Demonstration of Object Detection for Event-driven Cameras on FPGAs and GPUs

Masayuki Shimoda
Tokyo Institute of Technology, Japan
Email:shimoda@reconf.ict.e.titech.ac.jp

Shimpei Sato
Tokyo Institute of Technology, Japan
Email: satos@ict.e.titech.ac.jp

Hiroki Nakahara
Tokyo Institute of Technology, Japan
Email: nakahara@ict.e.titech.ac.jp

I. Abstract

We demonstrate an object detection system using a sliding window method for an event-driven camera[3] which outputs a subtracted frame (usually a binary value) when changes are detected in captured images. Fig. 1 shows the overall architecture of the object detector using a sliding window. First, our system extracts the object region candidates by applying a sliding window to the picture output by an event-driven camera. Since the event-driven camera outputs are binary, the sliding window determines whether an object exists or not based on the proportion of white pixels inside the bounding box as shown in Fig. 2. It reduces the number of proposed regions from 1,485 to average 10. The extracted pictures are resized to 40×40 size and applied to the ABCNN (All Binarized Convolutional Neural Network[7]), which is a BCNN (Binarized Convolutional Neural Network[4]) in which the first convolutional layer is done in binary form as well. All binarization is found to improve both the area and computation time, while the classification accuracy decreases. If the ABCNN infers the detected object as *human*, then it draws a box on the proposed region. Since more than one bounding box is commonly drawn against a single detected object, non-maximum suppression is used to reduce the extras to a single box. The proposed system is implemented on an FPGA and a GPU to show the comparison of them.

II. Event-Driven Camera

An event-driven camera is a camera that only outputs a picture when the captured image changes. Such cameras work by extracting contrast differences between two successive frames. If no object motions are detected, then the camera does not output anything. Once motions are observed, the camera subtracts the background with the reference image stored in pixel memory, converts the input into binary form by using a preset threshold, and then outputs the binary images. Use of an event-based sensor leads to a reduction in input/output (IO) energy when compared with a frame-based camera. One of the more practical use scenarios of an event-driven camera is security monitoring.

III. Sliding window

One of the commonly used object region methods is called the sliding window. A bounding box moves along the *dx* and *dy* axes from top left to down right on the picture to identify object region candidates. Since this scan is typically performed on RGB three-channel pictures, all regions are normally selected. However, when an event-driven camera is used, the picture is output in binary form and candidate regions are only proposed when the ratio of white pixels exceeds a certain threshold. This leads to a significant reduction in CNN computation requirements. In the case of the pedestrian dataset used in this paper, the number of RGB picture candidates is normally in the thousands, while a binary bitmap picture is used, the number drops to between zero and 20.

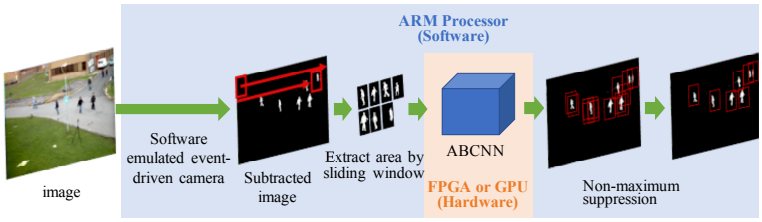

Fig. 1. Overview of an object detector system using sliding window for an event-driven camera

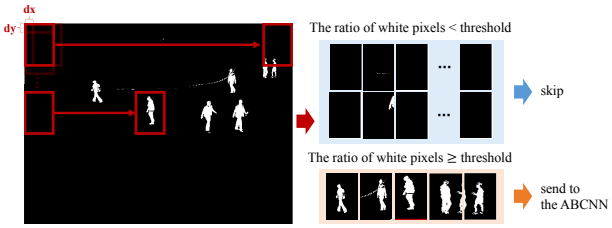

Fig. 2. Sliding window using an event-driven camera

IV. All binarized CNN

An ABCNN is a BCNN in which the first convolutional layer is done in binary form as well. Since it does not require a multi-bit precision convolutional layer to process the input data, all computations can be done in binary. To realize an ABCNN, it is necessary to use another method to decompose the input data into binary form. It reduces resource and energy expenditures while increasing performance and providing a comparable level of accuracy.

V. Dataset Preparation

Since there is currently no dataset available for the output of an event-driven camera, it is necessary to make an imitation

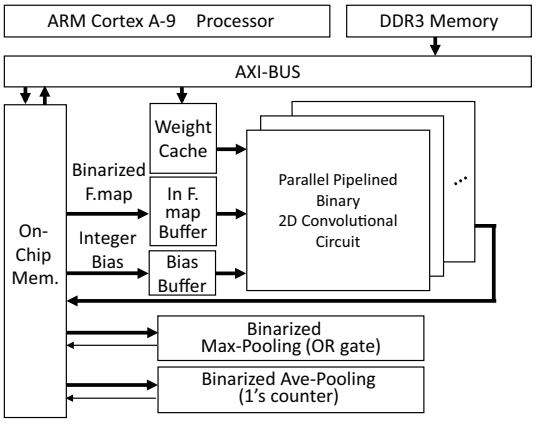

Fig. 3. Overall architecture

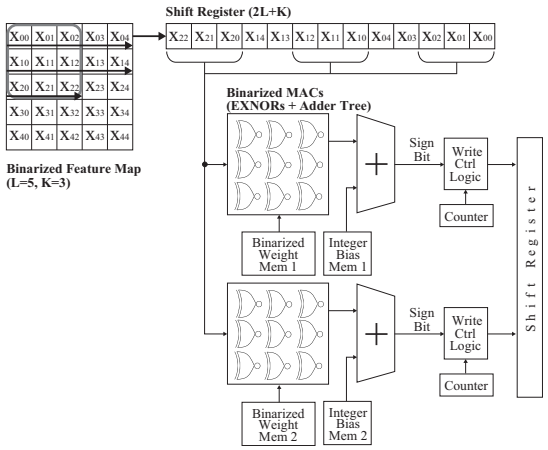

Fig. 4. Parallel pipelined binarized 2D convolutional circuit

dataset. Accordingly, we referred to an existing work[3] and created a dataset using the following procedure:

1) Convert three-channel RGB into grayscale
2) Let p_o be a base pixel value, p_n be a pixel value above the base pixel, p_e be a pixel value on the right of the base pixel. Then, apply the following conversion:

$$V_s = \frac{\max(|p_E - p_o|, |p_N - p_o|)}{\max(p_E, p_o, p_N)},$$

where V_s is an intermediate value.
3) By using the following threshold, we convert the intermediate value into binary:

$$V_o = sgn(V_s - V_{th}),$$

where V_o is a converted pixel value on the base pixel.

Using the above steps, RGB three-channel 8-bit width images were converted into bitmap images.

VI. Implemented Circuit

Fig. 3 shows the overall circuit[5]. Since the on-chip BRAMs realize the memory part of the architecture, this circuit provides increased power efficiency. Since a binarized convolutional circuit is shared among all convolutional layers, this circuit achieves a reduction in area. We improved conventional binarized convolutional circuit[8] to make it possible to compute MACs in parallel. Fig. 4 shows the parallel kernel computation circuit. Since the kernel number of each layer used in this paper is fixed at 64, we create a convolutional circuit that has two binarized MAC circuits to compute in parallel, and then we also confirmed this improvement does not affect the area significantly.

VII. Demonstration

We demonstrate an ABCNN with a software emulated event-driven camera on a zcu102 FPGA evaluation board and a Jetson TX2 GPU (NVIDIA Corp.) using the PETS 2009 Benchmark Data[1]. We developed the ABCNN circuit using the GUINNESS[6] development environment based on Chainer[2] that generates base circuit, and GPU model is realized by Chainer. The experimental results are shown in Table. I. The GPU system power consumption was 2.5 watts while the FPGA consumed only 0.2 watts, thus indicating that our proposed method achieves a 92% power reduction. As for performance, the FPGA system was 4.3 times faster than the comparable GPU system, and regarding performance per power consumption (FPS/W), the FPGA was 54.2 times higher than the GPU. Thus, our proposed method is more suitable for the embedded systems based on cameras with fixed angles and position (such as security cameras).

TABLE I
EVALUATE EVENT-BASED OBJECT DETECTION SYSTEM

	Mobile GPU	FPGA
Platform	Jetson TX2	zcu102
Speed [avg. FPS]	3.0	13.0
Power [watts]	2.5	0.2
Efficiency [FPS/watt]	1.2	65.0

References

[1] PETS 2009 Benchmark Data, http://www.cvg.reading.ac.uk/PETS2009/a.html
[2] Chainer: A powerful, flexible, and intuitive framework of neural networks, http://chainer.org/
[3] L. Benini, "Deep Learning with Low Precision Hardware", http://si.epfl.ch/files/content/sites/si/files/shared/Logic2017.
[4] M. Courbariaux, I. Hubara, D. Soudry, R.E.Yaniv, Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1", *Computer Research Repository (CoRR)*, 2016.
[5] H. Nakahara, T. Fujii and S. Sato, "A Fully Connected Layer Elimination for a Binarized Convolutional Neural Network on an FPGA," *FPL*, pp.1-4, 2017.
[6] H. Nakahara, H. Yonekawa, T. Fujii, M. Shimoda and S. Sato, "A demonstration of the GUINNESS: A GUI based neural NEtwork SyntheSizer for an FPGA," FPL, pp.1, 2017.
[7] M. Shimoda, S. Sato, and H. Nakahara, "All Binarized Convolutional Neural Network and Its implementation on an FPGA," The International Conference on Field-Programmable Technology (FPT 2017), pp.291-294, 2017.
[8] H. Yonekawa and H. Nakahara, "An On-chip Memory Batch Normalization Free Binarized Convolutional Deep Neural Network on an FPGA", *IPDPS*, pp. 98-105, 2017.