A Binary Weight Convolutional Neural Network Hardware Accelerator for Analysis Faults of the CNC Machinery on FPGA

Ching-Che Chung, *Senior Member*, *IEEE*, Yu-Pei Liang, *Member*, *IEEE*, Ya-Ching Chang, and Chen-Ming Chang
Department of Computer Science and Information Engineering
National Chung Cheng University
No. 168, University Rd., Min-Hsiung, Chia-Yi, Taiwan
Email: wildwolf@cs.ccu.edu.tw, ypliang@cs.ccu.edu.tw

2023 International VLSI Symposium on Technology, Systems and Applications (VLSI-TSA/VLSI-DAT) | 979-8-3503-3416-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/VLSI-TSA/VLSI-DAT57221.2023.10134316

***Abstract*—With the development of machine learning technologies and the internet of things, many traditional industrial problems can be solved by combining these two technologies. For example, the CNC machinery fault can be detected by collecting the data from vibration sensors and analysis with machine learning algorithms. However, previous studies only focused on the accuracy of the machine learning model but did not consider the overhead of hardware implementation. This paper proposes a real-time CNC machinery fault detection solution using a binary weight convolutional neural network to identify vibration signals. In addition, the operations in the designed neural network have been converted to fixed-point operations to speed up the calculations and reduce memory usage. Finally, the proposed method has been implemented on FPGA to evaluate the capabilities. According to the experimental result, the proposed method can achieve 95.07% accuracy while implemented on the FPGA with 130MHz clock frequency.**

Keywords—binary neural network (BNN), convolutional neural network (CNN), CNC vibration signal identification

I. INTRODUCTION

Recently, the internet of things, cloud computing technology, and artificial intelligence (AI) have been overgrowing; therefore, AI has been able to apply in different fields. Notably, it facilitates the industry's progress from traditional human controlling to AI automatic monitoring. When AI technology enters the traditional production line in the industrial field, it can transform traditional factories into smart factories and achieve higher production quality. For example, the machine's running status can be monitored in real-time, and mechanical damage can be detected. Once the damaged machinery can be repaired early, it can reduce maintenance costs and downtime and maintain the safety of operators.

On the other hand, among the various applications in the industry, CNC (computer numerical control) machinery has been one of the most commonly used machinery in the production line for decades. However, most traditional CNC machines use a closed control system that only provides a fixed user interface and requires a professional operator to control the machine. If a professional operator manually operates it, the health status of the CNC machine cannot be closely monitored. Also, judging visually and audibly in the early stage of mechanical failure is difficult. Therefore, many researchers attempt to implement AI solutions in such a real industry scene.

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST-111-2221-E-194-049-.

For example, the bearing fault of the CNC machinery is one of the most common issues of machine breakdown [1], and many research papers focus on such fault diagnosis of rolling bearings. Within this research, the data provided by Case Western Reserve University (CWRU) Bearing Data Center [2] has been an essential dataset in this field. For example, some researchers [3,4,5,6] made the continuously sampled vibration signals in the CWRU dataset into images to train deep neural networks and classify the data into several categories. In those researches, the existing classification method that has been adopted includes support vector machine (SVM), multi-layer perceptron (MLP), deep belief network (DBN), artificial neural network (ANN), convolutional neural network (CNN), and deep neural network (DNN). In addition, some literature discusses the pre-processing image methods for feature extraction, such as cyclic spectral coherence (CSCoh) [7], local binary pattern(LBP)[8], compound interpolation envelope (CIE) [9], ensemble empirical mode decomposition (EEMD)[10], and wavelet semi-soft threshold (WSST) [10].

According to the results from prior research [3,5,6], CNN can achieve the highest accuracy in the classification. However, previous research only focuses on software accuracy but might neglect hardware overhead. Therefore, this paper aims to implement a hardware-cost-aware CNN network to detect the bearing faults of CNC machinery and further carry out real-time monitoring and self-prevention of mechanical damage. For achieving real-time processing, complex data pre-processing procedures are not required, a software model implementation has been constructed to ensure accuracy, and the parameters in the network would be subsequently quantized to reduce the hardware overhead. After that, the hardware of the proposed network was implemented on the FPGA. Note that, to the best of our knowledge, there was no existing literature discussing the hardware implementation for detecting CNC machinery faults by AI solutions.

The rest of this paper is organized as follows: the proposed architecture is presented in Section II. Then, the detail of the hardware implementation will be revealed in Section III. In addition, the experimental results are shown in Session IV and followed by the conclusion in Section V.

II. THE PROPOSED CNN ARCHITECTURE

This paper aims to develop a hardware-cost-aware CNN hardware accelerator for CNC machinery fault detection. For cost consideration, only four integer bits and five decimal bits present the original vibrational sensor data from the CWRU

dataset. Moreover, the fan-end (FE) accelerometer data is selected. In the fault type, the inner, ball, and outer race faults are selected for classification. These faults are divided into three fault diameters of 0.007, 0.014, and 0.021 mils, respectively. Therefore, there are ten categories of bearing data. Also, 4,096 consecutive vibration signal data points are arranged from left to right and top to bottom and create a 64×64 image, as shown in Fig. 1. For the CWRU dataset, a total of 7,989 images were generated in this way. After shuffling all the images, 80% is used to train the CNN network model and 20% to test the accuracy of the network model. The number of train and test images on each label is shown in Table I.

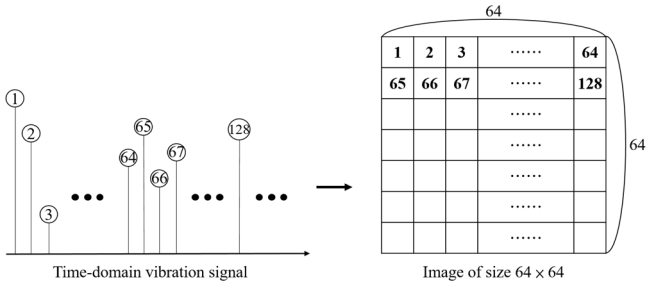

Fig. 1. Vibration signals convert to images.

Table I. The number of train and test images on each label.

Label	Fault Diameter	Type of Damage	# train images	# test images
0	0	Normal Baseline Data	423	108
1	0.007	Ball Fault	474	114
2	0.014	Ball Fault	506	111
3	0.021	Ball Fault	446	112
4	0.007	Inner Race Fault	509	137
5	0.014	Inner Race Fault	455	122
6	0.021	Inner Race Fault	609	157
7	0.007	Outer Race Fault	1390	342
8	0.014	Outer Race Fault	265	57
9	0.021	Outer Race Fault	1307	345

The proposed CNN network architecture for fault classification of CNC machinery is shown in Table II to overcome the difficulty of hardware implementation. The CNN network architecture includes zero-padding layers, convolution layers with a 3×3 kernel, batch normalization layers, ReLU activation layers, max-pooling layers, a global average pooling layer (GAP), and a fully connected layer (FC). The FC layer outputs the final classified label.

Initially, the input image uses a floating-point format for CNN architecture. Due to the hardware implementation, the input image must be converted from a floating-point format to a fixed-point format. A trade-off is made between the number of bits in each pixel and the model's accuracy to achieve the best result. Finally, it is determined that the pixel value of the input image will retain 9 bits, including 4-bit integer bits and 5-bit decimal bits. The accuracy is 99.18 %.

After completing the training of the CNN network model with Tensorflow, binary weight quantization is performed before the hardware implementation. The proposed binary CNN network (BNN) creates a lightweight CNN network for fault classification. In the binary CNN network, the weights are converted from floating-point numbers to +1 and -1, which can be expressed by 1 bit.

Table II. The input/output feature map of each layer.

Input (W×H×C)	Operator	Output (W×H×C)	stride
64×64×1	CONV	64×64×32	1
64×64×32	Max Pooling	32×32×32	2
32×32×32	CONV	32×32×64	1
32×32×64	Max Pooling	16×16×64	2
16×16×64	CONV	16×16×128	1
16×16×128	Max Pooling	8×8×128	2
8×8×128	CONV	8×8×128	1
8×8×128	Max Pooling	4×4×128	2
4×4×128	GAP	1×1×128	-
1×1×128	FC	1×1×10	-

Considering that the batch normalization layer has four parameters, this can take up too much memory space. Therefore, the parameters of the batch normalization layer are quantized using the K-mean clustering algorithm. However, the quantized value is still a floating-point number and must be converted to a fixed-point number, and then a lookup table can be created. Then, when the parameters of the CNN network have been quantized, the hardware implementation can be started, and the accuracy can be kept within an acceptable range.

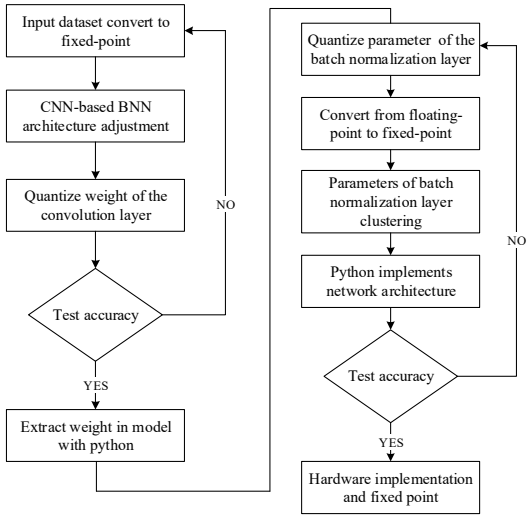

Fig. 2. Flowchart for CNN hardware implementation.

The flowchart from the software model to the CNN hardware accelerator implementation is shown in Fig. 2. The first step is to convert the vibration data of CWRU from a floating point to a fixed point. The second step is to input the fixed point obtained in the previous step to CNN-based BNN for training. The third step will be to adjust the structure and training for the weight of CNN-based BNN. Then it will test its accuracy. If the accuracy is unacceptable, it will go back to the first to the third steps to adjust the number of input bits and the network structure. When building and adapting the network architecture, the weights of the convolution layer are quantized at the same time.

The fourth step is to extract the weight of the network model. The fifth step quantizes the parameters in the batch normalization layer. The sixth step is to group the parameters in the batch normalization layer. The seventh step converts

the parameters obtained in the previous step from floating-point to fixed-point numbers and creates a lookup table. The eighth step will use python to implement network architecture for simulation accuracy of fixed-point computations. The integer digits of the feature maps remain unchanged, and the number of retained decimal bits gradually decreases. The network accuracy is tested until the accuracy is no longer acceptable. There is a trade-off between the number of bits and the accuracy of the network after many tests. If the accuracy is unacceptable, it will go back to the fifth to the eighth steps to adjust the number of input bits. Otherwise, the CNN hardware accelerator can be implemented.

III. Hardware Implementation

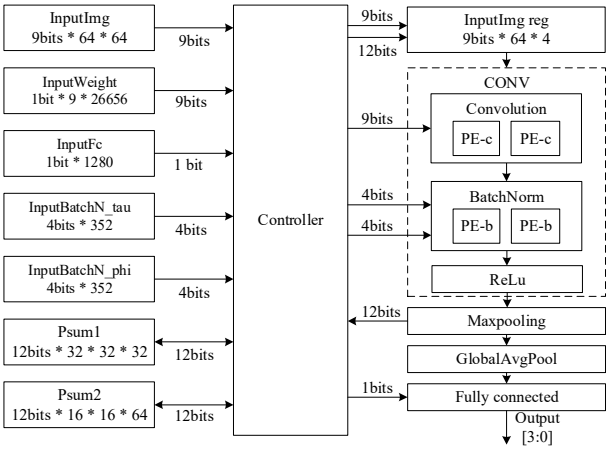

Fig. 3. The architecture of the proposed binary CNN hardware accelerator.

The architecture of the proposed binary CNN hardware accelerator is shown in Fig. 3. When training the proposed CNN network architecture, the weights of the convolution layers are quantized with +1 and -1. Moreover, after simplifying the formula, the parameters of the batch normalization layer are reduced to two parameters. Finally, these parameters are clustered and converted from floating to fixed-point numbers. In the hardware implementation of the neural network, the read-only memory (ROM) stores the weights extracted from the neural network model, including the weights of the convolution layers, the weights of the fully connected layers, and the parameters of the batch normalization layers.

All the above parameters are stored in ROM. The input vibrational sensor data for fault classification is stored in the random access memory (RAM) for testing the accuracy of the proposed binary CNN network. The output feature map of each layer is stored in RAM and used as the input feature map for the next layer. Two RAMs are used simultaneously to store the current layer's output feature map when inputting the previous layer's feature map.

Moreover, the number of parameters for each layer in the proposed binary CNN network is shown in Table III. The total parameters of the weights and batch normalization parameters are 242k. The quantized parameters can effectively reduce the number of bits for parameters and reduce the memory used in hardware implementation. Table IV compares the bits of the original parameter with the quantized parameter and calculates the percentage of memory reduction.

Table III. The number of parameters in the proposed binary CNN network.

Input channel	Operator	Output channel	Sum of parameters
1	Convolution (3x3)	32	288
32	Batch Normalization (2 parameters)	32	64
32	Convolution (3x3)	64	18,432
64	Batch Normalization (2 parameters)	64	128
64	Convolution (3x3)	128	73,728
128	Batch Normalization (2 parameters)	128	256
128	Convolution (3x3)	128	147,456
128	Batch Normalization (2 parameters)	128	256
128	GAP	128	0
128	Fully Connected	10	1280

Table IV. The memory usage reduction.

Memory	Memory name	Total bits without /with quantization	Reduction percentages
ROM	InputWeight	7,676,928/239,904	96.875%
	InputBatchN_phi	11,264/1,408	87.5%
	InputBatchN_tau	11,264/1,408	87.5%
	InputFC	40,960/1,280	96.875%
RAM	Psum1	1,048,576/393,216	62.5%
	Psum2	524,288/196,608	62.5%
Total bits		9,313,280/833,824	91.047%

This section discusses the hardware implementation issues to be considered, including network architecture depth, convolution layer weighting, batch normalization layer parameter quantization, and floating-point numbers to fixed-point number conversion. These network structures and quantization methods will affect the amount of memory used and the operation speed of the neural network computation.

IV. Experimental Results

Table V. Accuracy of each stage operation.

Stage operation	Accuracy
Build a CNN-based BNN model (9 bits of the input image)	0.9918
K-means clustering the parameters (floating-point) in the batch normalization layer	0.9894
φ' in the batch normalization layer convert from floating-point to 7 bits	0.9894
τ' in the batch normalization layer convert from floating-point to 11 bits	0.9744
RTL-level (Bit reduction in the calculation.)	0.9495
FPGA	0.9507

The bit width for adders and multipliers should be considered for hardware implementation. After addition and multiplication, the digits cannot grow indefinitely. Therefore, some decimal digits are discarded in the calculation without excessively affecting the accuracy. In addition, an FPGA evaluation board, Xilinx Virtex-7 FPGA VC707, is used to verify the accuracy of the hardware implementation. The use of memory includes ROM and RAM. The ROM stores the parameters that will not be changed, including the weights of the convolution layer and fully connected layer and the two parameters φ' and τ' of the batch normalization layer. RAM is used to store partial sum. Two RAMs will be used simultaneously to speed up the calculation. Two RAMs can store this partial sum while outputting the last partial sum.

In the experiment results, only one image is input into the memory at a time, with 4,096 vibration data points. The proposed FPGA circuit's operating frequency can run at 130MHz after implementation, and power consumption is 0.523 W. The accuracy is 95.07%.

Table VI. Comparison table with other software methods.

	[3] Sensors'17	[5] KBS'18	[7] MSSP'20	Proposed work
Architecture	CNN	CNN + HMMs	CNN	CNN + BNN
Dataset	CWRU	CWRU	CWRU	CWRU
# of trains	1000	800	1200	1390
# of tests	250	400	1200	342
Pre-processing method	Hanning window + RMS	Hankel matrix	CSCoh	reduce the bits
Image size	32×32	50×50	112×112	64×64
Classified labels	10	12	10	10
Accuracy	98.4%	98.08%	99.02%	99.18%

The different image sizes, pre-processing methods, and classification labels are compared in Table VI. The proposed pre-processing method is easier to implement and simpler than others. In addition, the proposed design achieves better accuracy than these software models.

Table VII. Comparison Table for CNN Hardware Accelerator

	[11] DATE'19	[12] Journal '19	[13] ITAIC'19	Proposed work
Technology	FPGA	FPGA	FPGA	FPGA
Platform	Zynq XC7Z020	Zynq XC702	ZYNQ7020	Virtex 7 VC707
Algorithm	CNN	CNN	CNN	CNN
Architecture	BNN	BNN	BNN	BNN
Dataset	**MNIST**	**Cifar10**	**MNIST**	**CWRU**
Parameters (k)	2,079	13,400	1,060	242
Frequency (MHz)	200	143	N/A	130
Power(mW)	500	3,300	4,000	523
Memory	N/A	1.7MB	1.07MB	Input 4.5KB Weight 29.79KB Output 72KB
LUT	7,954	29,629	39,824	7,317
FF	7,188	31,763	66,785	5,471
BRAMs	68	103	55.50	34.5
DSP	16	0	36	6
Accuracy	99.34	88.61%	99.49%	95.07%
GOPS	61.62	722	N/A	0.015158
GOPS/W	123	219	N/A	0.02898

Then the proposed CNN hardware accelerator is implemented on an FPGA evaluation board. Table VII shows the performance of different CNN networks implemented on FPGA. The proposed work and designs [11, 12, 13] both use BNN to quantize the weight to 1 and -1. However, the complexity of the CWRU dataset used in this work is not simpler than MNIST and Cifar10. Therefore, when the proposed binary CNN network is applied to the MNIST dataset, the classification accuracy is 99.22%. As a result, the proposed binary CNN network architecture can also be applied to the MNIST dataset. Besides, the number of parameters and memories is much smaller than [11, 12, 13].

An image has 4,096 vibration data points, and one iteration for the proposed CNN network requires 20,387,288 cycles, and the clock period of each cycle is 7.692 ns. Therefore, one iteration requires 156,819,019ns (0.156 seconds) in the proposed circuit on FPGA at 130MHz. Therefore, real-time can be achieved when the fastest vibration signal was collected at 26,119 samples/second.

V. CONCLUSION

This paper implements a CNN hardware accelerator network to detect the bearing faults of the CNC machinery. When constructing CNN-based BNN, the accuracy is 99.18%. After establishing the network model, the parameters of the batch normalization layer were quantized with an accuracy of 97.44%. Finally, the hardware is implemented in FPGA with an accuracy of 95.07%, and the total parameter is 242k.

REFERENCES

[1] Wade A. Smith and Robert B. Randall, "Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study," *Mechanical Systems and Signal Processing,* vol. 64-65, pp. 100-131, Apr. 2015.

[2] Case Western Reserve University Bearing Data Center Website ⟨http://csegroups.case.edu/bearingdatacenter/home⟩.

[3] Shaobo Li, Guokai Liu, Xianghong Tang, Jianguang Lu, and Jianjun Hu, "An ensemble deep convolutional neural network model with improved DS evidence fusion for bearing fault diagnosis," *Sensors,* vol. 17, no. 8, 1729, Jul. 2017.

[4] Hongmei Li, Jinying Huang, and Shuwei Ji, "Bearing fault diagnosis with a feature fusion method based on an ensemble convolutional neural network and deep neural network," *Sensors*, vol. 19, no. 9, 2034, Apr. 2019.

[5] Shuhui Wang, Jiawei Xiang, Yongteng Zhong, and Yuqing Zhou, "Convolutional neural network-based hidden Markov models for rolling element bearing fault identification," *Knowledge-Based Systems*, vol. 144, pp. 65-76, Mar. 2018.

[6] Changchang Che, Huawei Wang, Qiang Fu, and Xiaomei Ni, "Deep transfer learning for rolling bearing fault diagnosis under variable operating conditions," *Advances in Mechanical Engineering*, vol. 11, no. 12, pp. 1-11, Dec. 2019.

[7] Zhuyun Chen, Alexandre Mauricio, Weihua Li, and Konstantinos Gryllias, "A deep learning method for bearing fault diagnosis based on cyclic spectral coherence and convolutional neural networks," *Mechanical Systems and Signal Processing*, vol. 140, 106683, Jun. 2020.

[8] Kaplan Kaplan, Yılmaz Kaya, Melih Kuncan, Mehmet Recep Minaz, and H. Metin Ertunç, "An improved feature extraction method using texture analysis with LBP for bearing fault diagnosis," *Applied Soft Computing*, vol. 87, 106019, Feb. 2020.

[9] Xiang Li, Jun Ma, Xiaodong Wang, Jiande Wu, and Zhuorui Li, "An improved local mean decomposition method based on improved composite interpolation envelope and its application in bearing fault feature extraction," *ISA Transactions*, vol. 97, pp. 365-383, Feb. 2020.

[10] Jianghua Ge, Tianyu Niu, Di Xu, Guibin Yin, and Yaping Wang, "A rolling bearing fault diagnosis method based on EEMD-WSST signal reconstruction and multi-scale entropy," *Entropy*, vol. 22, no. 3, 290, Mar. 2020.

[11] Jeng-Hau Lin, Atieh Lotfi, Vahideh Akhlaghi, Zhuowen Tu, and Rajesh K. Gupta, "Accelerating Local Binary Pattern Networks with Software-Programmable FPGAs," *in Proceedings of 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2019.

[12] Peng Guo, Hong Ma, Ruizhi Chen, and Donglin Wang "A High-Efficiency FPGA-Based Accelerator for Binarized Neural Network," *Journal of Circuits, Systems, and Computers*, vol. 28, no. 1, 1940004, Jan. 2019.

[13] Yihan Yuan, Ruilian Zhao, and Songwei Pei, "Quantitative research of convolutional neural network and FPGA deployment," *in Proceedings of 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, May 2019.