Debris Pose Estimation by Deep Learning on FPGA

Shintaro Hashimoto
Research and Development Directorate
Japan Aerospace Exploration Agency
2-1-1, Sengen, Tsukuba,
Ibaraki, 305-8505, Japan
hashimoto.shintaro@jaxa.jp

Naoki Ishihama
Research and Development Directorate
Japan Aerospace Exploration Agency
2-1-1, Sengen, Tsukuba,
Ibaraki, 305-8505, Japan
ishihama.naoki@jaxa.jp

***Abstract*— Although CNNs for regression problems are rarely implemented with FPGAs, our research installed debris pose estimation on an FPGA using the latest edge technology such as quantization neural network. Pose estimations were run on a workstation using 32bit floating-point precision and on an FPGA using 8bit int precision. The average median errors were 4.98% and 5.38%, respectively. This demonstrates that the regression problem can be transferred to an FPGA without a significant loss of accuracy. The FPGA power efficiency is more than 218k times that of a workstation implementation.**

Keywords—FPGA, Deep Learning, Pose Estimation, Debris

I. INTRODUCTION

Deep learning has evolved dramatically in recent years. In particular, object detection using convolutional neural networks (CNNs) is beginning to be used with various tasks such as surveillance camera. Running state-of-the-art CNNs requires considerable machine resources and power because it must calc matrix computation of millions of parameters consisting of floating points. It is difficult to implement a CNN for edge processing in satellites, automobiles, and more, where machine resources and power are limited. FPGAs meet such constraints of machine resources and power associated with CNNs. Unlike GPUs, FPGAs have limited resources (e.g., processing speed and memory) and are ineffective in handling floating-point operations. FPGAs do provide superior performance in integer and low precision arithmetic. Many research on edge processing has applied binary neural networks (BNNs), which are CNNs with binarized parameters. This greatly reduces the number of CNN parameters and allows an entire CNN model to be kept on-chip in FPGAs [1].

However, CNN inference by converting floating-point numbers to integers and low precision reduces classification accuracy. In the classification problem, even if the Logits values change due to the adverse effects of a BNN, the estimation result is robust if the ratio of output values is not changed significantly by a softmax layer. However, there is no softmax layer in models for regression problems, and the output values of the logits are used directly, so the inference results are incorrect. In addition, convolution by binary parameters cannot effectively use the brightness values of pixels, so it is difficult to obtain detailed features.

Many CNNs implemented on FPGAs were used only for simple classification tasks. Recently, a method has been proposed to enable accurate estimation by making the parameters of a CNN multi-valued. This is a quantization neural network (QNN). QNNs have fewer parameters (bit depth) than CNNs and better estimation accuracy than BNNs.

We focused on this QNN technique and attempted to install pose estimation of space debris into an FPGA. Pose estimation is a regression problem whose output consists of six values. We have already reported on our past research into pose estimation [2]. Debris removal satellites must know their pose at all times to remove debris. A highly accurate real-time inference is required to prevent collisions between the removal satellite and the target debris. However, satellites cannot use high-performance CPUs or GPUs due to power and heat constraints. Pose estimation with high accuracy was impossible with conventional BNNs on FPGAs. Even a combination of BNNs and QNNs did not work well. Even if quantization was used in the input and output layers, no inference could be made if an intermediate layer was binarized. This is thought to be because the information reduction by binarization is too large. For this reason, all the layers need to be quantized.

To our knowledge, no experiments on debris pose estimation on FPGA in satellites have been conducted. Our contribution demonstrates the high accuracy and processing speed that can be achieved by quantizing pose estimation on an FPGA. Note that the development cost of FPGAs is high, and there is concern about using FPGAs as the main processing unit. New FPGA code had to be developed every time the CNN model was changed for high-level synthesis. However, the development cost has been significantly lowered by using Xilinx libraries and IP core.

II. IMPLEMENTATION

A. Development Environment

Our development environment is shown in TABLE I. We have adopted Ultra 96 v2 as the SoC FPGA. The CLB LUT of ZU3EG was 70,560. Triply redundant circuits are used in satellites to prevent hardware processing errors caused by the single event setup by space radiation. The XQRKU060 has a proven history as an FPGA used in space, and its CLB LUT (e.g., 331k) is more than three times that of ZU3EG. If it works with ZU3EG, it will work with FPGAs in space.

TABLE I. DEVELOPMENT ENVIRONMENT

FPGA	SoC	Ultra 96 v2
	Chipset	Zynq UltraScale+ MPSoC ZU3EG
	CLB LUT	70,560
Workstation	OS	Ubuntu 18.04
	GPU	RTX3090
	CPU	i9-7900X
	Memory	64GB

B. Development Procedure

We used TensorFlow 1.x and 2.x for CNN training and evaluation and Vitis-AI 1.4x for FPGA installation. The development procedure is shown in Figure 1. The processes that reduce FPGA performance included in CNN step (a), such as matrix decomposition, normalization, and Bayesian inference, were excluded from step (b). Step (c) trains and evaluates CNN. Step (d) quantizes, calibrates and evaluates CNN model of step (c) to achieve highly accurate estimation, even if low precision is used. Generally, evaluation in step (d)

is the expected accuracy index for an FPGA. step (e) compiles the CNN model into an FPGA instruction set. The FPGA is processed to accelerate matrix operations in CNNs.

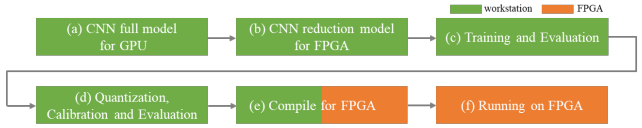

Figure 1. Procedure for installing FPGA

Figure 2 shows the final CNN model for the FPGA. The model on the FPGA is shown as white boxes.

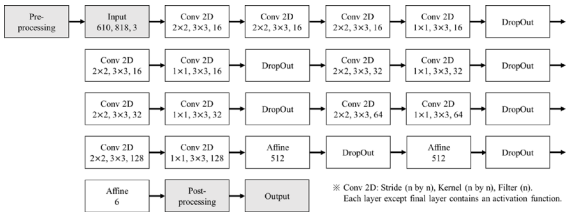

Figure 2. CNN model: Gray boxes represent CPU processing.

III. Evaluation

We evaluated the system with SoC in TABLE I. The input image is pre-processed by CPU in SoC such as resizing, normalization, and data transfer, and then the CNN processing is started by the FPGA. The FPGA processing result was acquired by the CPU and post-processed such as normalization to obtain the final output result. We evaluated with running all of these flows. The input images are shown in Figure 3. The original image was 1636(w) x 1220(h) pixels and resized in half by pre-processing.

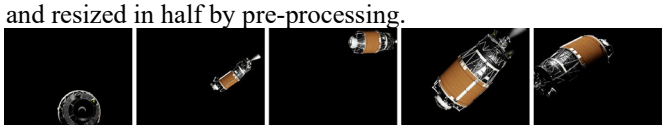

Figure 3. Examples of test images.

TABLE II shows a comparison of accuracy at each precision in step (d) in Figure 1. There are 400 test images. x, y, z in design space are 28 [m], 20.8 [m], and 40 [m] respectively. n_x, n_y, n_z is pose parameters with no rotation around the axis. The error rate in this research is the median error divided by the design space. The median value was used because it was difficult to estimate the pose of some of the test images. All results were run on the workstation. The power consumption of the system that included GPU, CPU and more was at least 250 [W]. The 400 images took 28.21 [s] for inference with no pre- and post-processing. There were no differences between the float 32bit and int 16bit error rate. Although the difference error rate between the int 16bit and 8bit is small, the amount of parameter compression was large. Compared to the int 8bit calculation, the accuracy of 4bit and 2bit calculations is significantly lower. We adopted the int 8bit parameters because of its accuracy and reduction rate.

TABLE II. Comparison of Pose Estimation Error Rate at The Depth of Quantization

Precision	x [%]	y [%]	z [%]	nx [%]	ny [%]	nz [%]	Err.
float 32bit	2.5	3.1	9.7	4.3	4.2	6.1	351.7
int 16bit	2.5	3.1	9.7	4.3	4.2	6.1	351.8
int 8bit	3.3	4.1	10.3	4.4	4.0	5.8	371.9
int 4bit	15.7	16.0	25.4	25.7	20.4	27.9	1200.1
int 2bit	16.7	16.8	26.6	28.3	19.6	49.4	1278.3

TABLE III shows the detailed accuracy when the int 8bit CNN is operated on an FPGA. There was no significant difference between operating the CNN on an FPGA and a workstation. The total error was 371.2, which was better than the workstation. However, this is due to differences in calculation processing such as rounding error and is not significant. From this, we were found that step (a) – (d) in Figure 1 should ran repeatedly, and when a highly accurate model is completed, its model should be placed on the FPGA.

TABLE III. The Detailed Accuracy by int 8bit CNN on FPGA

	x	y	z	n_x	n_y	n_z
Error rate	3.3 [%]	4.1 [%]	10.6 [%]	4.4 [%]	4.0 [%]	5.9 [%]
Ave. err.	1.28 [m]	1.04 [m]	4.94 [m]	0.143 [-]	0.138 [-]	0.209 [-]
Med. err.	0.93 [m]	0.86 [m]	4.24 [m]	0.088 [-]	0.079 [-]	0.117 [-]

The power consumption for an int 8bit CNN on an FPGA is shown in TABLE IV. Ultra96 v2 was run from a regulated power supply, and the power consumption was measured. The power consumption of the FPGA during CNN operation was estimated by comparing it with the power consumption in the steady state. Considering overhead, this power consumption is the final amount required for the system to operate. The CPU operated the OS and file system while the FPGA was running. TABLE IV indicates peak power in processing 400 test images. The peak power required for this system to operate was 6.084 [W]. Power efficiency is 12.4k [images/W] and more than 218k times that of a workstation implementation. The processing speed of FPGA was about 5.3k times faster than that of GPU.

TABLE IV. Power Consumption of FPGA

State	Wattage	Ampere	Voltage
(α) Standby power: No processing	5.796	0.483	12.00
(β) CPU-processing: Pre-processing	α+0.264	0.505	12.00
(γ) FPGA-processing: CNN-processing	α+0.288	0.507	12.00

The processing speed of the FPGA is shown in TABLE V. Items (α) and (β) in TABLE V were measured at the same timing as items (β) and (γ) in TABLE IV, respectively. Since item (γ) is a light process, such as normalization, there was little change in the power. Item (β-1) is the result of processing each image individually. Item (β-2) is batch processing and operated the FPGA continuously.

TABLE V. Processing-Speed of System

State	Chipset	Average time	Average deviation
(α) Pre-processing	CPU	178.08 [ms]	9.40 [ms]
(β-1) FPGA (Sequential)	FPGA	71.04 [μs]	3.94 [μs]
(β-2) FPGA (Continuous)	FPGA	13.25 [μs]	7.75 [μs]
(γ) Post-processing	CPU	8.79 [ms]	0.05 [ms]

IV. Conclusion

The same pose estimation was done with float 32bit on the workstation and int 8bit parameters on the FPGA, with average median errors of 4.98% and 5.38%, respectively. The estimation accuracy of the regression problem did not decrease significantly with 8bit quantization. Since the original accuracy of CNN was lower than that of the 8bit scale, the decrease in accuracy due to quantization was at a level that was not a problem. The power efficiency of an FPGA is higher than that of a GPU, so FPGAs are suitable for use in satellites.

References

[1] Yaman Umuroglu, et al., "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," The 2017 ACM/SIGDA International Symposium on FPGA, 2017.

[2] Shintaro Hashimoto, et al., "6-DoF Pose Estimation for Axisymmetric Objects Using Deep Learning with Uncertainty," 2020 IEEE Aerospace Conference, 2020.