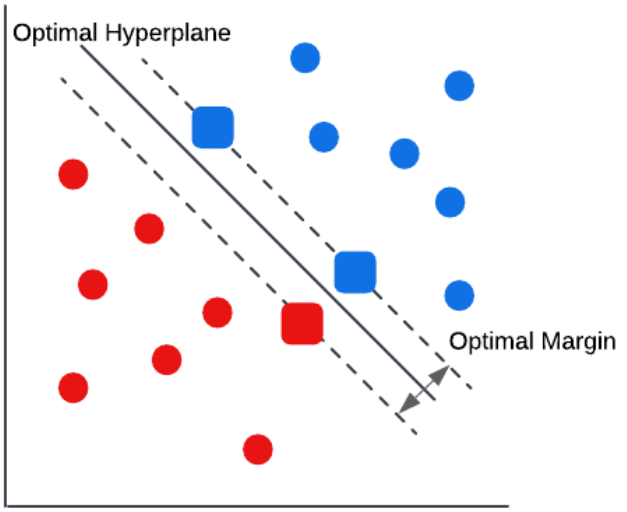


**FIGURE 2.** "An example of a separable problem in a 2 dimensional space. The support vectors, marked with grey squares, define the margin of largest separation between the two classes" [18].

### B. FEATURE SPACE

One of the conclusions drawn from the literature review was that CMLA's in the context of bearing fault detection require feature extraction for adequate performance. For that reason, we explore hardware efficient features in this section.

In the feature space exploration, we look at the insights from [19], which details a variety of statistical features in context to bearing fault diagnosis. This reference explores 11 different feature extraction methods from time series data, namely a sound recording, and implements them on a decision tree to improve its performance. The conclusion the paper draws is that only 4 of the extracted features improve the performance of the algorithm, namely, skewness, mean, median, and kurtosis in the order of importance.

Authors in [20] present an array of fault detection methods implemented on the Case Western University bearing dataset via a Support Vector Machine. The paper presents a feature selection method for the various features that it introduces.

Authors in [21] explore a series of time-domain feature extraction methods while using the CWRU bearing dataset as a reference point. Furthermore, the paper analyses how stride and window size have an impact on the accuracy of several classification algorithms.

A series of hardware efficient feature extraction methods were chosen out the papers above based on following criteria.

- No multiplication in the feature extraction.
- Only divisions with magnitudes of 2 are allowed.
- Only features in the time-domain are allowed.

**TABLE 2.** Studied hardware-efficient features.

| Feature | Formula |
|---|---|
| Mean | $\frac{1}{N}\sum_{i=1}^{N} x_i$ |
| Mean Absolute Value (MAV) | $\frac{1}{N}\sum_{i=1}^{N} \lvert x_i \rvert$ |
| Waveform Length (WL) | $\sum_{i=1}^{N} \lvert x_i - x_{i-1} \rvert$ |
| Peak value | $\max \lvert x_i \rvert$ |
| Slope Sign Change (SSC) | $\left(x_i > x_{i-1} \wedge x_i > x_{i+1}\right) \vee \left(x_i < x_{i-1} \wedge x_i < x_{i+1}\right)$ for $\lvert x_i - x_{i-1} \rvert \geq \varepsilon$ |
| Zero Crossing (ZC) | $\left(x_i > 0 \wedge x_{i+1} < 0\right) \vee \left(x_i < 0 \wedge x_{i+1} > 0\right)$ for $\lvert x_i - x_{i+1} \rvert \geq \varepsilon$ |

A final subset picked from Table 2 has been chosen by first grading the features using the Laplacian score [21], which scores each feature on how well it separates the classes. The scores were: *1) MAV, 2) WL, 3) Peak, 4) ZC, 5) SSC, 6) Mean.* Furthermore, visual analysis of the the features, as shown in Fig. 3, was done to ensure there was as little correlation between the features as possible. The features MAV and WL share high correlation, for this reason WL was not used.

The analysis showed that Mean Absolute Value (MAV), Peak value, and Zero Crossing (ZC) were the highest-ranked features. Although both MAV and Waveform Length (WL) performed well, their strong correlation led to excluding WL, resulting in the final selection of MAV, Peak value, and ZC as the feature set. To avoid counting invalid sign changes caused by quantization noise, a small threshold of Epsilon $= 1$ is used, corresponding to the smallest non-zero quantization step in our 16-bit representation. This ensures that only meaningful zero-crossing events are detected.

### C. QUANTIZATION

Quantization is applied to the dataset and trained variables to ensure minimal hardware cost. The dataset is quantized down to signed 8-bit integer while the weights and biases are quantized to 18-bit fixed point. Floating-point representation was avoided to ensure minimal hardware cost in the arithmetic units. We attempt to mimic the behavior of an 8-bit analog-to-digital converter (ADC) by quantizing the dataset down to signed 8-bit. For this quantization we take inspiration from [22]. We use uniform symmetric quantization to remove the need for any dequantization.

Fixed point 18 was chosen to ensure that no overflow would occur after the feature extraction. We use a window size of 1024 which equals to $2^{10}$ to calculate the features. Our worst case scenario would be that the value 128 would be multiplied 1024 times, signed fixed point 18 would be able to handle this case.

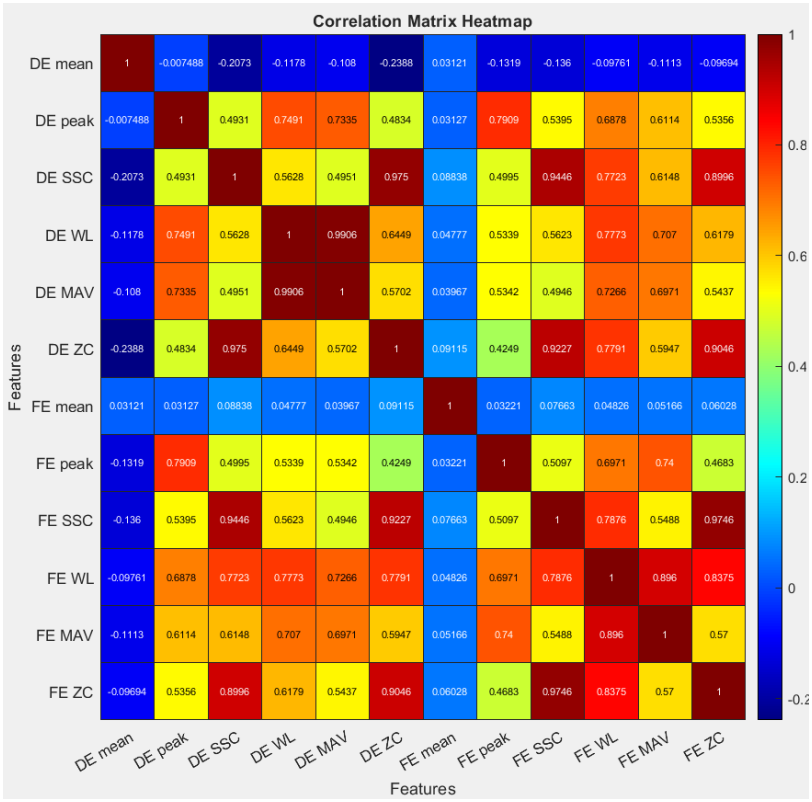


**FIGURE 3.** CWRU feature correlation matrix.

### D. NORMALIZATION

Extracted features vary greatly in means and standard deviations which in turn makes it difficult to have high precision when applying fixed point arithmetic on them. Normalization of the features has been applied to mitigate this problem. The normalization of the features can be described using the formula 5 where $x \in \vec{x}$ in which $\vec{x}$ describes the feature, $\mu$ describes the mean of that feature and $\sigma$ describes the standard deviation of the feature. The normalization function ensures that the mean of each feature equals zero and the standard deviation equals one.

Equations 6 and 7 describe the inference of the support vector machine. The normalization of the features happens before the inference so when combining the two functions $g(f(x))$ it results in formula 8. $\Sigma^{-1}$ is an inverted diagonal matrix of the standard deviations of the features $X$ where each column represents one feature. $\vec{\mu}$ is a vector of all the means of the features in $X$. The formula can be written back to it's original form shown by equation 6 if $W^T \rightarrow W^T\Sigma^{-1}$ and $\vec{b} \rightarrow -W^T\vec{\mu}\Sigma^{-1} + \vec{b}$.

$$f(x) = (x - \mu)/\sigma \tag{5}$$

$$g(x) = W^T X + \vec{b} \tag{6}$$

where

$$W^T\vec{x} + \vec{b} = \begin{cases} \geq 0 & \text{if correct} \\ < 0 & \text{if incorrect} \end{cases} \tag{7}$$

$$W^T X\Sigma^{-1} - W^T\vec{\mu}\Sigma^{-1} + \vec{b} \tag{8}$$

### E. TRAINING PROCESS

Fig. 4 shows the training process of the SVM. We ensure that there is no data leakage by splitting the train set and the test set prior to the feature extraction. The SVM is tested by utilizing the non-normalized test set with the normalized, quantized parameters. No overlap of classes occurs in the feature extraction, ensuring clear separation of classes.

### F. ARCHITECTURE

We have implemented single-variate version with DE (Drive-end) data and multi-variate with DE and FE (Fan-end) data. The architecture for both single-variate and multi-variate is the same. The reason is that 1) the frequency of our synthesized design is the order of magnitude higher the input frequency(12 KHz), so we can extract the features for both FE and DE with the same hardware, 2) we have just one shared multiplier that performs dot product for both FE and DE time series. Fig. 5 shows the architecture of our accelerator. Only DE data is used in the single-variate implementation.

We use a window size of 1024 values with a stride of 16 to calculate each of the extracted features. The window size was chosen to be as small as possible to ensure minimal hardware usage while not sacrificing accuracy. In contrast, the stride was chosen to be as large as possible to ensure that no timing issues would arise. Insights were taken from [21]. Moreover, the fixed window size of 1024 enables a simpler hardware implementation. As shown in Table 2, the MAV feature requires a division by the window size N. This division

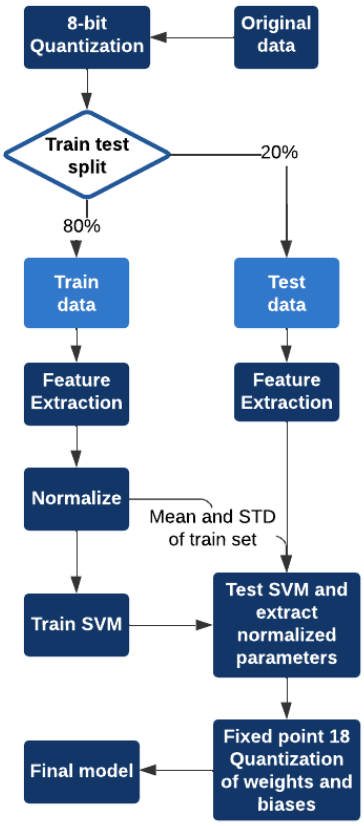


**FIGURE 4.** Training process of the SVM.

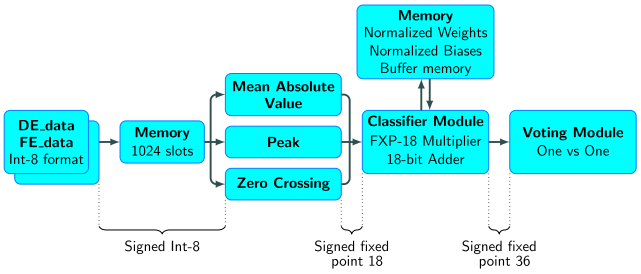


**FIGURE 5.** Single- and multi-variate architecture.

can be implemented as a 10-bit right shift in hardware. As illustrated in Fig. 7, this shift-based implementation eliminates the need for a dedicated division unit and keeps the Feature Extraction Unit lightweight and hardware-efficient. In hardware, coefficients (weights and biases) are stored as ROM: on the FPGA they are placed in BRAMs, while in the ASIC they are synthesized as fixed registers. This avoids any run-time normalization and keeps the classifier datapath lightweight. The 1-vs-1 classifier requires six weight arrays (three per class), two bias arrays, and the class-pair mapping for 45 class pairs, resulting in a total coefficient storage of 8460 bits (≈1.03 kB). This memory is implemented as ROM on FPGA and as synthesized constants in the ASIC design.

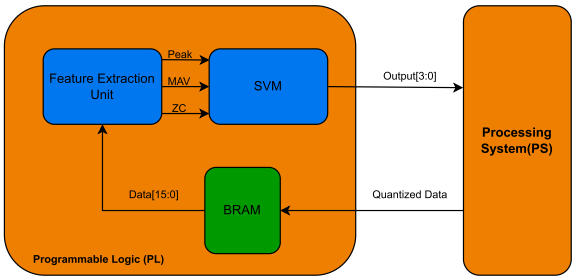


**FIGURE 6.** Overall FPGA accelerator architecture.

### G. HARDWARE IMPLEMENTATION

This section describes the hardware architecture of both the FPGA and ASIC implementations.

#### 1) FPGA

The proposed design was implemented in VHDL on a Zedboard equipped with the Xilinx Zynq-7020 SoC. Fig. 6 illustrates the overall FPGA architecture. The Processing Subsystem (PS) is responsible for pre-processing and loading the input data into the on-chip BRAM. The raw sensor data are quantized and packed into 16-bit words, such that each BRAM word stores two sensor samples. All quantization and data packing operations are performed in the PS.

A controller implemented in the Programmable Logic (PL) orchestrates the complete operation and manages data transfers between the modules, although it is not shown in Fig. 6. As described previously, the window size is set to 1024 samples. Once the BRAM contains a full window of data, processing is initiated. The BRAM data are fetched and passed to the Feature Extraction Unit, which computes three features: 1) *Peak*, the maximum value within the window, 2) *MAV*, the mean absolute value, and 3) *ZC*, the zero-crossing count. The epsilon in ZC is 1. The computed feature vector is provided as input to the SVM module, which implements a one-vs-one multi-class SVM classifier. The SVM classifies the fault label as defined in Table 1 and sends the result back to the PS. All communication between the PS and PL is handled via AXI interfaces.

Fig. 7 illustrates the datapath of the feature extraction unit, which computes the peak value (PV), mean absolute value (MAV), and zero-crossing (ZC) features. The internal data width is 18 bits; therefore, the 8-bit input samples are sign-extended to 18 bits prior to feature extraction. The control signals between the datapath and the controller are omitted in the figure for clarity. The MAV is accumulated over a fixed window and subsequently normalized by a logical right shift of 10 bits (srl 10), corresponding to division by the window size $N = 1024$, which provides an efficient hardware implementation of the averaging operation. The Sign Check block detects sign inversion between two consecutive samples and produces a Boolean output indicating a potential zero crossing. To remove noise-induced crossings, the absolute difference between consecutive samples is evaluated by the `abs_comp` block, which asserts a Boolean output only when

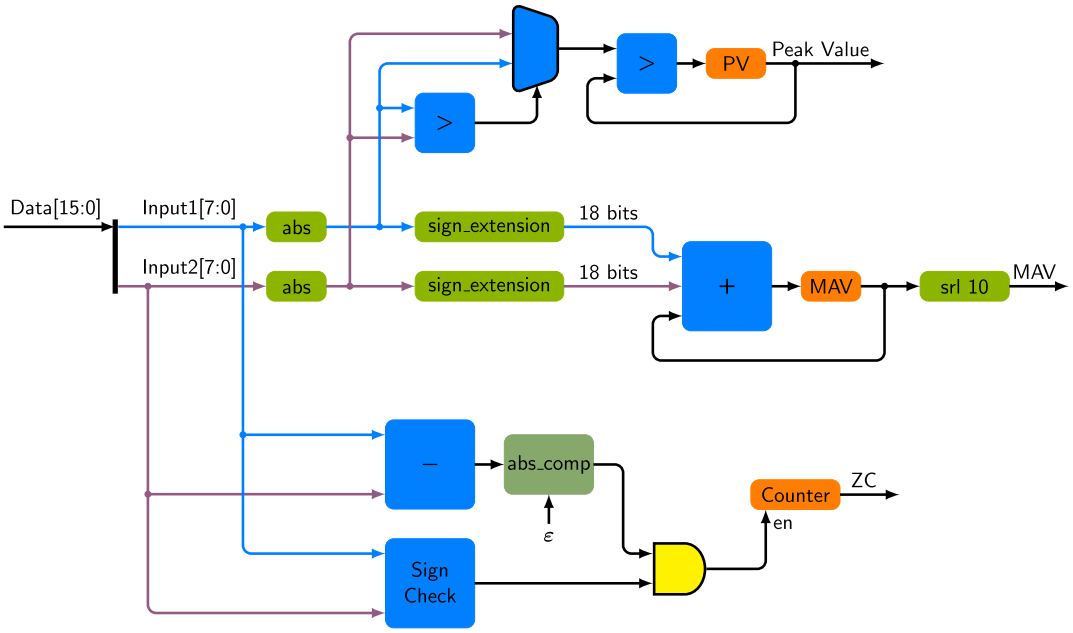


**FIGURE 7.** Datapath of the feature extraction unit.

the magnitude exceeds the predefined threshold $\varepsilon$, which is 1 in our implementation. The ZC counter is enabled only when both a sign change and sufficient amplitude variation are detected, ensuring robust zero-crossing detection.

#### 2) ASIC

The proposed architecture was also synthesized as an ASIC accelerator using the TSMC 65 nm standard-cell library. The ASIC implementation closely follows the FPGA architecture described previously, with the compute core consisting of the Feature Extraction Unit, the one-vs-one SVM classifier, control logic, and a small on-chip buffer for storing data samples.

System-level interfaces, such as the analog-to-digital converter (ADC), pad ring, and host communication, are outside the scope of this work, as our primary research focus is on hardware-efficient classification. We assume that sensor signals are quantized by an external 8-bit ADC at a sampling rate of 12 kHz, and the resulting samples are stored in the on-chip buffer before processing. For the 1024-sample window required in our design, a 2-KB buffer was implemented to store the samples and support multi-variate operation. Since the required memory size is small, it was implemented using synthesized registers instead of dedicated SRAM macros. All results reported are pre-layout synthesis estimates obtained using Cadence Genus.

## IV. RESULTS

In this section, we present the results of our work in terms of both model accuracy and hardware implementation. Tables 3 and 4 show the confusion matrices for the multi-variate and single-variate implementations, respectively.

### A. FPGA

Table 5 reports the synthesis results of our FPGA implementation and compares the resource utilization, classification accuracy, frequency, and power consumption with related works. Our focus in this work is on a hardware-efficient implementation of bearing fault detection, which in our case is realized through the Feature Extraction Unit and the SVM classifier. For this reason, the reported numbers correspond only to the Programmable Logic (PL) part of the Zynq device, where all feature extraction and classification computations are executed. The Processing System (PS) is limited to communication and control tasks, and therefore the AXI logic contribution is excluded from the synthesis results.

As shown in Table 5, the proposed SVM-based accelerator achieves competitive accuracy while drastically reducing hardware cost compared to existing FPGA implementations. Our design reaches 96.93% accuracy in the single-variate case and 98.42% in the multi-variate case, which is only slightly lower than the accuracies reported by deep neural network–based approaches (e.g., 99.61% for the convolutional autoencoder in [5]).

These works rely on large deep-learning models built on CNNs, which act as powerful feature extractors but require substantially higher hardware resources. Therefore, achieving slightly higher accuracy is expected given their significantly larger model capacity and hardware requirement.

However, this small accuracy drop comes with large reduction in hardware usage: our design requires only 409 LUTs, 299 FFs, 1 DSP, and 2 BRAMs, compared to 151,637 LUTs, 180,099 FFs, and 185 DSPs in [4], or 9,306 LUTs and 5,703 FFs in [11]. Even against the more lightweight autoencoder design in [5], our implementation still consumes less than

**TABLE 3.** Confusion matrix (single-variate).

| True Labels | Predicted Labels | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | B007 | B014 | B021 | IR007 | IR014 | IR021 | Normal | OR007 | OR014 | OR021 |
| B007 | 6001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B014 | 0 | 4462 | 964 | 0 | 605 | 0 | 0 | 0 | 0 | 0 |
| B021 | 0 | 820 | 5196 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IR007 | 0 | 0 | 0 | 6059 | 0 | 0 | 0 | 0 | 0 | 0 |
| IR014 | 0 | 628 | 0 | 0 | 5412 | 0 | 0 | 0 | 0 | 0 |
| IR021 | 0 | 0 | 0 | 0 | 0 | 6019 | 0 | 23 | 0 | 0 |
| Normal | 0 | 0 | 0 | 0 | 0 | 0 | 21153 | 0 | 0 | 0 |
| OR007 | 0 | 0 | 0 | 0 | 0 | 30 | 0 | 18253 | 0 | 0 |
| OR014 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6035 | 0 |
| OR021 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 18230 |

**TABLE 4.** Confusion matrix (multi-variate).

| True Labels | Predicted Labels | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | B007 | B014 | B021 | IR007 | IR014 | IR021 | Normal | OR007 | OR014 | OR021 |
| B007 | 6001 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B014 | 0 | 5151 | 609 | 0 | 271 | 0 | 0 | 0 | 0 | 0 |
| B021 | 0 | 436 | 5580 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IR007 | 0 | 0 | 0 | 6059 | 0 | 0 | 0 | 0 | 0 | 0 |
| IR014 | 0 | 233 | 0 | 0 | 5807 | 0 | 0 | 0 | 0 | 0 |
| IR021 | 0 | 0 | 0 | 0 | 0 | 6010 | 0 | 32 | 0 | 0 |
| Normal | 0 | 0 | 0 | 0 | 0 | 0 | 21153 | 0 | 0 | 0 |
| OR007 | 0 | 0 | 0 | 0 | 0 | 12 | 0 | 18271 | 0 | 0 |
| OR014 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6035 | 0 |
| OR021 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 18230 |

one-fifth of the LUT resources. This comparison highlights that by leveraging classical machine learning methods such as SVMs, it is possible to build a high-accuracy bearing fault detection system while achieving massive savings in FPGA resources, making the design highly suitable for deployment in resource-constrained embedded devices.

In addition to the accuracy and hardware resources, the FPGA implementation was also evaluated in terms of power, latency, and energy efficiency. Power consumption was obtained from post-implementation analysis at a clock frequency of 50 MHz in Vivado. The reported power corresponds exclusively to the programmable-logic inference hardware, including feature extraction, SVM classification, and on-chip RAM/ROM blocks. The total on-chip power is 111 mW. Compared with other FPGA-based works, the proposed design exhibits competitive or lower power consumption: BearingPGA-Net reports 670 mW at 100 MHz on a Kintex-7 device [4], the multiplier-free CNN reports 342 mW at 5 MHz on a Virtex-7 device [11], and the autoencoder-based approach of Vitolo et al. [5] reports 122 mW at 45 MHz on an Artix-7 device. These results indicate that the proposed lightweight SVM-based architecture achieves comparable inference functionality with significantly reduced hardware complexity and power consumption.

At 50 MHz, the end-to-end latency, measured from the time all 1024 samples are available until the classification result is produced, is 28.88 $\mu$s (1444 clock cycles), corresponding to an energy consumption of 3.21 $\mu$J per classification, or 3.13 nJ per sample for a 1024-sample window. Direct comparison of these energy-efficiency metrics with prior FPGA-based works is not possible, as existing studies do not report cycle-accurate latency or window-level timing required to derive energy per classification.

### B. ASIC

Table 6 presents the results of our ASIC implementation in comparison with other ASIC-based designs for bearing fault detection. Our design was synthesized using the TSMC 65-nm technology node.

To ensure a fair comparison across ASIC implementations in different technology nodes, we normalize all reported results to 65 nm using the DeepScaleTool [23]. DeepScaleTool derives technology-scaling factors from empirically extracted silicon trend data reported across multiple process generations and applies regression-based models to generate consistent normalization factors between nodes. The tool achieves prediction errors of only 1–5% across key metrics, making it appropriate for cross-node normalization in digital hardware implementation. Using this tool, we scaled the reported results of other works to the TSMC 65-nm node, which are listed in Table 6.

In terms of area, our design is the most compact among the compared works. As shown in Table 6, the proposed SVM accelerator occupies only 0.1808 mm$^2$ in TSMC 65-nm technology. This footprint is significantly smaller than the reported areas of other state-of-the-art designs, while achieving accuracy that is only marginally lower than theirs. This demonstrates that a very large reduction in hardware cost can be obtained without sacrificing competitive accuracy. Our proposed design has a power consumption of 17.3 mW at 100 MHz, which is considerably lower than the 89.863 mW required by the CNN-based

**TABLE 5. FPGA results and comparison.**

| Metric | [4] | [5] | [11] | Our Work |
|---|---|---|---|---|
| Algorithm | CNN | Autoencoder | CNN | SVM |
| Dataset | CWRU + PU | CWRU | CWRU | CWRU |
| Accuracy | 97.12% | 99.61% | 95.12% | 96.93% (Single-variate)98.42% (Multi-variate) |
| FPGA Platform | Xilinx Kintex-7 | Xilinx Artix-7 | Xilinx Virtex-7 | Xilinx Zynq-7000 |
| Frequency (MHz) | 100 | 45 | 5 | 50 |
| Power Consumption (W) | 0.67 | 0.122 | 0.342 | 0.111 |
| LUTs | 151,637 | 2,449 | 9,306 | 409 |
| FFs | 180,099 | 2,319 | 5,703 | 299 |
| LUTRAMs | 936 | 211 | – | – |
| BRAMs | 259 | – | 8.5 | 2 |
| DSPs | 185 | 0 | 0 | 1 |

accelerator in [10]. The smaller power reported in [5] and [14] arise from fundamentally different design choices and evaluation conditions, which complicate direct comparison. In [5], power is reported for an autoencoder-based anomaly detector with an operating frequency of 2.46 MHz, and the overall system follows a HW/SW co-design approach in which the classifier is invoked conditionally rather than as a continuously operating end-to-end pipeline. In [14], classification is performed using an analog SVM optimized with subthreshold region techniques and low supply voltage, and their reported power consumption excludes analog memories and pre-processing circuits. Due to these architectural and evaluation differences, a direct one-to-one comparison of power consumption across designs is not fully feasible; nevertheless, we have attempted to perform the comparison as fairly as possible by normalizing reported results and explicitly clarifying the scope and operating conditions of each work.

Energy efficiency is evaluated using energy per classification, defined as the energy required to produce one valid classification result, and, where applicable, energy per sample, obtained by normalizing the classification energy by the input window size. For this work, these metrics are derived from total pre-layout power consumption at 100 MHz and cycle-accurate latency measured from the point at which all input samples are available until the classification output is produced. To enable a fair comparison, metrics such as latency and window size must be reported; however, not all prior works provide this information. In [14], neither the sampling window nor the clock frequency is reported, and only the energy per classification is provided. In [5], energy efficiency is evaluated as a function of output data rate (ODR), reflecting system-level operation rather than cycle-accurate inference latency, which prevents direct extraction of per-classification energy. As can be seen in Table 6, both energy-efficiency metrics of our work are significantly lower than those reported in [10], due to the use of a much simpler and more hardware-efficient algorithm and implementation. However, the superior efficiency metrics reported in [14] can be primarily attributed to the use of analog computation combined with subthreshold circuit techniques.

**TABLE 6. ASIC results and comparison.**

| | [10] | [5] | [14] | This Work |
|---|---|---|---|---|
| Technology | TSMC 40nm | TSMC 65nm LP HVT | TSMC 90nm | TSMC 65nm |
| Algorithm | CNN | Auto Encoder | SVM | SVM |
| Dataset | CWRU | CWRU | VSBD [24] | CWRU |
| Area * ($mm^2$) | 2.6098 | 0.49 | 0.4236 | 0.1808 |
| Power Consumption * | 89.863 mW | 341 $\mu$W ** | 62.609 $\mu$W | 17.3 mW |
| Frequency (MHz) | 100 | 200 | - | 100 |
| Energy per Classification * ( $\mu$J/Class) | 296.5 | - | $0.444 \times 10^{-3}$ | 0.25 |
| Energy per Sample * (nJ/Sample) | 72.4 | - | - | 0.244 |
| Number of Classes | 10 | 10 | 2 | 10 |
| Accuracy | 95.31% | 99.54% | 83.2% | 96.93% Single-variate ——— 98.42% Multi-variates |

* Reported area, power consumption, and energy are scaled to 65nm.
** Reported for the clock frequency of 2.46 MHz.

## V. CONCLUSION

This paper presented a hardware-efficient accelerator for bearing fault detection based on Support Vector Machines (SVMs). By integrating lightweight feature extraction, quantization, and normalization, the proposed approach demonstrates that classical machine learning methods can be effectively adapted for compact hardware implementations. The results confirm that, for bearing fault detection, SVM-based accelerators can deliver performance on par with neural network–based designs while consuming substantially fewer resources, making them highly suitable for FPGA and ASIC deployment in industrial monitoring systems.

More broadly, this work highlights that many state-of-the-art accelerators rely on over-parameterized neural networks that provide only limited benefits relative to their hardware

cost. In contrast, our findings show that carefully designed classical machine learning algorithms can achieve efficient and practical solutions for predictive maintenance. This opens up a promising direction for future research in developing resource-conscious hardware accelerators that seek hardware efficiency without compromising diagnostic effectiveness.

## ACKNOWLEDGMENT

The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding.

## REFERENCES

[1] D. T. Hoang and H. J. Kang, "A motor current signal-based bearing fault diagnosis using deep learning and information fusion," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 6, pp. 3325–3333, Jun. 2020.
[2] S. Zhang, S. Zhang, B. Wang, and T. G. Habetler, "Deep learning algorithms for bearing fault diagnostics—A comprehensive review," *IEEE Access*, vol. 8, pp. 29857–29881, 2020.
[3] S. Mushtaq, M. M. M. Islam, and M. Sohaib, "Deep learning aided data-driven fault diagnosis of rotatory machine: A comprehensive review," *Energies*, vol. 14, no. 16, p. 5150, Aug. 2021.
[4] J.-X. Liao, S.-L. Wei, C.-L. Xie, T. Zeng, J. Sun, S. Zhang, X. Zhang, and F.-L. Fan, "BearingPGA-net: A lightweight and deployable bearing fault diagnosis network via decoupled knowledge distillation and FPGA acceleration," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–14, 2024.
[5] P. Vitolo, A. De Vita, L. D. Benedetto, D. Pau, and G. D. Licciardo, "Low-power detection and classification for in-sensor predictive maintenance based on vibration monitoring," *IEEE Sensors J.*, vol. 22, no. 7, pp. 6942–6951, Apr. 2022.
[6] D. Ghimire, D. Kil, and S.-H. Kim, "A survey on efficient convolutional neural networks and hardware acceleration," *Electronics*, vol. 11, no. 6, p. 945, Mar. 2022.
[7] *Case Western Reserve University Bearing Data Center Website*. Accessed: Aug. 2025. [Online]. Available: https://engineering.case.edu/bearingdatacenter
[8] *Paderborn University Bearing Data Center Website*. Accessed: Aug. 2025. [Online]. Available: https://mb.uni-paderborn.de/kat/forschung/kat-datacenter/bearing-datacenter
[9] C.-C. Chung, Y.-P. Liang, and H.-J. Jiang, "CNN hardware accelerator for real-time bearing fault diagnosis," *Sensors*, vol. 23, no. 13, p. 5897, Jun. 2023.
[10] Y.-P. Liang, Y.-S. Hsu, and C.-C. Chung, "A low-power hierarchical CNN hardware accelerator for bearing fault diagnosis," *IEEE Trans. Instrum. Meas.*, vol. 73, pp. 1–11, 2024.
[11] Y.-P. Liang, M.-Y. Hung, and C.-C. Chung, "A multiplier-free convolution neural network hardware accelerator for real-time bearing condition detection of CNC machinery," *Sensors*, vol. 23, no. 23, p. 9437, Nov. 2023.
[12] S. K. Bose, B. Kar, M. Roy, P. K. Gopalakrishnan, L. Zhang, A. Patil, and A. Basu, "ADEPOS: A novel approximate computing framework for anomaly detection systems and its implementation in 65-nm CMOS," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 3, pp. 913–926, Mar. 2020.
[13] Y.-P. Liang, H. Chen, and C.-C. Chung, "A one-dimensional depthwise separable convolutional neural network for bearing fault diagnosis implemented on FPGA," *Sensors*, vol. 24, no. 23, p. 7831, Dec. 2024.
[14] V. Alimisis, G. Gennis, M. Gourdouparis, C. Dimas, and P. P. Sotiriadis, "A low-power analog integrated implementation of the support vector machine algorithm with on-chip learning tested on a bearing fault application," *Sensors*, vol. 23, no. 8, p. 3978, Apr. 2023.
[15] R. Liu, B. Yang, E. Zio, and X. Chen, "Artificial intelligence for fault diagnosis of rotating machinery: A review," *Mech. Syst. Signal Process.*, vol. 108, pp. 33–47, Aug. 2018.
[16] X. Zhang, B. Zhao, and Y. Lin, "Machine learning based bearing fault diagnosis using the case western reserve university data: A review," *IEEE Access*, vol. 9, pp. 155598–155608, 2021.
[17] N. R. Dreher, I. O. de Almeida, G. C. Storti, G. B. Daniel, and T. H. Machado, "Feature analysis by k-means clustering for damage assessment in rotating machinery with rolling bearings," *J. Brazilian Soc. Mech. Sci. Eng.*, vol. 44, no. 8, p. 330, Aug. 2022.
[18] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[19] M. Amarnath, V. Sugumaran, and H. Kumar, "Exploiting sound signals for fault diagnosis of bearings using decision tree," *Measurement*, vol. 46, no. 3, pp. 1250–1256, Apr. 2013.
[20] Y. Li, W. Dai, and W. Zhang, "Bearing fault feature selection method based on weighted multidimensional feature fusion," *IEEE Access*, vol. 8, pp. 19008–19025, 2020.
[21] B. R. Nayana and P. Geethanjali, "Analysis of statistical time-domain features effectiveness in identification of bearing faults from vibration signal," *IEEE Sensors J.*, vol. 17, no. 17, pp. 5618–5625, Sep. 2017.
[22] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," 2021, *arXiv:2103.13630*.
[23] S. Sarangi and B. Baas, "DeepScaleTool: A tool for the accurate estimation of technology scaling in the deep-submicron era," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.
[24] H. Huang and N. Baddour, "Bearing vibration data collected under time-varying rotational speed conditions," *Data Brief*, vol. 21, pp. 1745–1749, Dec. 2018.

**JAN TIMMER** received the B.Sc. degree in electrical engineering from the Avans University of Applied Sciences, Den Bosch, The Netherlands, in 2021, and the M.Sc. degree in electrical and electronic engineering from Eindhoven University of Technology, Eindhoven, The Netherlands, in 2024. His research interests include FPGA and ASIC designs and hardware-efficient accelerators.

**ARDAVAN ELAHI** (Member, IEEE) received the B.Sc. degree in computer engineering from the Isfahan University of Technology, Iran, in 2014, and the M.Sc. degree in computer engineering from the University of Tehran, Iran, in 2017. He is currently pursuing the Ph.D. degree in computer engineering with Vienna University of Technology (TU Wien), Austria. His research interests include the optimization of AI algorithms and the design of hardware-efficient architectures for AI workloads.

**PHILIPP LEHNINGER** (Member, IEEE) received the B.S. degree in computer engineering and the M.S. degree in embedded systems from Vienna University of Technology (TU Wien), in 2022 and 2025, respectively, where he is currently pursuing the Ph.D. degree in electrical engineering.

His research interests include digital design for hardware-constrained applications, beyond-CMOS technologies, and asynchronous design.

**AXEL JANTSCH** (Fellow, IEEE) received the Dipl.Ing. and Ph.D. degrees in computer science from Vienna University of Technology (TU Wien), Vienna, Austria, in 1987 and 1992, respectively. From 1997 to 2014, he was an Associate Professor with KTH. Since 2002, he has been a Full Professor of electronic systems design with the Royal Institute of Technology, Stockholm, Sweden. Since 2014, he has been a Professor of systems on chips with TU Wien.