A Rapid Design Process for Efficient CNN Hardware Accelerator on FPGA

Ching-Che Chung[a] **and Pin-Hua Chen**[a]

[a] Department of Computer Science and Information Engineering,
National Chung Cheng University, Taiwan
E-mail address: wildwolf@cs.ccu.edu.tw

Abstract

This paper presents a rapid design flow for efficiently determining quantization and computation methods in convolutional neural network (CNN) hardware accelerators. The proposed process provides eight quantization configurations to address various complexity and performance requirements. Additionally, an improved Integer-Arithmetic-Only (IAO) 8-bit quantization technique is introduced, which outperforms existing tools like TensorFlow Lite in both hardware feasibility and accuracy. For instance, the IAO method achieves 99.11% accuracy on the PU vibration dataset compared to 32.81% using TensorFlow Lite's full-integer quantization, and 76.60% versus 18.76% on the more complex PU current dataset. On simpler tasks like MNIST, IAO maintains about 99.5% accuracy, while TensorFlow Lite's full-integer approach drops to approximately 91%. Implemented on a VC707 FPGA board at 70 MHz, the proposed accelerator closely matches the predicted accuracy, with less than 0.5% accuracy loss, confirming the practicality and effectiveness of the design process.

Keywords: Convolutional Neural Network, Quantization, FPGA, Integer-Arithmetic-Only, Edge AI

1. Introduction

In recent years, the advances in artificial intelligence (AI) and the Internet of Things (IoT) have accelerated the development of deep learning. Among various deep learning models, Convolutional Neural Networks (CNNs) are well-known for achieving high accuracy in vision-related tasks. However, CNNs typically involve large numbers of parameters and extensive computations, posing challenges for direct deployment on resource-constrained edge devices. While cloud-based platforms efficiently handle such complexity, network bandwidth and latency constraints hinder real-time, on-device inference.

To overcome these challenges, parameter optimization and quantization techniques are employed before on-chip implementation. Previous works (Liang, et al., 2024; Chung, et al., 2023; Goyal, et al., 2021) explored fixed-point quantization to reduce parameter storage and computation complexity, but finding suitable integer and fractional bit-widths often requires exhaustive trial-and-error, especially when adapting to different datasets or CNN architectures. TensorFlow Lite (TensorFlow Lite website, 2025) offers built-in quantization options to expedite this process and can shrink model size without significant accuracy loss for simpler

tasks. However, it often struggles to maintain performance on more complex models.

This paper introduces a rapid design process that assists in determining suitable hardware computation methods for CNNs. This approach provides multiple quantization options and an improved 8-bit integer-arithmetic-only (IAO) quantization method that achieves higher accuracy without retraining. Additionally, an FPGA-based accelerator using 16-bit floating-point and 16-bit fixed-point formats validates the proposed strategies on both the MNIST dataset and a bearing dataset.

The main contributions in this paper are summarized as follows:

1. A rapid design process that quickly determines hardware computation methods for diverse CNN models, reducing manual optimization efforts.
2. An improved IAO 8-bit integer quantization that enhances accuracy for complex tasks without model retraining.
3. An FPGA-based accelerator implementing both 16-bit fixed-point and 16-bit floating-point formats to support different inputs, validated on MNIST and a bearing dataset.

The remainder of this paper is organized as follows: Section 2 introduces the proposed quantization process and compares its results with TensorFlow Lite. Section 3 describes the hardware implementation, Section 4 presents the experimental results, and Section 5 concludes the paper.

2. Proposed Rapid Design Process

This section describes the proposed quantization strategy and its validation using the MNIST dataset (The MNIST database of handwritten digits website, 2025) and the Paderborn University (PU) bearing dataset (Paderborn University Bearing Data Center website, 2025). Two CNN architectures are considered: a small 2-layer CNN for MNIST and a 5-layer 1-D CNN for the PU datasets. Both models apply convolution, batch normalization (BN), ReLU activation, and max-pooling. After quantizing parameters, each CNN is tested to evaluate accuracy under different formats.

Tables 1 and 2 present the input and output sizes for each layer of the proposed MNIST and PU models, respectively. The MNIST model processes a 28×28 image flattened to 1D, while the PU models use down-sampled signals of length 1,600. All data inputs are pre-processed into a fixed-point format (4-bit integer, 5-bit decimal).

TensorFlow Lite (TFLite) (TensorFlow Lite website, 2025) quantization results are shown in Table 3 (MNIST) and Table 4 (PU current). Dynamic range and Float16 quantization yield good accuracy but still rely on floating-point operations or additional scaling factors. Full integer quantization significantly reduces accuracy in complex tasks like the PU current dataset.

Table 1 Input and output data size in MNIST Model.

Operation	Input data size	Output data size	Stride
Convolution 1	$1 \times 784 \times 1$	$1 \times 784 \times 16$	1

Max pooling 1	1 × 784 × 16	1 × 196 × 16	4
Convolution 2	1 × 196 × 16	1 × 196 × 36	1
Max pooling 2	1 × 196 × 36	1 × 49 × 36	4
Fully connected	1×1×1,764 (after flattening)	1×1×10	-

Table 2 Input and output data size in PU Model.

Operation	Input data size	Output data size	Stride
Convolution 1	1×1,600×1	1×1,600×8	1
Max pooling 1	1×1,600×8	1×400×8	4
Convolution 2	1×400×8	1×400×16	1
Max pooling 2	1×400×16	1×100×16	4
Convolution 3	1×100×16	1×100×32	1
Max pooling 3	1×100×32	1×25×32	4
Convolution 4	1×25×32	1×25×32	1
Max pooling 4	1×25×32	1×7×32	4
Convolution 5	1×7×32	1×7×64	1
Max pooling 5	1×7×64	1×2×64	4
Fully connected	1×1×128 (after flattening)	1×1×4	-

Table 3 Results of MNIST model under Tensorflow lite.

Operation (parameter)	Original	Dynamic range	Float 16	Full integer
Convolution 1 (400)	FP32	FP32	FP16	Int8
Convolution 2 (14,400)	FP32	Int8	FP16	Int8
Fully connected (17,640)	FP32	Int8	FP16	Int8
Accuracy	96.15%	96.18%	96.15%	91.33%

To address these limitations, a rapid design process is introduced that provides eight quantization configurations, as listed in Table 5. The proposed design process, as illustrated in Fig. 1, starts with a TensorFlow-trained model—either the original or a retrained one with reduced parameter bit-width. After merging batch normalization parameters, each parameter is

quantized into one of four formats: floating-point (FP16, FP8, FP8+16), fixed-point (16-bit and 8-bit), integer (Int8 IAO), and Posit formats. BN parameters are compressed by merging mean and variance with γ and β, halving the parameter count.

Table 4 Results of PU model under Tensorflow lite.

Operation (parameter)	Original	Dynamic range	Float 16	Full integer
Convolution 1 (56)	FP32	FP32	FP16	Int8
Convolution 2 (896)	FP32	FP32	FP16	Int8
Convolution 3 (3,584)	FP32	Int8	FP16	Int8
Convolution 4 (7,168)	FP32	Int8	FP16	Int8
Convolution 5 (14,336)	FP32	Int8	FP16	Int8
Fully connected (512)	FP32	Int8	FP16	Int8
Accuracy	99.04 %	98.82%	99.03%	**18.76%**

Table 5 Quantization provided by the proposed design process

Numerical	Format of the storage
Floating-point	FP16, FP8, FP8+16
Fixed-point	Fixed-point16, Fixed-point8
Integer	Int8 (IAO)
Posit	Posit16, Posit8

FP8+16 quantization is suitable for models undergoing prior retraining to reduce parameter bit-width, while fixed-point formats maintain good performance at 16 bits but degrade significantly at 8 bits in complex tasks. To address this issue, an improved Int8 IAO quantization method is introduced. This method applies a two-stage quantization—from floating-point to 16-bit fixed-point, then to 8-bit integers—and stores scaling factors in fixed-point. Tables 6–10 present various quantization results, demonstrating that the Int8 IAO approach significantly improves accuracy over standard 8-bit methods, especially for challenging datasets like the PU vibration signals.

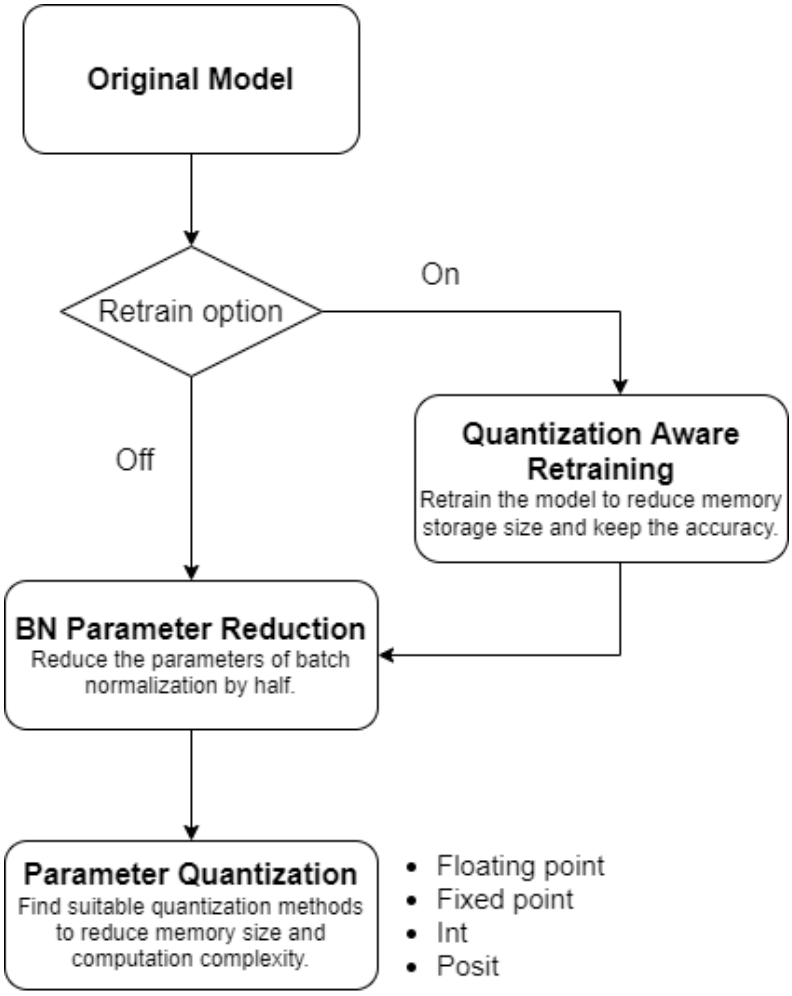

Figure 1 The flowchart of the proposed design process.

Table 6 Comparison of FP8(1,3,4) and FP8(1,4,3) in PU current model.

	Original	**FP8 (1,3,4)**	**FP8 (1,4,3)**
Weight	FP32	FP8	FP8
γ of BN	FP32	FP8	FP8
β of BN	FP32	FP8	FP8
Test accuracy	98.78%	**25.25%**	**49.16%**

Table 7 Results of using FP8+16 quantization in PU current model.

	FP16	**FP8+16**
Weight (DoReFa-Net)	Int5	Int5
Weight (lookup table)	FP16	FP8
γ of BN	FP16	FP16
β of BN	FP16	FP16
Test accuracy	97.52%	95.08%

Table 8 Results of using fixed-point quantization in PU current model.

	FP16	Fixed-point16		Fixed-point8	
		Integer bits	Decimal bits	Integer bits	Decimal bits
Weight	FP16	5	11	3	5
γ of BN	FP16	5	11	5	3
β of BN	FP16	5	11	5	3
Test accuracy	98.78%	93.08%		**20.66%**	

Table 9 Results of using improved IAO INT8 quantization in PU vibration model.

	Fixed-point8	Int8 (IAO)			TensorFlow Lite Full integer
		Quantized number	Zero point	Scale	
Weight	3+5	Int8	Int8	0+16	Int8
γ of BN	5+3	Int8	Int8	0+16	Int8
β of BN	5+3	Int8	Int8	0+16	Int8
Test accuracy	**38.44%**	99.11%			**32.81%**

Table 10 Results of PU vibration model using proposed design process.

	Original	FP16	FP8	Fixed 16	Fixed8	Int8 (IAO)	Posit 16	Posit8
Input data	4+5	4+5	4+5	4+5	4+5	4+5	4+5	4+5
Weight	FP32	FP16	FP8	5+11	3+5	Int8	Posit 16	Posit8
γ of BN	FP32	FP16	FP8	5+11	3+5	Int8	Posit 16	Posit8
β of BN	FP32	FP16	FP8	5+11	3+5	Int8	Posit 16	Posit8
Activation	FP32	FP16	FP16	5+11	5+11	5+11	FP16	FP16
Accuracy	99.45%	99.32%	98.09%	99.27%	38.44%	**99.11%**	99.41%	99.11%

Unlike standard 8-bit quantization, which often suffers from severe accuracy loss, the proposed IAO approach carefully preserves scale and zero-point information. Convolution weights and BN parameters are initially quantized to 16-bit fixed-point and then compressed to 8 bits. During inference, they are efficiently dequantized back to 16-bit fixed-point using the stored scale and zero point. This process avoids retraining and achieves near-original precision. Moreover, the memory footprint is drastically reduced. For example, in the PU current signal model's first convolutional layer, the storage requirement drops from 1,792 bits (FP32) to just 472 bits with IAO Int8 quantization. This balance of lower memory usage and higher accuracy indicates that the improved IAO quantization is well-suited for complex recognition tasks

where standard 8-bit quantization methods fall short.

Comparisons of the proposed quantization methods and TFLite for MNIST, PU vibration, and PU current signals are summarized in Tables 11–13. The proposed IAO quantization maintains higher accuracy than TFLite's full integer approach, demonstrating improved feasibility for on-chip deployment.

Table 11 Comparison of the proposed flow with Tensorflow lite using MNIST model.

	Design process (Original 99.51%)				**TensorFlow Lite (Original 96.15%)**		
	FP 16	**Fixed 16**	**Fixed 8**	**Int8 (IAO)**	**Float 16**	**Dynamic range**	**Full integer**
Weight	FP16	5+11	3+5	Int8	FP16	Int8	Int8
γ of BN	FP16	5+11	5+3	Int8	FP32	FP32	Int8
β of BN	FP16	5+11	5+3	Int8	FP32	FP32	Int8
Cal. process	Float	Int	Int	Int	Float	Float	Int
Accuracy	99.45	99.48	99.45	99.48	96.15	96.18	91.33

Table 12 Comparison of the proposed flow with Tensorflow lite using PU vibration model.

	Design process (Original 99.45%)				**TensorFlow Lite (Original 98.81%)**		
	FP 16	**Fixed 16**	**Fixed 8**	**Int8 (IAO)**	**Float 16**	**Dynamic range**	**Full integer**
Weight	FP16	5+11	3+5	Int8	FP16	Int8	Int8
γ of BN	FP16	5+11	5+3	Int8	FP32	FP32	Int8
β of BN	FP16	5+11	5+3	Int8	FP32	FP32	Int8
Cal. process	Float	Int	Int	Int	Float	Float	Int
Accuracy	99.32	99.27	**38.44**	99.11	98.81	98.78	**32.81**

Table 13 Comparison of the proposed flow with Tensorflow lite using PU current model.

	Design process (Original 99.01%)				**TensorFlow Lite (Original 99.04%)**		
	FP 16	**Fixed 16**	**Fixed 8**	**Int8 (IAO)**	**Float 16**	**Dynamic range**	**Full integer**
Weight	FP16	5+11	3+5	Int8	FP16	Int8	Int8
γ of BN	FP16	5+11	5+3	Int8	FP32	FP32	Int8
β of BN	FP16	5+11	5+3	Int8	FP32	FP32	Int8

Cal. process	Float	Int	Int	Int	Float	Float	Int
Accuracy	98.78	93.08	**20.66**	**76.60**	99.03	98.82	**18.76**

3. Hardware Implementation

The proposed hardware design supports flexible CNN configurations on an FPGA, accommodating varying input sizes, network depths, and channel counts. Both FP16 and Fixed16 implementations are considered, reflecting the quantization options identified by the rapid design process. Fig. 2 illustrates the overall architecture, where all layers, including convolutions and fully connected layers, operate with consistent precision. The design allows adjustments of parameters such as Img_size, Layer_num, Out_channel, and Label_num to accommodate different models.

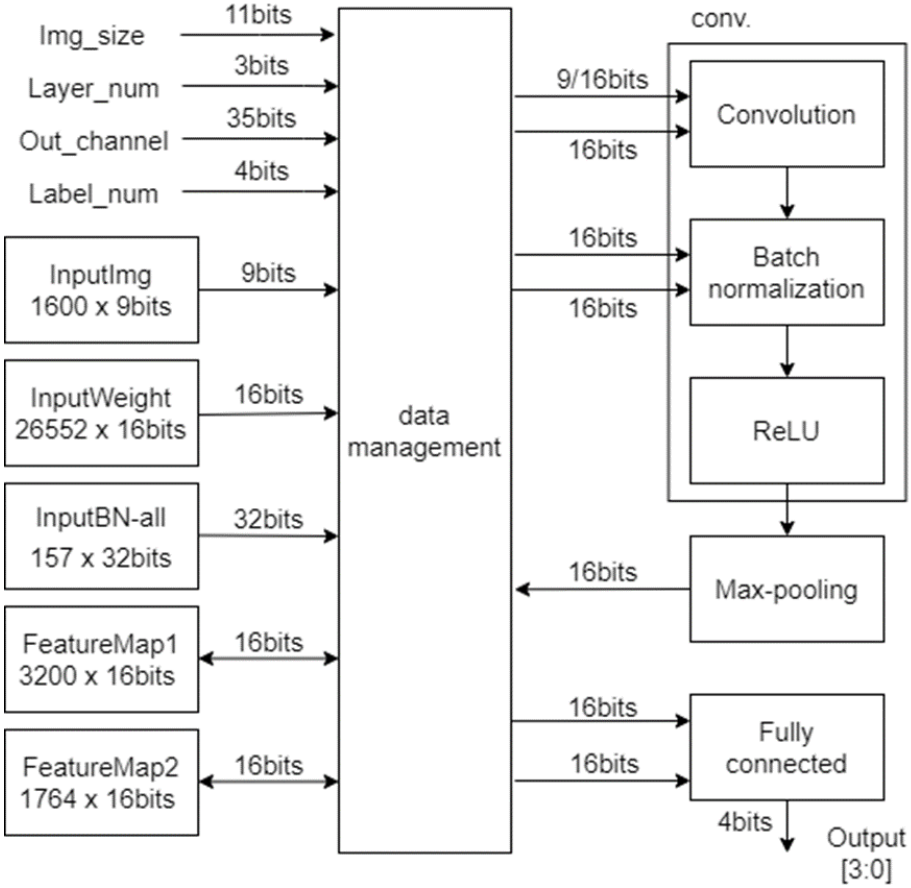

Figure 2 Overall architecture of the proposed CNN hardware design.

All input data, model weights, and BN parameters are stored in dedicated on-chip memory. The BN parameters are preprocessed to merge mean and variance with γ and β, reducing the required computations. Intermediate feature maps are stored in feature map memories for reuse by subsequent layers. Hardware-based verification shows minimal accuracy loss compared to the software-level results of the proposed design flow.

4. Experimental Results

The proposed CNN accelerator was implemented on an AMD Virtex-7 VC707 FPGA

evaluation board. Fig. 3 shows that the maximum operating frequency without timing violations is 70 MHz. At this frequency, Fig. 4 illustrates the on-chip power consumption when processing one sample of the PU current signal, measuring approximately 753 mW. Increasing the frequency beyond 70 MHz introduces timing issues and raises power consumption.

Intra-Clock Paths - clk_out1_clk_wiz_0 - Setup

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source C
Path 1	0.051	18	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[148][46]/D	14.227	6.016	8.211	14.3	clk_out1
Path 2	0.057	18	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[394][46]/D	14.304	6.016	8.288	14.3	clk_out1
Path 3	0.064	17	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[166][44]/D	14.225	5.898	8.327	14.3	clk_out1
Path 4	0.085	18	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[249][46]/D	14.270	6.016	8.254	14.3	clk_out1
Path 5	0.086	17	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[143][44]/D	14.203	5.898	8.305	14.3	clk_out1
Path 6	0.092	18	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[264][47]/D	14.263	5.857	8.406	14.3	clk_out1
Path 7	0.093	18	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[119][46]/D	14.239	6.016	8.223	14.3	clk_out1
Path 8	0.101	17	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[96][44]/D	14.189	5.898	8.291	14.3	clk_out1
Path 9	0.112	17	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[189][44]/D	14.177	5.898	8.279	14.3	clk_out1
Path 10	0.112	18	10051	test1/bn2_img_reg[0]/C	test1/fmap2_tmp_reg[63][46]/D	14.239	6.016	8.223	14.3	clk_out1

Figure 3 The timing report of the proposed design running at 70MHz.

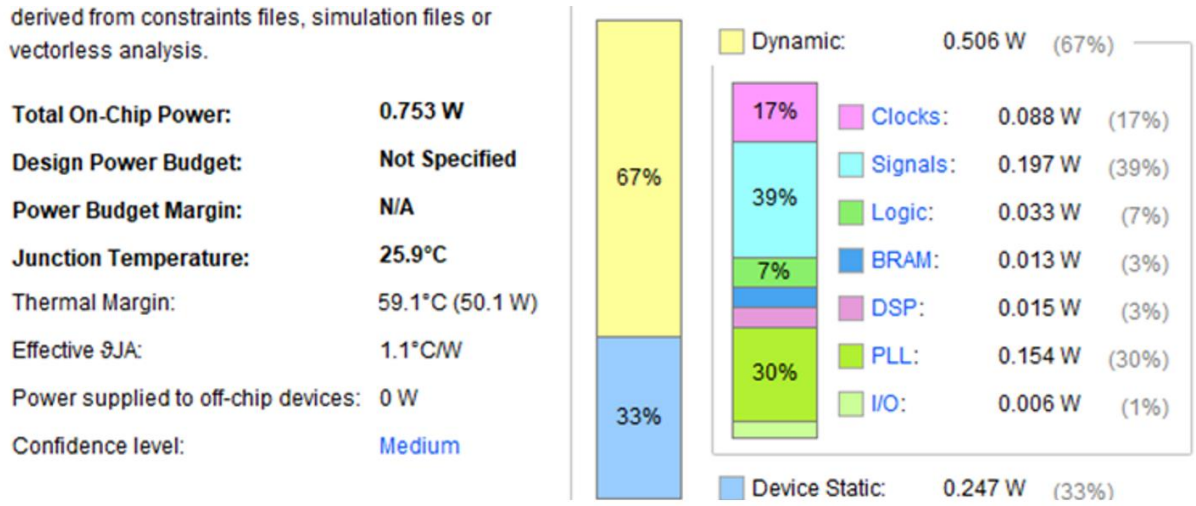

Figure 4 The on-chip power analysis on FPGA at 70MHz.

In hardware verification, both MNIST and PU datasets were tested using a subset of the data. The FPGA predictions match closely with the results obtained from the proposed design process, with less than 0.5% accuracy degradation. This confirms that the hardware implementation is consistent with the intended quantization strategies.

In terms of memory efficiency, the final quantization strategy reduces parameter storage to as little as one-quarter of the original model size. Moreover, due to careful memory block planning and parameter compression, the FPGA implementation utilizes under 2% of the available BRAM and approximately 25% of the available LUT resources. These results confirm that the selected quantization strategies are highly scalable and can accommodate larger network models or additional layers as needed.

5. Conclusion

This paper presented a rapid design process for determining suitable quantization and

computation methods in CNN hardware accelerators. The proposed approach offers eight quantization configurations, including an improved Int8 IAO method that significantly enhances accuracy without requiring model retraining. Experimental results demonstrated that the Int8 IAO method consistently outperforms TensorFlow Lite's full-integer quantization. For instance, in the PU vibration dataset, Int8 IAO achieved 99.11% accuracy, compared to only 32.81% using TensorFlow Lite's full-integer approach. In the more challenging PU current dataset, Int8 IAO reached 76.60% accuracy, surpassing the 18.76% result of TensorFlow Lite's full-integer quantization. Even for the simpler MNIST dataset, Int8 IAO sustained about 99.5% accuracy, whereas the TensorFlow Lite full-integer quantization dropped to approximately 91%.

Beyond these accuracy improvements, the proposed design process supports a broad spectrum of quantization formats—including floating-point, fixed-point, integer, and Posit—offering greater flexibility and feasibility for diverse hardware platforms. When implemented on an AMD Virtex-7 FPGA at 70 MHz, the accelerator closely matched the predicted accuracy of the proposed design process, exhibiting less than a 0.5% deviation for both MNIST and the PU datasets. The results confirm that the proposed process not only streamlines the selection of quantization strategies but also ensures efficient, high-accuracy deployment of CNN inference at the network edge.

6. References

Liang, Y.-P., Chang, H.-H., & Chung, C.-C. (2024). A low-power convolutional neural network implemented in 40-nm CMOS technology for bearing fault diagnosis. Proceedings of International VLSI Symposium on Technology, Systems and Applications (VLSI TSA), April 2024.

Chung, C.-C., Liang, Y.-P., & Jiang, H.-J. (2023). CNN hardware accelerator for real-time bearing fault diagnosis. Sensors, 23(13), 5897. doi:10.3390/s23135897

Goyal, R., Vanschoren, J., Van Acht, V., & Nijssen, S. (2021). Fixed-point quantization of convolutional neural networks for quantized inference on embedded platforms, arXiv:2102.02147v1 [cs.CV]. arXiv, February 2021.

TensorFlow Lite website (2025). Retrieved from https://www.tensorflow.org/lite/guide (accessed on 13 March 2025).

The MNIST database of handwritten digits website (2025). Retrieved from http://yann.lecun.com/exdb/mnist/ (accessed on 13 March 2025).

Paderborn University Bearing Data Center website (2025). Retrieved from https://mb.uni-paderborn.de/en/kat/research/kat-datacenter/bearing-datacenter/data-sets-and-download (accessed on 13 March 2025).