A Low-Power Hierarchical CNN Hardware Accelerator for Bearing Fault Diagnosis

Yu-Pei Liang, *Member, IEEE*, Yao-Shun Hsu, and Ching-Che Chung, *Senior Member, IEEE*

***Abstract*— This article presents a 2-D hierarchical convolutional neural network (HCNN) hardware accelerator that is implemented in a 40-nm CMOS technology for Case Western Reserve University (CWRU) bearing fault diagnosis. The hierarchical structure of the convolutional neural network (CNN) contributes to a reduction in both power consumption and computation time. The entire neural network parameters are 29k, and the total CNN computation is completed within 330 000 cycles, showcasing its real-time capability. The proposed design substantially diminishes the number of cycles necessitated for hardware calculations. Furthermore, this work incorporates Gaussian white noise into the vibration signal dataset for signal-to-noise ratio (SNR) analysis. A noisy training dataset is added to the original dataset for neural network training to improve the accuracy. In summary, the postlayout simulation of the proposed design facilitates real-time fault diagnosis at a clock frequency of 100 MHz, achieving an accuracy of 95.31%, and a power consumption of 65.608 mW. Also, when the proposed HCNN circuit was implemented on a field-programmable gate array (FPGA) evaluation board, it consumed 0.533 W at 55 MHz.**

***Index Terms*— Convolution, digital circuits, digital signal processing, fault diagnosis, field-programmable gate arrays (FPGAs), fixed-point arithmetic, neural networks, quantization, real-time systems, signal sampling.**

I. Introduction

RECENTLY, electronic devices have been widely used in our daily lives. One of the most important types of devices in manufacturing for improving efficiency is the bearing device. The faults of these bearing devices can significantly impact the efficiency of manufacturing. In other words, if the faulty part of the bearing device can be diagnosed earlier, the production line can be restored as soon as possible, and maintenance costs can be reduced. Therefore, detecting bearing faults is an essential research topic in this field. Regarding bearing fault detection, researchers often utilize publicly available datasets, such as Case Western Reserve University (CWRU) [1], Paderborn [2], and Mechanical Failure Prevention Technology (MFPT) [3], to conduct their investigations.

First of all, the CWRU dataset is employed for computer numerical control (CNC) machines, capturing the vibration sensor data for normal, drive-end (DE), and fan-end (FE) defective bearings. More specifically, CWRU bearing data includes three kinds of faults: normal bearing, single-point DE, and FE. The FE bearing data are acquired at 12 000 samples/s, while the DE bearing data exhibit two sampling rates, specifically 12 000 and 48 000 samples/s. In this articl, the FE bearing data are utilized in experiments. Nevertheless, it is important to note that certain noisy industrial settings may introduce noise and subsequently give rise to unstable factors.

Bearing failures due to minor damage are difficult to identify. Therefore, prior researchers strive to classify and identify these faults through various methodologies. For instance, Waziralilah et al. [25] examined multiple studies that employed convolutional neural networks (CNNs) for bearing fault diagnosis and highlighted the existence of three principal bearing faults: ball faults (BFs), outer race (OR) faults, and inner race (IR) faults. Moreover, the fault types were further subdivided into nine fault types in [4]; each characterized by fault diameters ranging from 0.007 to 0.021 in, associated with the three faults mentioned above (IR, OR, and BF) in the CWRU dataset. In addition, the normal condition (NC), which indicates a fault-free state, can be considered an additional type so that the bearing fault types can be extended to the ten types in the CWRU dataset.

In addition to the CWRU dataset, the Paderborn dataset [2], [6], [7] is another popular dataset for bearing faults detection. Compared with the CWRU dataset, the Paderborn dataset gathers current and vibration signals, and the sampling rate is 64 kHz. Furthermore, the Paderborn dataset can be mainly divided into four types: normal, OR fault (ORF), IR fault (IRF), and outer IR fault (OIRF). The data within the Paderborn dataset are obtained through accelerated life testing and the manual acquisition of damage data related to bearing failures. On the other hand, the MFPT dataset is also a commonly used dataset for fault detection and diagnosis, and it consists of three sets of bearing vibration data, including the IR and OR data under various loads from 0 to 1.34 kN. There were several researchers [8], [9] used the MFPT dataset as their research target.

While numerous outstanding researchers are actively involved in this field, and their work boasts remarkable accuracy, there has been little consideration given to the

Manuscript received 27 September 2023; revised 29 November 2023; accepted 18 December 2023. Date of publication 8 January 2024; date of current version 22 January 2024. This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST-111-2221-E-194-049 and in part by the Advanced Institute of Manufacturing With High-Tech Innovations (AIM-HI) through The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan. The Associate Editor coordinating the review process was Dr. Gabriele Patrizi. *(Corresponding author: Ching-Che Chung.)*

The authors are with the Department of Computer Science and Information Engineering and the Advanced Institute of Manufacturing With High-Tech Innovations, National Chung Cheng University, Chiayi 62102, Taiwan (e-mail: wildwolf@cs.ccu.edu.tw).

Digital Object Identifier 10.1109/TIM.2024.3351229

memory usage and power consumption associated with detecting and classifying bearing faults. Notably, there have been some excellent research efforts focused on reducing resource requirements, such as memory usage and power consumption, for machine learning models. For instance, MobileNet [23] incorporates depthwise separable convolution to decrease the model architecture size and reduce parameters. More recently, microcontroller units (MCU) Net [24] further optimizes parameters to enable its performance on the Internet-of-Things (IoT) devices. However, these models are generally designed for broader applications, and their memory usage and power consumption might not be fully optimized for specific use cases like bearing fault diagnosis. As a consequence, when deployed in resource-constrained environments, the accuracy of such solutions tends to decrease to some extent. In other words, integrating the existing solutions into CNC machines for real-time accurate detection is still a big challenge.

On the other hand, more and more hardware accelerators are proposed in various excellent literature sources [26]. However, in deep neural network (DNN) accelerators that utilize a spatial architecture implemented on an application-specified integrated circuit (ASIC) or field-programmable gate array (FPGA), the critical constraint arises from the efficiency of memory access. Therefore, among various hardware accelerator research endeavors, in-memory computing techniques gain a lot of attention to accelerate DNN computation. For example, in PattPIM [27], a resistive random-access memory (ReRAM) crossbar array optimizes space and computation via CNN weight pattern repetition for efficient compression and reuse. To address the nondeterministic results in processing in-memory technology, a framework was introduced in [28] to systematically evaluate the accuracy of analog in-memory computing across various network topologies. The study investigates sensitivity and robustness to a broad spectrum of nonidealities. Although methods based on in-memory computing significantly improve the computation time of deep learning, they usually require special kinds of memory (such as ReRAM), which typically raises the hardware cost.

Notably, the production cost is usually one of the most critical considerations in the industrial field. In order to provide a low-cost and real-time solution for analyzing bearing faults, we believe that designing a specialized integrated circuit (IC) could be a prudent choice. To the best of our knowledge, there has been no prior work focusing on developing a low-cost, real-time machine-learning-based hardware solution for processing data from the CWRU dataset. As the pioneer work in this research field, the primary objective is to develop a solution that can be implemented as an IC and directly applied to CNC machines for real-time fault detection. In addition to the computing time and accuracy, to control the cost of the designed IC, in this article, we minimize the memory usage by reducing the bitwise and number of parameters in our model. In addition, we also reduce power consumption by the hierarchical structure and power gating technology. More specifically, this article proposes a real-time hierarchical CNN (HCNN) with a power gating feature for diagnosing bearing faults. The hierarchical structure helps the model terminate in the early stage in normal cases to prevent unnecessary computing and reduce the computing time and energy.

Moreover, the power gating technology turns off the components' power after being used to mitigate power consumption further. On the other hand, this article demonstrates a design flow of the IC design for a specific application and provides other researchers with a framework to design other solutions. Furthermore, the proposed solution has been compared with other solutions in software simulations and implemented as hardware for verification. Finally, the CWRU dataset is considered the most comprehensive and complex dataset for bearing fault diagnosis. Thus, it has been selected as the study case in this article.

The main contributions of this article can be summarized as follows.

1) The proposed HCNN demonstrates reduced memory and power requirements compared with the existing methods.
2) The hardware implementation of the proposed HCNN, with postlayout simulation results, indicates lower power consumption and resource usage than other existing solutions.
3) The computing time of the proposed HCNN satisfies real-time requirements, making it suitable for implementation in hardware and adoption in CNC machinery for real-time diagnostics.

The remainder of this article is organized as follows. Section II presents an overview of the related work on bearing fault diagnosis. Section III shows the architecture of the proposed HCNN. Section IV describes the hardware implementation of the proposed HCNN. Section V provides the postlayout simulation and FPGA implementation results, while the conclusion is presented in Section VI.

II. Related Work

Bearing failure is a critical concern in manufacturing, leading to numerous studies focused on analyzing and classifying bearing fault types in the past. Alessandro Paolo et al. [5] asserted that the power spectrum density (PSD) diagram of vibration data could distinguish between various faults. Within the context of the PSD diagram, discrete frequency components are identifiable across a broad frequency spectrum in the signal captured by the accelerometer. Therefore, upon the manifestation of a bearing fault, the fault frequency of the bearing can be determined by analyzing the frequency spectrum of the vibration signal.

After that, various methods and tools have been proposed and assessed, particularly machine learning-based approaches [10], [11], such as support vector machines (SVMs), random forests, CNN, and DNN.

First, SVM is a supervised learning method that aims to identify a classification decision boundary to separate two distinct data types. As a result, SVM is better suited for binary classification tasks. While SVM has been extensively employed in bearing fault analysis applications, its accuracy in past architectures ranges from 60% to 90% [12]. Moreover, in the case of complex data, it will affect the classification ability of SVM. Consequently, Lee et al. [13]

combined autoencoder (AE) and SVM techniques to enhance bearing faults diagnostics accuracy. Furthermore, previous research [10], [11] has demonstrated that the accuracy of existing bearing fault diagnostics using ML-based approaches (e.g., deep belief network (DBN), k-nearest neighbors (KNNs), CNN, and DNN) typically exceeds 90%, with CNN-based methods generally achieving higher accuracy.

In [14], principal component analysis (PCA) and linear discriminant analysis (LDA) were employed for three-phase induction motor bearings. A learner composed of multiple decision trees combines several "weak learners" to create a more robust model called a "strong learner." This approach is also known as the ensemble method. In [15], random forest and CNN were combined, with results demonstrating that the random forest algorithm can effectively diagnose faults when provided with appropriate feature extraction.

CNN is commonly employed for image recognition, with numerous models built upon CNN architecture. The primary goal of convolution is to extract local features from images and subsequently perform image classification. However, the output remains linear following the convolution layer, which poses limitations when simulating and identifying more complex data. Furthermore, the activation function is crucial for a neural network to achieve nonlinear output. Therefore, if the nonlinear activation function is not used, the model trained by the neural network is meaningless.

In [7], an input feature mapping (IFMs)-based deep residual network (ResNet) has been proposed for the Paderborn dataset, and the accuracy can achieve almost up to 100%.

For the MFPT dataset, Sharma et al. [8] developed a 1-D CNN approach, which achieved an impressive accuracy of up to 98.9%. Sun et al. [9] initially employed a second-order time-assigned multisynchrosqueezing transform to introduce a novel time–frequency analysis technique to obtain higher resolution time–frequency images for training the CNN model. The simulation outcomes demonstrated that the proposed CNN network achieved recognition accuracies of 99.83% and 98.67% for the CWRU and MFPT datasets, respectively.

During calculations, neural networks undergo millions of operations, resulting in significant power consumption and posing challenges for hardware implementation. Goel et al. [16] proposed a modular neural network tree architecture to address this issue. This approach divides the classification results into several groups and trains a submodel for each group, organizing these submodels into a tree structure. Once an image has been identified as belonging to a group of categories, the corresponding submodel works further to distinguish the target image within a more detailed subgroup. This process is repeated across multiple modules until the final classification comes out. Experimental results demonstrate that this HCNN tree can reduce memory requirements and power consumption by 50%–90%.

More recently, Chung et al. [21] introduced a 1-D CNN model designed to process current data from the Paderborn dataset while also aiming to devise a real-time hardware solution. In their study, the authors proposed a down-sampling method and a quaternary quantization technique to enhance the model's accuracy. Furthermore, they reduced the memory requirements by limiting the bit usage within the model.

III. Proposed HCNN Architecture

A. HCNN Architecture Overview

This article introduces a hierarchical CNN-based method, called HCNN, for diagnosing bearing faults with low power consumption and can be used for real-time diagnostics. As previously mentioned, the CWRU dataset is widely used and recognized as the most comprehensive dataset for bearing fault research. Therefore, this work employs it as the study case for designing the method.

Initially, the CWRU dataset must be preprocessed to generate suitable input images for neural network architecture. In this article, every 4096 vibration signal points in the CWRU dataset are taken to produce a 64×64 image, resulting in a total of 8424 generated images. After shuffling the images, 80% of them (i.e., 6744 images) are employed to train the CNN model, while the remaining 20% (i.e., 1641 images) are used to test the model.

Following image preprocessing, the HCNN is trained using the images. This study groups datasets of the same faulty bearing at different levels into the same category to establish the hierarchy. More specifically, a three-layer root model performs four classifications, followed by three two-layer submodels for the final classification. Each convolutional layer comprises three typical architectures: 3×3 kernels, batch normalization (BN), and Rectified Linear Unit (ReLU).

When considering hardware implementation, it is crucial to reduce the number of parameters as much as possible to conserve resources. However, decreasing parameters directly impacts the model's accuracy, leading to a trade-off between hardware resources and accuracy. This work conducts several experiments to establish the final model (including the root and child models) and determine the appropriate balance between resource allocation and accuracy.

For a clearer understanding of the workflow, Fig. 1 shows the flowchart for designing the software HCNN model. As depicted in Fig. 1, the initial step involves converting the signal data into the CWRU bearing fault dataset from floating-point numbers to fixed-point numbers and arranging them into images for training and testing. In the second step, the images mentioned above are used to train the proposed HCNN repeatedly until a predefined condition is satisfied. Note that, the processes described above work on a famous machine-learning framework, Pytorch. When the accuracy and number of parameters meet the requirements in Pytorch, the subsequent step involves training the neural network's weights and BN layers. While PyTorch is powerful and convenient for developing machine learning models, it does not provide detailed implementation information for each application programming interface (API) and does not support variable bit length settings. Therefore, we then write Python code to implement each API in detail and to test the influence of various bit lengths on parameters to decide the bit lengths for hardware design. Notably, the parameters are extracted from the training results of PyTorch.

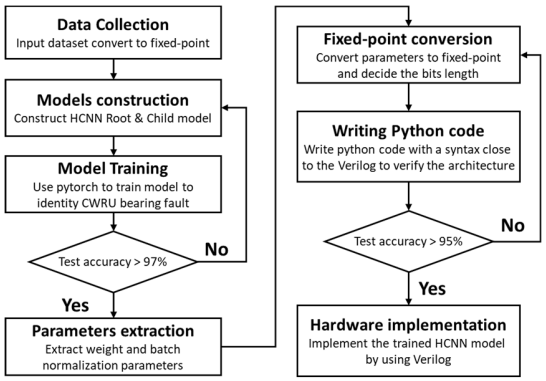

Fig. 1. Flowchart for designing the software HCNN model.

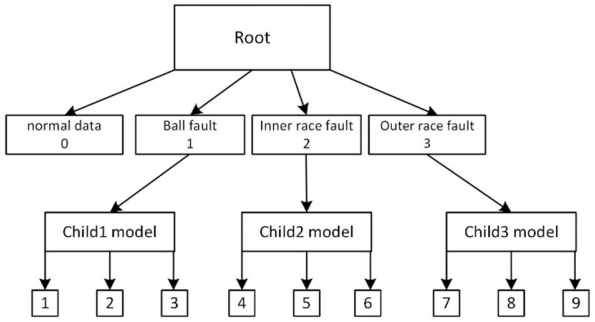

Fig. 2. Hierarchical CNN model corresponds to Table I.

TABLE I
LABEL AND FAULT TYPES IN CWRU DATASET

Root Label	Child Label	Fault Diameter (inch)	Type of fault	IDs on CWRU bearing data center website [1]
0	N/A	0	Normal Healthy Data	Normal_0 to Normal_3
1	1	0.007	Ball Fault	B007_0 to B007_3
1	2	0.014	Ball Fault	B014_0 to B014_3
1	3	0.021	Ball Fault	B021_0 to B021_3
2	4	0.007	Inner Race Fault	IR007_0 to IR007_3
2	5	0.014	Inner Race Fault	IR014_0 to IR014_3
2	6	0.021	Inner Race Fault	IR021_0 to IR021_3
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,
3	8	0.014	Outer Race Fault	OR014@6_0, OR014@3_0 to OR014@3_3
3	9	0.021	Outer Race Fault	OR021@6_0, OR021@3_1 to OR021@3_3

Moreover, Python code ensures that the computational method closely resembles hardware implementation, so the syntax used in the Python code is designed to be similar to Verilog. Furthermore, to facilitate a better understanding of the HCNN's implementation in Python code, we have provided the source code on GitHub (URL: https://github.com/fcu-D0550770/Hierarchical-CNN-CWRU). Finally, the hardware implementation can start once the accuracy and parameters are confirmed.

The FE accelerometer data and FE bearing data used in this article are gathered at 12 000 samples/s. Table I provides a detailed summary of the fault types in the dataset, including IRF, BF, and ORF with different fault diameters (i.e., 0.007, 0.014, and 0.021). The normal healthy bearing condition is also considered as a different type, bringing the total number of types in the classification model to ten.

The proposed HCNN model has two layers of classifications for the final decision. More specifically, a root model is initially used to distinguish fault types (i.e., healthy data, BF, IRF, and ORF). Then, three child models are employed to classify fault diameters within each fault type. This means that the first layer of the classification method represents fault types. As shown in Table I, each type shares the same root label number. On the other hand, the second layer of the hierarchical CNN represents fault diameters, so the child models determine the final decision of the bearing fault with nine detailed labels. Fig. 2 illustrates the architecture of the proposed HCNN model, with labels corresponding to those in Table I. Note that the last column in Table I represents the corresponding dataset IDs on the CWRU bearing data center website [1].

TABLE II
INPUT AND OUTPUT DATA SIZES OF THE ROOT MODEL

Input (width×height×channel)	Operator	Output (width×height×channel)	stride
64×64×1	CONV	64×64×8	1
64×64×8	Max Pooling	32×32×8	2
32×32×8	CONV	32×32×16	1
32×32×16	Max Pooling	16×16×16	2
16×16×16	CONV	16×16×16	1
16×16×16	Max Pooling	8×8×16	2
1×1×1024	FC	1×1×4	-

TABLE III
INPUT AND OUTPUT DATA SIZES OF CHILD1–CHILD3 MODEL

Input (width×height×channel)	Operator	Output (width×height×channel)	stride
8×8×16	CONV	8×8×16	1
8×8×16	Max Pooling	4×4×16	2
4×4×16	CONV	4×4×32	1
4×4×32	Max Pooling	2×2×32	2
1×1×128	FC	1×1×3	-

To further elaborate on the architecture of the proposed HCNN, Tables II and III present the number of layers and channels in each layer for the root model and Child1–Child3 models, respectively.

Using the architecture of the proposed HCNN, the total number of parameters is approximately 29k. The sum of the model parameters is shown in Table IV.

The accuracy of the proposed HCNN software model before hardware implementation is shown in Table V. A total of

TABLE IV
PARAMETERS IN HCNN

Input channel	Operator	Output channel	Sum parameters
1	Root Convolution 1 (3×3)	8	72
8	Batch Normalization (2 parameters)	8	16
8	Root Convolution 2 (3×3)	16	1,152
16	Batch Normalization (2 parameters)	16	32
16	Root Convolution 3 (3×3)	16	2,304
16	Batch Normalization (2 parameters)	16	32
16	Root Fully connected (FC)	4	4,096
16	Child Convolution 1 (3×3)	16	2,304
16	Batch Normalization (2 parameters)	16	32
16	Child Convolution 2 (3×3)	32	4,608
32	Batch Normalization (2 parameters)	32	64
32	Child Fully connected (FC)	3	384
Total parameters Root (7,704)+ Child 1~3 (7,392 × 3)			29,880

TABLE V
FINAL ACCURACY BEFORE HARDWARE IMPLEMENTATION

Type of fault	Number of images (Correct/Test)
Root label 0	85/85
Root label 1	356/376
Root label 2	338/352
Root label 3	787/828
Final accuracy	1,566/1,641 = 95.43%

1641 images were used to test the accuracy of the proposed HCNN architecture, and 75 images were misclassified. In Table V, input sensor data, weight, and BN values are represented using fixed-point notation. Furthermore, when solely expressing input sensor data with fixed-point notation, the accuracy of the proposed HCNN reaches 97.7%.

B. CWRU Dataset With SNR Analysis

In addition to the classification model for the CWRU dataset, this work adds the white Gaussian noise to the CWRU dataset to assess the noise immunity capability of the proposed HCNN. Different noise levels impact the original signal and influence recognition accuracy. Generally, the signal-to-noise ratio (SNR) value measures the magnitude of the noise to the signal. A smaller SNR value indicates more significant noise effects, while a larger SNR value signifies reduced noise effects and less degradation of the original signal.

After considering the SNR range used in other papers, this study adds white Gaussian noise to the CWRU dataset. The SNR experimental range includes −4, −2, 0, 2, 6, 8, and 10, comprising seven groups. Finally, a comparison of different noise levels is provided to assess the antinoise performance of the proposed HCNN model for the CWRU dataset, as shown in Table VI.

When the white Gaussian noise is only added to the test dataset, as shown in Table VI, the accuracy of both the root and child models declines due to the reduced recognition accuracy caused by noise. The results also reveal that Child2 model is relatively less affected by noise. Therefore, to enhance the model's resistance to noise, the training dataset with noise will be incorporated into the training dataset in subsequent experiments. As shown in Tables VII and VIII, the accuracy of root and Child1 and Child3 models can be significantly improved.

The results show that the accuracy of the overall model significantly improves when noise is added to the training dataset. Thus, it can be inferred from the experimental outcomes that incorporating an appropriate amount of noisy training datasets can enhance the resistance of the proposed HCNN to noise.

TABLE VI
IMPACT OF NOISE ON THE PROPOSED HCNN FOR THE CWRU DATASET

SNR (dB)	Root	Child1	Child2	Child3
-4	73%	65.7%	95.14%	81.7%
-2	85.17%	78.6%	98.65%	93.61%
0	90.17%	77.4%	98.38%	95.9%
2	93.9%	86.5%	99.46%	94.33%
6	97.2%	89%	99.73%	98.31%
8	97.97%	88%	99.46%	98.43%
10	98.27%	90%	99.19%	99.28%

TABLE VII
IMPROVE THE ACCURACY OF THE ROOT MODEL WITH NOISE

SNR	Root model Train: original training data Test: noisy test data	Root model Train: original + noisy training data Test: noisy test data
-4	73%	83%
-2	85.17%	88%
0	90.17%	93%
2	93.9%	97.2%
6	97.2%	98.15%
8	97.97%	98.8%
10	98.27%	98.93%

TABLE VIII
IMPROVE THE ACCURACY OF CHILD1 AND CHILD3 MODELS WITH NOISE

SNR (dB)	Child1 model Train: original training data Test: noisy test data	Child1 model Train: original + noisy training data Test: noisy test data
-4	65.7%	75.6%
-2	78.6%	76.4%
0	77.4%	88%
2	86.5%	91.6%
6	89%	93.4%
8	88%	95.68%
10	90%	97.2%

SNR (dB)	Child3 model Train: original training data Test: noisy test data	Child3 model Train: original + noisy training data Test: noisy test data
-4	81.7%	92.2%
-2	93.61%	96%
0	95.9%	98.9%
2	94.33%	98.43%
6	98.31%	98.31%
8	98.43%	99.63%
10	99.28%	99.39%

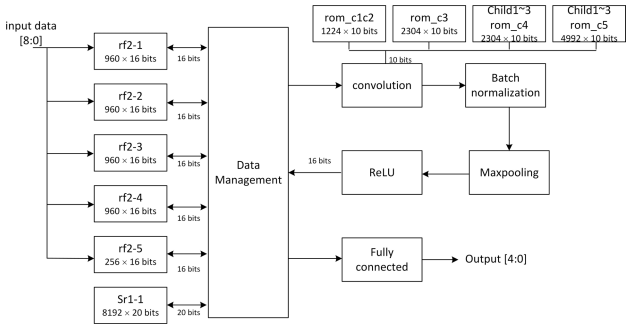

Fig. 3. Proposed HCNN hardware architecture.

IV. Hardware Implementation

This section provides a detailed description of the memory allocation and the proposed hardware implementation. The hardware architecture of the proposed HCNN is shown in Fig. 3.

As depicted in Fig. 3, the register files and static random access memory (SRAM) are utilized to read the input data and write the feature map of each layer. In contrast, the read-only memory (ROM) is responsible for storing the weights of each layer. Moreover, the convolution block handles the calculations for zero-padding and convolution. After the calculations are completed, the resulting values are sent to the BN block to compute the two parameters, gamma, and beta, of BN. The max-pooling block carries out the pooling process, reducing the size of the feature map. Finally, the ReLU block is responsible for activation operations and activation quantization. Then, the output of the final model judgment result of the fully connected (FC) block is finally performed.

In hardware implementation, it is crucial to simplify the preprocessing stage and ensure that it can be represented in a fixed-point format for hardware operations. To achieve this, one practical approach is to reduce the number of bits required for data representation. By doing so, computational efficiency can be enhanced during hardware implementation, and memory requirements can be decreased. This contributes to better resource utilization and enables faster and more efficient processing, which is particularly important for real-time applications.

In Section III-A, it was mentioned that to implement the model in hardware, the image data input must be converted into fixed-point representation. Similarly, for the convolution operation with the weights and the subsequent operations with the gamma and beta of BN, these values should also be quantified using fixed-point representation to facilitate hardware implementation. Before proceeding with the hardware implementation, it is essential to determine the appropriate bit width for all weights and BN values. This is because the bit-width directly impacts the model's accuracy. To ensure that the selected bit width does not significantly compromise the model's performance, verification should be conducted using Python code. By carefully choosing the appropriate bit width, the trade-off between the resource usage and model accuracy can be effectively managed, ultimately resulting in a hardware implementation that meets the desired requirements.

TABLE IX
Test Accuracy for Weight Bit Width With BN Fixed at 9 Bits

Integer bits	Decimal bits	Total bits	Test accuracy
2	11	13	0.956
2	10	12	0.9554
2	9	11	0.95333
2	**8**	**10**	**0.9543**
2	7	9	0.94285
2	6	8	0.8762

TABLE X
Test Accuracy for BN Bit Width With Weight Fixed at 10 Bits

Integer bits	Decimal bits	Total bits	Test accuracy
4	8	12	0.95461
4	7	11	0.95287
4	6	10	0.95521
4	**5**	**9**	**0.9543**
4	4	8	0.932142
4	3	7	0.85416

TABLE XI
Memory Usage for the Feature Maps of Each Layer

Stored data	Feature map size (width×height×channel × bit-width)	Total size (bits)	Data management
Feature maps of layer 1	32 × 32 × 8 × 20	163,840	Sr1-1
Feature maps of layer 2	16 × 16 × 16 × 16	65,536	rf2-1, rf2-2, rf2-3, rf2-4, rf2-5
Feature maps of layer 3	8 × 8 × 16 × 16	16,384	Sr1-1
Feature maps of layer 4	4 × 4 × 16 × 16	4,096	rf2-5
Feature maps of layer 5	2 × 2 × 32 × 16	8,192	rf2-1

Table IX presents the experimental results, which indicate that if the decimal bits of weight are too small, it can significantly impact the overall accuracy. Several attempts were made to determine the appropriate number of bit widths during the software verification stage. The results show that the model can maintain acceptable accuracy when the weights are represented with two integer bits and eight decimal bits, given that BN values are fixed at 9 bits. This configuration strikes a balance between resource usage and model accuracy, making it suitable for hardware implementation.

Table X presents the experimental results for various bit-width configurations of BN values when the weights are fixed with two integer bits and eight decimal bits. After examining the results from Tables IX and X, it was determined that the optimal configuration for hardware implementation is the weights with two integer bits and eight decimal bits, BN values with four integer bits and five decimal bits, and input sensor data with four integer bits and five decimal bits. This configuration achieves a balance between resource usage and model accuracy, resulting in an acceptable accuracy of 95.43%.

Table XI presents the memory requirements for the hardware implementation of the proposed HCNN. Since the input image size is 64 × 64, and each sensor data has 9 bits (4 bits for integer and 5 bits for decimal), the memory allocation

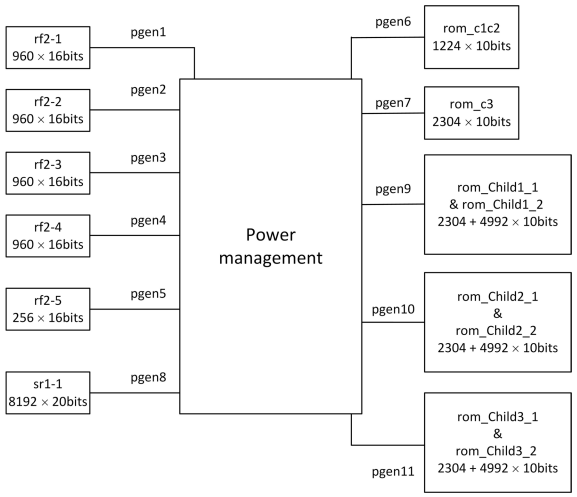

Fig. 4. Proposed design power control diagram.

for each register file (rf2-1, rf2-2, rf2-3, rf2-4, and rf2-5) is calculated accordingly. For the second layer, rf2-1, rf2-2, rf2-3, rf2-4, and rf2-5 are responsible for reading the initial image data and the feature maps. For the fourth and fifth layers, due to the reduction in image size resulting from max-pooling, only one rf2-5 and rf2-1 are required for writing and reading the data. In total, the memory requirement for rf2-1, rf2-2, rf2-3, rf2-4, and rf2-5 is 65 536 bits, as shown in Table XI.

In the proposed HCNN, since the feature map of the first layer is too large and exceeds the size limitation by the register file memory compiler, an additional SRAM memory component, Sr1-1, is used to handle the read and write operations for the first and third layer feature map data. The size used by Sr1-1 is 163 840 bits. In the proposed design, Sr1-1 and the five register files mentioned earlier (rf2-1, rf2-2, rf2-3, rf2-4, and rf2-5) work interactively to manage the operations, reads, and writes between layers. This approach helps to optimize memory usage and maintain efficient processing across the hierarchical CNN model. The overall memory reduction, including register files, SRAM, and ROM, is 60.9%, benefitting from quantizing the input sensor data, the model parameter, and the feature map.

The power gating technology is employed further to reduce power consumption in the proposed hardware architecture. Power gating involves shutting off the power supply to idle circuit modules, effectively minimizing the system's overall power consumption. As a result, the proposed HCNN can achieve low power consumption and real-time performance for diagnosing bearing faults by integrating power gating into the hardware design.

In the proposed HCNN hardware implementation, power gating is integrated to minimize power consumption while maintaining real-time performance. Fig. 4 illustrates the architecture with power control features. As shown in Fig. 4, there are multiple power gating switches included in the design.

1) *pgen1, pgen2, pgen3, pgen4, and pgen5:* These switches control the power gating for the register files in the design, effectively managing the power supply to the register files based on their utilization.

TABLE XII
POWER GATING CONTROL WHEN RUNNING CHILD1 MODEL

layer \ Power domain	pgen1	pgen2	pgen3	pgen4	pgen5	pgen6	pgen7	pgen8	pgen9
Root layer1	***On***	***On***	***On***	***On***	***On***	***On***	Off	***On***	Off
Root layer 2	***On***	***On***	***On***	***On***	***On***	***On***	Off	***On***	Off
Root layer 3	Off	***On***	***On***	***On***	***On***	Off	***On***	***On***	Off
RootFC	Off	Off	Off	Off	Off	Off	Off	***On***	Off
child layer1	Off	Off	Off	Off	***On***	Off	Off	Off	***On***
child layer 2	***On***	Off	Off	Off	***On***	Off	Off	Off	***On***
child FC	***On***	Off	Off	Off	Off	Off	Off	Off	***On***

2) *pgen6, pgen7, pgen9, pgen10, and pgen11:* These switches control the power gating for the ROMs in the design. By managing the power supply to the ROMs, the power consumption can be optimized based on the ROMs' usage.
3) *pgen8:* This switch controls the power gating for the Sr1-1 in the design, managing the power supply to the SRAM based on its utilization.

These power gating switches enable the HCNN hardware implementation to optimize power consumption by controlling the power supply to different components based on their utilization. This results in a more energy efficient for diagnosing bearing faults.

Incorporating power gating technology into the proposed HCNN architecture enables a more energy-efficient design by dividing register, SRAM, and ROM components into multiple power domains. These power domains can be powered on or off interactively depending on the operational requirements of each layer. This approach reduces power consumption during idle periods, leading to a more power-efficient system overall.

In the TSMC 40-nm memory compiler, there is an option to use memory modules with power-gating capabilities. By using these memory modules and leveraging the power gating switch control, the overall power consumption of the proposed HCNN hardware can be significantly reduced.

In accordance with different modes, the initial three layers constitute the root model, while the final two layers form the child model. The power domains are activated or deactivated accordingly, with the corresponding switches for each layer shown in Table XII. As shown in Table XII, the memory utilized in every layer is activated. Conversely, when the memory is idle, it is deactivated. This design persists throughout the entire neural network model operation, culminating with the FC layer outputting the classification results.

The comprehensive operation flowchart for the entire hardware is illustrated in Fig. 5. Once the third layer of convolution is completed, the root classification result is output through the FC layer, which determines which child models (Child1, Child2, or Child3) to enter. Subsequently, the process advances to the final two convolutional layers. Ultimately, the classification is determined through the child model's FC output, yielding the final identification and classification result. Finally, power gating is executed concurrently with the hardware circuit operations, and idle memory is switched

TABLE XIII
COMPARISONS WITH OTHER ARCHITECTURES IN THE SOFTWARE PLATFORM

	[17] IEEE Access'21	[18] MLBDBI'21	[19] IEEE Sensors Journal'19	[20] IEEE TIM'21	Proposed work
Architecture	SN-CNN	MS-CNN	CNN+RNN	CGAN+CNN	HCNN
Dataset	CWRU	CWRU	CWRU	CWRU	CWRU
Parameters(k)	258.6	98	355.2	314.7	29
Preprocessing method	Flipping Samples	continuous wavelet transform scalogram	equitable sliding stride segmentation (ESSS)	Convert to 2D gray image	Convert to the fixed-point and reduce bits
Quantization method	No	No	No	No	Reduce bits
Classified labels	10	10	10	10	10
Accuracy	99.4%	98.5%	99.92%	98.6%	97.7%

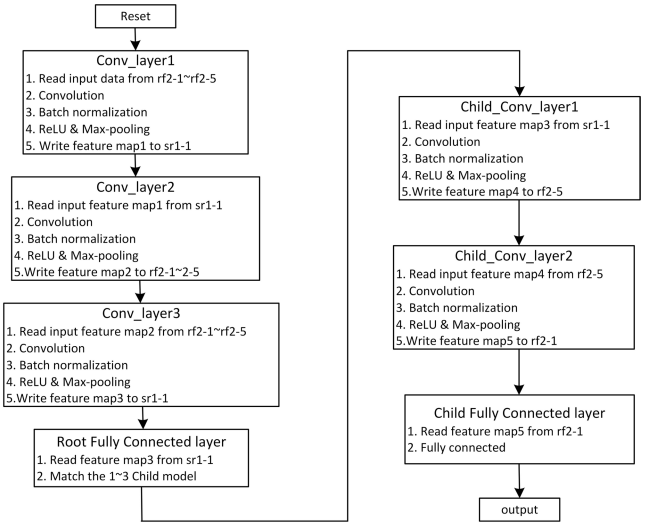

Fig. 5. Flowchart for the proposed HCNN hardware circuit.

off via the signal line. This approach achieves the goal of minimizing power consumption in the design.

V. EXPERIMENTAL AND SIMULATION RESULTS

To evaluate the proposed HCNN, it is implemented in both software and hardware implementations. Consequently, in Sections V-A and V-B, the experimental results of software and the postlayout simulation results of hardware implementation will be reported in detail.

A. Software Implementation

First, Table XIII presents a comparison of the proposed HCNN software implementation with other state-of-the-art methods. As can be observed from Table XIII, the parameter number of the proposed HCNN is significantly lower than that of other recent studies focusing on the CWRU dataset. More specifically, the overall parameter number is only 29 K without sacrificing much accuracy. In other words, the proposed HCNN can still achieve over 97% accuracy with merely 29k parameters. Regarding quantization, the method of limiting bits is used, and make adjustments continuously during the process. This means that the experiment is not deemed complete until the software stage reaches an acceptable level of accuracy and falls within an appropriate range.

Conversely, a control group that employs the conventional CNN architecture to explore the benefits of the proposed design's hierarchical architecture is created. Under the same CWRU training dataset and comparable accuracy to the HCNN architecture, the control group also utilizes a five-layer network structure incorporating BN and ReLU. The specifics of the control group are outlined in Table XIV.

TABLE XIV
CONTROL GROUP NEURAL NETWORK ARCHITECTURE

Input (width×height×channel)	Operator	Output (width×height×channel)	Stride
64×64×1	CONV	64×64×8	1
64×64×8	Max Pooling	32×32×8	2
32×32×8	CONV	32×32×16	1
32×32×16	Max Pooling	16×16×16	2
16×16×16	CONV	16×16×32	1
16×16×32	Max Pooling	8×8×32	2
8×8×32	CONV	8×8×32	1
8×8×32	Max Pooling	4×4×32	2
4×4×32	CONV	4×4×64	1
4×4×64	Max Pooling	2×2×64	2
1×1×256	FC	1×1×10	-

Table XIV reveals that, compared with the control group, there is a substantial increase in the parameter count by 7.6k in the control group when the accuracy between the two architectural groups is comparable. This increase results in additional overheads for subsequent hardware implementation. Consequently, when contrasted with the control group, it is evident that the HCNN architecture offers significant benefits for hardware implementation.

In addition, floating-point operations (FLOPs) are a metric used to assess the computational complexity of a neural network. Consequently, the FLOPs of the control group and the HCNN architecture are listed in Table XV. It is readily evident that the FLOPs of the proposed HCNN architecture are fewer than those of the control group. This indicates that the computational complexity of the HCNN model is less than that of the conventional neural networks.

B. Hardware Implementation

As previously mentioned, it is important to note that there have been no previous works that have implemented their solutions on hardware for handling the data from the same dataset. As a result, for the evaluation of the hardware implementation

TABLE XV
COMPARISONS WITH AN EXPERIMENTAL CONTROL GROUP

	Traditional CNN	My work HCNN
Architecture	5 layers convolutional, Batch normalization, and ReLU.	Root 3 layers & Child 2 layers convolutional, Batch normalization and ReLU.
Dataset	CWRU	CWRU
Parameters(k)	36.6	29
FLOPS	1,760,000	1,030,000(normal) /1,138,000(fault)
Classified labels	10	10
Accuracy	98%	97.7%

TABLE XVI
ACCURACY OF HCNN IN DIFFERENT STAGES

Operation of each stage	Test accuracy
Build HCNN-tree model Root & Child1~3 (9 bits of the input image)	0.9770
Weight quantization to 10 bits by reduce bits	0.9628
Convert γ' in BN to the 9 bits fixed-point	0.9543
Convert β' in BN to the 9 bits fixed-point	0.9543
Verilog Register Transfer Level (RTL)	0.9531

in this study, we are only able to provide the results from our specific hardware implementation.

In order to minimize the resource utilization in hardware implementation, it is necessary to implement quantization and fixed-point operations, which could potentially lower the accuracy. As such, experiments are conducted to examine the impact of these hardware-specific operations on accuracy. The outcomes of these different stage operations are presented in Table XVI.

As shown in Table XVI, the accuracy of the proposed HCNN can reach 97.7% before hardware implementation. However, after each step in the hardware implementation process (such as quantization and fixed-point operations), the accuracy sees a marginal decrease, with the Verilog register transfer level (RTL) implementation achieving the lowest accuracy of 95.31%. This can be attributed to the fact that during the transition of each value, there will be instances where values exceed the bit width determined for that layer. Consequently, a small number of values will invariably contain errors due to the bit-width relationship across the five layers of convolution, BN, and max pooling. Hence, it is nearly impossible for the accuracy of the hardware implementation to match the values in the software stage exactly.

In addition, to validate the real-time computational capability of the proposed HCNN design, the computing cycles are used as a measure to estimate computation time. If HCNN model calculation can be executed within one data collection period without significant data point loss, the proposed HCNN design possesses real-time computational capability.

Therefore, the first step in validating the real-time capability involves determining the computing cycles of the proposed HCNN design. In the RTL simulation, the classification cycles of the proposed design are shown in Fig. 6. According to Fig. 6, the proposed HCNN requires a total of 330 000 cycles to identify faulty bearing conditions. Of these 330 000 cycles,

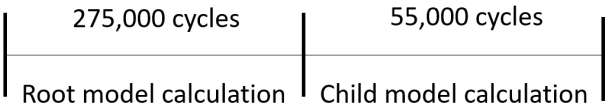

Fig. 6. Different situations RTL-level cycles count.

the root model calculation accounts for 275 000 cycles, while the child model accounts for the remaining 55 000. This implies that if the input data corresponds to a normal (healthy) condition, the classification cycles can be reduced to 275 000 rather than the full 330 000, as the results can be returned without executing the child model.

It is worth noting that in the analysis of real-time computing capability, the worst case scenario for computing time is considered, i.e., 330 000 cycles. To elaborate, in the proposed HCNN, all sensor data are collected at a rate of 12 000 samples per second, and each image includes 4096 data points. Given a clock cycle of 10 ns for the circuit and a requirement of 330 000 clock cycles for model inference in a circuit implemented with TSMC 40-nm CMOS process, the response time from receipt of 4096 sensor data points to the output of bearing health status can be determined, as shown in (1). It takes 3 300 000 ns (or 0.0033 s) to complete one classification. Moreover, the response time from receipt of 4096 sensor data points to the output of the result takes 344 633 196 ns, as shown in (2)

$$(330000 \times 10 \text{ ns}) = 3\,300\,000 \text{ ns} \quad (1)$$

$$(83333.3 \text{ ns} \times 4096) + 330000 \times 10 \text{ ns} = 344\,633\,196 \text{ ns}. \quad (2)$$

In the proposed HCNN, there may be instances where data signals received during the computational process of classification might be overlooked. A constant stream of vibration signals will be sent to the circuit during the classification process for real-time consideration. As such, an auxiliary buffer is needed to keep these vibration signals when the circuit has not yet finished its process. As indicated in the following equation, the proposed HCNN circuit would need to store an additional 40 points of vibration signals:

$$3\,300\,000 \div 83\,333.3 \text{ ns} = 40 \text{ points}. \quad (3)$$

From (1)–(3), if the operational speed of the proposed HCNN circuit falls below 100 MHz, both the circuit's operation time and the additional buffer memory requirements will increase. Therefore, if the circuit's running speed is too slow, the real-time performance of the proposed HCNN could be compromised.

Furthermore, the chip layout is shown in Fig. 7, where the locations of the SRAM, register files, ROM, and I/O pads are indicated. The total chip area is $980 \times 980\ \mu\text{m}^2$. The details of the implementation of the proposed HCNN hardware design are provided in Table XVII.

The power consumption of the proposed HCNN is 65.608 mW when operating at 100 MHz with a 0.9-V power supply in the worst case scenario, which requires the execution of the child model. Importantly, this power consumption result is achieved through the implementation of the power gating method, which effectively reduces power usage.

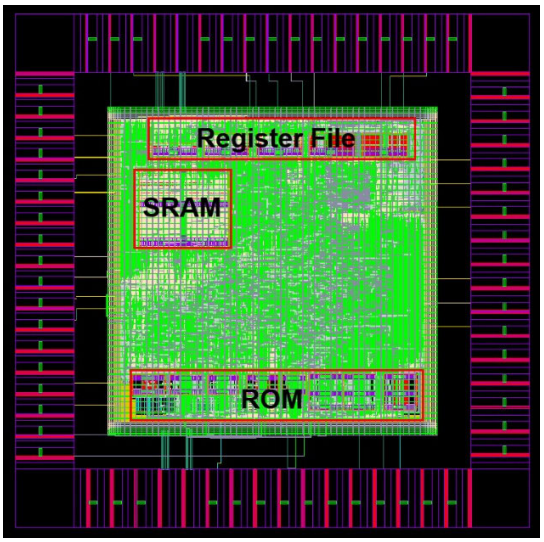

Fig. 7. Layout of the proposed HCNN circuit.

TABLE XVII
HCNN HARDWARE IMPLEMENTATION RESULTS

	The proposed architecture
Technology	TSMC 40nm CMOS
Frequency (MHz)	100
Power (mW)	65.608 (faults)
FLOPS (#)	1,138,000 (faults)
Chip area (mm^2)	0.9604
Parameters (k)	29
Accuracy	95.31%

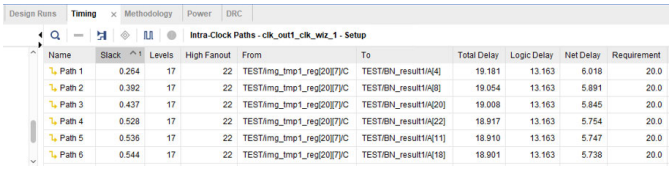

Fig. 8. Timing report of the proposed HCNN hardware implemented in a Virtex-7 FPGA.

To investigate the specific benefits brought about by the power gating method, we conducted simulations without implementing power gating, resulting in a power consumption of 70.3 mW. This comparison demonstrates that the power gating method successfully reduces power consumption by approximately 6.7%. Moreover, the hierarchical design of the HCNN model also contributes to power reduction when the input data are in an NC. In such cases, the model terminates at the root model without calculating the child model. When combined with the power gating method, the power consumption is further reduced to 52.296 mW when classifying NCs. In other words, the proposed design achieves even lower power consumption during the normal case.

Besides implementing the proposed HCNN using the TSMC 40-nm CMOS process and providing postlayout simulation results, this article also utilizes an FPGA evaluation board (Virtex-7 VC707) to prototype the HCNN circuit as a preliminary verification before chip tape-out. Fig. 8 shows the timing report generated by the FPGA synthesis tool. As illustrated in Fig. 8, the total delay on the critical path is less than 20 ns on the FPGA. Consequently, the proposed HCNN can operate correctly at 55 MHz on the FPGA. Furthermore, Fig. 9 depicts the power consumption of the HCNN on the FPGA. The proposed HCNN circuit, when implemented on the FPGA, consumes 0.533 W of power at 55 MHz.

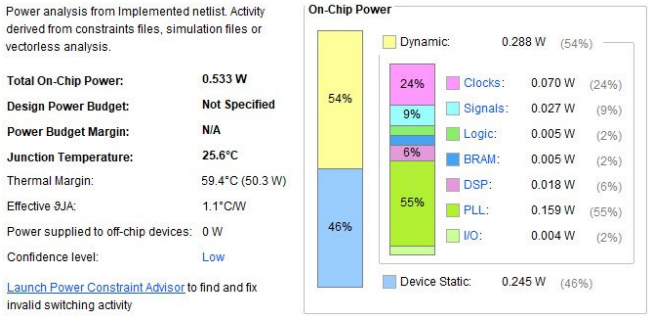

Fig. 9. Power consumption of the proposed HCNN on FPGA (at 55 MHz).

VI. CONCLUSION

In this article, an HCNN model for bearing fault analysis, specifically designed for CWRU datasets, is presented. Through the implementation of a two-step classification, the proposed HCNN model significantly reduces both memory usage and power consumption. Importantly, the proposed HCNN model achieves an impressive accuracy rate of 97.7% at the software stage, using only 29 K parameters.

When compared with other hardware implementations, the proposed HCNN stands out for its real-time computing capability. Specifically, in the worst case scenario, the proposed HCNN requires just 330 000 cycles to classify a bearing fault, with no loss of incoming sensor data points. Furthermore, this article also tests the effects of the white Gaussian noise and improves the noise resistance of the proposed HCNN.

As for the hardware implementation, the proposed HCNN is realized using TSMC 40-nm CMOS technology, incorporating power gating features. The maximum operating frequency of the proposed HCNN is 100 MHz. Under the 40-nm technology, the HCNN consumes 65.608 mW of power and achieves a classification accuracy rate of 95.31% in the final hardware test. In addition, the proposed hardware circuit was also deployed on a Virtex-7 FPGA to validate the feasibility of this architectural circuit on a real, hardware-based platform.

REFERENCES

[1] *Case Western Reserve University Bearing Data Center Website*. Accessed: Nov. 21, 2023. [Online]. Available: https://engineering.case.edu/bearingdatacenter

[2] C. Lessmeier, J. K. Kimotho, D. Zimmer, and W. Sextro, "Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification," in *Proc. Eur. Conf. Prognostics Health Manage. Soc.*, Jul. 2016, vol. 3, no. 1, pp. 1–17.

[3] E. Bechhoefer, "Condition based maintenance fault database for testing of diagnostics and prognostic algorithms," MFPT Fault Data Sets. Accessed: Nov. 21, 2023. [Online]. Available: https://www.mfpt.org/fault-data-sets

[4] Y. Jin, L. Hou, and Y. Chen, "A new rotating machinery fault diagnosis method based on the time series transformer," 2021, *arXiv:2108.12562*.

[5] D. A. Paolo, G. Luigi, F. Alessandro, and M. Stefano, "Performance of envelope demodulation for bearing damage detection on CWRU accelerometric data: Kurtogram and traditional indicators vs. targeted a posteriori band indicators," *Appl. Sci.*, vol. 11, no. 14, p. 6262, Jul. 2021.
[6] 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.
[7] L. Hou, R. Jiang, Y. Tan, and J. Zhang, "Input feature mappings-based deep residual networks for fault diagnosis of rolling element bearing with complicated dataset," *IEEE Access*, vol. 8, pp. 180967–180976, 2020.
[8] P. Sharma, S. Chandan, and B. P. Agrawal, "Vibration signal-based diagnosis of defect embedded in outer race of ball bearing using 1-D CNN," in *Proc. Int. Conf. Comput. Perform. Eval. (ComPE)*, Jul. 2020, pp. 531–536.
[9] G. Sun, Y. Gao, Y. Xu, and W. Feng, "Data-driven fault diagnosis method based on second-order time-reassigned multisynchrosqueezing transform and evenly mini-batch training," *IEEE Access*, vol. 8, pp. 120859–120869, 2020.
[10] D.-T. Hoang and H.-J. Kang, "A survey on deep learning based bearing fault diagnosis," *Neurocomputing*, vol. 335, pp. 327–335, Mar. 2019.
[11] S. Zhang, S. Zhang, B. Wang, and T. G. Habetler, "Deep learning algorithms for bearing fault diagnostics—A review," in *Proc. IEEE 12th Int. Symp. Diag. Electr. Mach., Power Electron. Drives (SDEMPED)*, Aug. 2019, pp. 257–263.
[12] Y. Shao, X. Yuan, C. Zhang, and C. Liu, "Rolling bearing fault diagnosis based on wavelet package transform and IPSO optimized SVM," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Aug. 2020, pp. 2758–2763.
[13] D. Lee, V. Siu, R. Cruz, and C. Yetman, "Convolutional neural net and bearing fault analysis," in *Proc. Int. Conf. Data Mining (DMIN)*, Jan. 2016, pp. 194–200.
[14] A. Bytyn, R. Leupers, and G. Ascheid, "An application-specific VLIW processor with vector instruction set for CNN acceleration," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.
[15] G. Xu, M. Liu, Z. Jiang, D. Söffker, and W. Shen, "Bearing fault diagnosis method based on deep convolutional neural network and random forest ensemble learning," *Sensors*, vol. 19, no. 5, p. 1088, Mar. 2019.
[16] A. Goel, S. Aghajanzadeh, C. Tung, S.-H. Chen, G. K. Thiruvathukal, and Y.-H. Lu, "Modular neural networks for low-power image classification on embedded devices," *ACM Trans. Design Autom. Electron. Syst.*, vol. 26, no. 1, pp. 1–35, Jan. 2021.
[17] D. Neupane, Y. Kim, and J. Seok, "Bearing fault detection using scalogram and switchable normalization-based CNN (SN-CNN)," *IEEE Access*, vol. 9, pp. 88151–88166, 2021.
[18] J. Zhang, Y. Zhou, B. Wang, and Z. Wu, "Bearing fault diagnosis base on multi-scale 2D-CNN model," in *Proc. 3rd Int. Conf. Mach. Learn., Big Data Bus. Intell. (MLBDBI)*, Dec. 2021, pp. 72–75.
[19] W. Zhang, D. Yang, H. Wang, X. Huang, and M. Gidlund, "CarNet: A dual correlation method for health perception of rotating machinery," *IEEE Sensors J.*, vol. 19, no. 16, pp. 7095–7106, Aug. 2019.
[20] J. Yang, J. Liu, J. Xie, C. Wang, and T. Ding, "Conditional GAN and 2-D CNN for bearing fault diagnosis with small samples," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–12, 2021.
[21] 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.
[22] R. Magar, L. Ghule, J. Li, Y. Zhao, and A. B. Farimani, "FaultNet: A deep convolutional neural network for bearing fault classification," *IEEE Access*, vol. 9, pp. 25189–25199, 2021.
[23] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[24] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "MCUNet: Tiny deep learning on IoT devices," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, Jul. 2020, pp. 11711–11722.
[25] N. F. Waziralilah, A. Abu, M. H. Lim, L. K. Quan, and A. Elfakharany, "A review on convolutional neural network in bearing fault diagnosis," in *Proc. Eng. Appl. Artif. Intell. Conf. (EAAIC)*, vol. 255, Jan. 2019, p. 06002.
[26] 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.
[27] Y. Zhang, Z. Jia, H. Du, R. Xue, Z. Shen, and Z. Shao, "A practical highly paralleled ReRAM-based DNN accelerator by reusing weight pattern repetitions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 4, pp. 922–935, Apr. 2022.
[28] M. J. Rasch et al., "Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators," *Nature Commun.*, vol. 14, no. 1, p. 5282, Aug. 2023.

Yu-Pei Liang (Member, IEEE) received the B.S. degree in computer science and information engineering from National Central University, Taoyuan, Taiwan, in 2015, and the M.S. and Ph.D. degrees from the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, in 2017 and 2020, respectively.

She is currently an Assistant Professor with National Chung Cheng University, Chiayi, Taiwan. Her current research focuses on improving the performance of general algorithms for the systems with nonvolatile memory and the storage systems with technologies of emerging storage devices.

Yao-Shun Hsu received the M.S. degree in computer science and information engineering from National Chung Cheng University, Chiayi, Taiwan, in 2022.

His current research interests include system-on-a-chip design methodologies and artificial intelligence (AI) chip design.

Ching-Che Chung (Senior Member, IEEE) received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1997 and 2003, respectively.

From 2004 to 2008, he was a Post-Doctoral Researcher with National Chiao Tung University, working in system-on-chip design methodologies and high-speed interface circuit design. In August 2008, he joined the Computer Science and Information Engineering Department, National Chung Cheng University, Chiayi, Taiwan, where he is currently a Professor. His research interests include wireless and wireline communication systems, low-power and system-on-a-chip design technology, mixed-signal IC design and sensor circuits design, artificial intelligence (AI) chip design, all-digital phase-locked loop, all-digital delay-locked loop, and its applications.