IEICE TRANS. INF. & SYST., VOL.E102–D, NO.5 MAY 2019

PAPER Special Section on Reconfigurable Systems

Power Efficient Object Detector with an Event-Driven Camera for Moving Object Surveillance on an FPGA

Masayuki SHIMODA[†a)], ***Nonmember***, **Shimpei SATO**[†b)], ***and*** **Hiroki NAKAHARA**[†c)], ***Members***

SUMMARY We propose an object detector using a sliding window method for an event-driven camera which outputs a subtracted frame (usually a binary value) when changes are detected in captured images. Since sliding window skips unchanged portions of the output, the number of target object area candidates decreases dramatically, which means that our system operates faster and with lower power consumption than a system using a straightforward sliding window approach. Since the event-driven camera output consists of binary precision frames, an all binarized convolutional neural network (ABCNN) can be available, which means that it allows all convolutional layers to share the same binarized convolutional circuit, thereby reducing the area requirement. We implemented our proposed method on the Xilinx Inc. Zedboard and then evaluated it using the PETS 2009 dataset. The results showed that our system outperformed BCNN system from the viewpoint of detection performance, hardware requirement, and computation time. Also, we showed that FPGA is an ideal method for our system than mobile GPU. From these results, our proposed system is more suitable for the embedded systems based on stationary cameras (such as security cameras).

key words: *event-driven camera, object detector, all binarized convolutional neural network, FPGA*

1. Introduction

Object detection systems are used to infer locations and object classes in a picture. Recently, they have been increasingly required in a variety of embedded systems such as robots, security cameras, and drones. However, when incorporated into embedded systems, it is essential to restrict the device size, reduce power consumption, and provide real-time processing performance.

Convolutional neural networks (CNNs) are widely used for computer vision tasks [1]–[3]. In these cited works, it can be seen that CNNs outperform conventional techniques. Recently, a variety of object detectors based on CNNs have been proposed. These systems can be roughly divided into two types. The first type includes a detection component for object area candidates and a classification component that is achieved using a CNN. In its representative method, which is known as regions with CNN features (R-CNN) [4], the object region proposal component is achieved by selective search [5], and the classification component is obtained using a CNN. Since the system consists of two parts, its primary disadvantage is that it requires time and effort to adjust all parameters for two components respectively. However, since each part only has to perform one task, both can be small. Also, an advantage of the system is that if it is necessary to detect new classes, then only the classification part needs to be updated. The second type is a single-shot detector that consists of a CNN and infers locations and classes simultaneously (representative methods are SSD [6] and YOLO9000 [7]). Although it computes detection and classification at a time, it requires a massive amount of hardware and computation time.

1.1 Proposal

We propose an object detector using a sliding window with an event-driven camera that outputs a diff-picture (usually binary value) when change is detected in the captured images. When an event-driven camera is used to operate a security camera with a fixed position and viewing angle, the camera only outputs a diff-picture (binary value) when change is detected. Focusing on this feature and an object detection system, we find that our proposed method improves computation time and reduces power consumption.

When considering the implementation of a camera-based object detector, we determined that larger systems, such as single-shot detectors, are unsuitable. In contrast, since a detection system that consists of two parts can be realized on a smaller circuit than a single-shot detector, this model would be more suitable for our purposes. Therefore, our proposed system uses a system that consists of two parts.

In a previous work related to object detector with an event-driven camera [8], it was that expanding the RISC-V for CNN made it possible to improve computation time and energy gains. However, since CPUs are unsuitable for real-time processing, we implement the CNN on an FPGA.

Realization of an object detector dedicated to bitmap leads to a reduction in computation. More specifically, when performing sliding window, use of binary data can reduce the number of object area candidates by order of magnitude since it only detects an area if the ratio of white pixels (the detected pixels of the moving object) inside the bounding box exceeds the preset threshold. Since the input data of a binarized CNN (BCNN) are usually RGB three-channel color images, its first convolutional layer computes in integer precision. On the other hand, since the event-driven camera output is binary, a BCNN can be replaced with an ABCNN, in which the input of all convolutional layers

Manuscript received July 27, 2018.
Manuscript revised December 3, 2018.
Manuscript publicized February 27, 2019.

[†]The authors are with Tokyo Institute of Technology, Tokyo, 152–8550 Japan.
a) E-mail: shimoda@reconf.ict.e.titech.ac.jp
b) E-mail: satos@eda.ict.e.titech.ac.jp
c) E-mail: nakahara@reconf.ict.e.titech.ac.jp

DOI: 10.1587/transinf.2018RCP0005

is binarized. Realization of all binarization leads to area reduction since a single binarized convolutional circuit is shared among all convolutional layers. Because of this, our proposed method is capable of reducing energy consumption and area usage in comparison with conventional object detection systems.

This paper makes the following contributions:

1. We suggest an object detector using a sliding window with an event-driven camera. Since the sliding window is performed on binary images, the number of object candidates to be classified decreases dramatically, which means that our system achieves high power efficiency and small area requirement.
2. We investigate the detection performance on PETS 2009 dataset. The results show that our system outperformed BCNN-based system in term of f-score.
3. We implement the ABCNN-based and the BCNN-based system on FPGAs and compare them. Compared with BCNN-based one, hardware requirement and computation time are dramatically improved.
4. We realize our proposal for an FPGA and a mobile GPU and compare them in term of speed, power, and power efficiency. The results show that FPGA is more suitable for our proposal than mobile GPU.

This paper is an update version of past works [9] and [10].

2. Related Work

2.1 Type of Situations

There are two scenarios under which object detection is performed. The first one is the situation where the detection is performed using moving camera, such as self-driving cars [11], [12], [12]–[14], a robot [15], and so on. Due to the moving camera, the situation does not allow us to perform background subtraction. Therefore, some works use semantic segmentation instead to obtain high precision.

The second one is the situation where object detection is performed using a stationary camera such as surveillance cameras. For any specific environments, many models are proposed, such as pedestrian detection [16], vehicle re-identification (Re-Id) [17]–[19], vehicle detection [20], [21], attribute recognition [22], action detection [23], change detection [24]–[26], vessel detection in maritime scenarios [27], [28]. While their works realize high accuracies and robustness, the detection models become bigger and more complex, and then some of them are not suitable for embedded systems. Our target situation is the second one, and we propose an object detector system for embedded systems. Because of performing object detection on bitmap images, while realizing efficient power-consumption compared with conventional ones, our proposal cannot be applied to some applications like attribute recognition.

2.2 Type of Object Detection Systems

For object detection realizations, there are two methods divided roughly: single shot detectors and two-component systems.

A single shot detector consists of a big CNN only, and localization and classes are inferred simultaneously [6], [7], [29]–[31]. Although such CNN ones accept to both detection and learning at a time, the computation time and area become too large to implement on embedded systems.

The other is a system which consists of both an area extractor and a classifier. An area extractor, such as selective search [5], sliding window, region proposal network (RPN) [30] extracts target object area candidates, and the extracted areas are fed into the classifier, such as support vector machine (SVM) [32]. Before CNN is successful for wide areas, the classifier classifies the extracted areas based on hand-crafted features [33]–[38]. After that, CNNs [1]–[4], [39] are often used as classifier. Since such area extractors propose a lot of candidates, the computation of their classifier dramatically increases. A region of interest filtering using saliency detection [40] and optimized sliding window [41] are proposed to decrease the number of proposed windows. In this paper, we realize object detector using two-component type. Since our proposed system requires the object detector to run on a bitmap, many candidates dramatically decreases, and its classifier can be smaller.

2.3 Quantized CNN and Implemental Circuit for FPGAs

Since full precision CNNs require vast amounts of computation and memory, low precision CNNs are proposed [42]–[53]. Among them, binarized CNNs (BCNNs) [43] which constrain the value to $+1$ or -1 provide dramatically lower storage size and bandwidth, while maintaining comparable accuracies to full precision networks. Additionally, Intel [54] shows FPGAs offer order of magnitude efficiency improvements over optimized BCNN software CPU and GPU implementations using Alexnet [3] and VGG [2]. Therefore, To realize dedicated circuit for quantized CNN on FPGAs, there are many works [55]–[64], and some of them use high-level synthesis (HLS) tools such as Xilinx Vivado HLS [65], Xilinx SDSoC [66] and Intel FPGA SDK for OpenCL [67], [68]. To realize high power efficiency circuit, we employ BCNN and the circuit based on [60]. Due to binarization, all parameters can be stored in on-chip memory. Also, since we realize all binarized CNN, the circuit based on [60] can be shared binarized among all convolutional layers, and then it provides high power efficiency.

3. Object Detector Using Sliding Window

Figure 2 shows the overall architecture of an object detector using a sliding window and an event-driven camera. This 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 region has been detected by the proportion of white pixels inside the bounding box. The extracted pictures are resized to

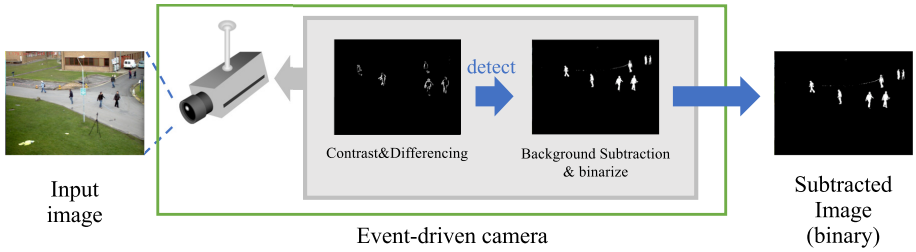

Fig. 1 Event-driven camera behavior

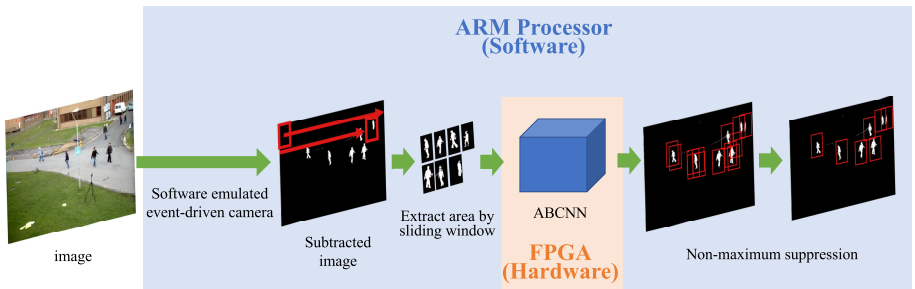

Fig. 2 Overview of an object detector system using a sliding window

Table 1 Comparison between event-driven and frame-driven cameras [8]

Scenario	Frame-driven camera	Event-driven camera
Image sensor power consumption	1.1mW @30fps	100μW @50fps
Image size	632,446 bits	8,192 bits
Image sensor energy for frame capture	66.7 μJ	2.0 μJ

40×40 size and applied to the ABCNN. All binarization was found to improve both the area and computation time. 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 [4] is used to reduce the extras to a single box.

The ARM on the FPGA creates an imitation of the event-driven camera outputs that are then applied to the sliding window to detect the object regions. Then, the proposed regions are sent to the ABCNN on the FPGA (or GPU) where computation is performed. The inference results are sent to the ARM on the FPGA which draws a bounding box on the corresponding proposed region if the result is designated as *human*. Finally, by non-maximum suppression, the excess bounding boxes are reduced to a single box. In this paper, we investigate the FPS and power consumption of the system using both an FPGA and a GPU.

3.1 Event-Driven Camera

Figure 1 shows the behavior of an event-driven camera, which is a camera that only outputs a picture when captured image changes. Such cameras extract 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

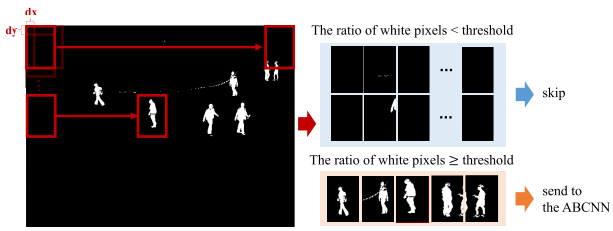

Fig. 3 Sliding window on an event-driven camera

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. In this study, we realize the camera artificially by BackgroundSubtractorMOG2 of openCV3, since the cameras are not available for purchase now. As for detail of event-driven camera performance, Table 1 compares with frame-driven one. Since it reduces the amount of data crossing the costly analog-to-digital border, the power consumption of the event-driven camera is more than ten times lower.

3.2 Sliding Window

One of the commonly used object region proposed methods is called the sliding window. Figure 3 shows an overview of this method. As can be seen in the figure, the bounding box moves along the x- and y-axis by preset dx and dy respec-

tively from top left to down right on the picture, and then the areas inside the box are cropped as object area candidates. Since the extractor generates a large number of windows to be classified, it requires high computational power. However, when an event-driven camera is used, the sliding window is performed on a bitmap, and then candidate regions are only proposed when the ratio of white pixels exceeds a certain threshold. It leads to a significant reduction in many proposed candidates. In the case of the PETS 2009 dataset, the number of RGB picture candidates is normally in the hundreds. However, when a binary grayscale picture is used, the number drops to between zero and 20.

Usually, to deal with various scale objects, an input picture is resized as the bounding box moves. However, in the case of a stationary camera, since most of the captured objects are close to the same size, we did not resize the input picture in our experiments.

4. Binarized CNN (BCNN)

Recently, Courbariaux et al. proposed a CNN whose weights and activation are binarized [43]. The binarized neuron model is given as follows:

$$s = \sum_{i=1}^{n} w_i^b x_i^b + w_0$$
$$z^b = f_{sign}(s),$$

where $x^b, w^b, z^b \in \{-1, +1\}$ and w_0 denotes a bias which corrects the input data deviation in integer precision (normally BCNNs require batch normalization (BN) [69]. However, [60] shows that neurons using BN are equal to neurons which have integer-precision bias), and $f_{sign}(Y)$ denotes the signed activation function as follows:

$$f_{sign}(x) = \begin{cases} +1 & \text{if x} \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

For implementation on it, since an FPGA cannot represent -1 directly, we map a logical zero to -1. In this case, since the binarized multiplication can be realized by an XNOR gate, the hardware area is dramatically reduced compared to a floating-point unit. Additionally, the binarized weights and activation lead to a comparative reduction in the required memory bandwidth. Thus, by using a BCNN, area and performance are further reduced and improved respectively.

4.1 Internal FC Layer Replacement with a Binarized Average Pooling Layer

Conventionally, most CNN parameters are focused on the FC layers, which requires a significant amount of memory. To reduce memory size, we replace the FC layers with a binarized average pooling layer [70]. We show how such a layer can infer correctly without the use of internal FC layers. A binarized average pooling operation with a $K \times K$ kernel is given as follows:

$$y_{i,j} = \frac{1}{K^2} \sum_{v=1}^{K-1} \sum_{u=1}^{K-1} x_{i+u,j+v},$$

where x is a binary input value, K is a kernel size and y is a binary output value. Here a majority operation is implied. The binarized average pooling layer is set next to the last convolutional layer. If the size of the feature map output by the last convolutional layer is $L \times L$, then the kernel size of the binarized average pooling is set to $L \times L$. In this case, the output of the average pooling is 1D. These features are then applied to the FC layer, which maps them to classes. In our proposed method, we use this technique to reduce memory utilization.

5. All Binarized CNN (ABCNN)

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. ABCNNs reduce resource and power energy while increasing performance and providing a comparable level of accuracy, compared with BCNNs.

6. Implemented Circuit

Figure 4 shows the overall circuit [70]. The design consists of buffer parts for parameters, binarized convolutional circuit part, binarized max-pooing circuit part, and binarized average pooling circuit part (BCNN design also includes integer convolutional circuit part). Firstly, all parameters such as weights and biases area loaded into on-chip buffers from DDR3. After that, the processor sends an input image to the design, and the binarized convolution/max pooling/average pooling operations are performed (in the case of BCNN design, the integer convolution operation is performed). The output goes to the on-chip memory, to be read to compute

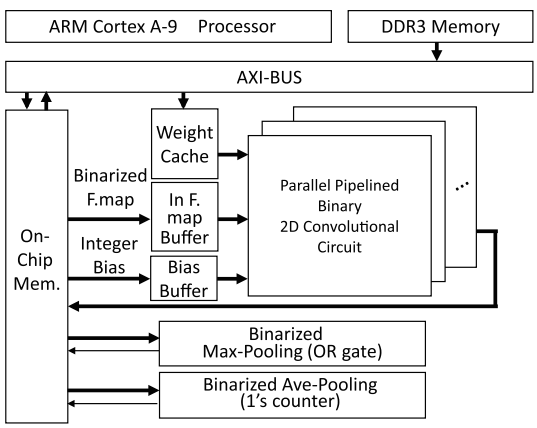

Fig. 4 Overall architecture

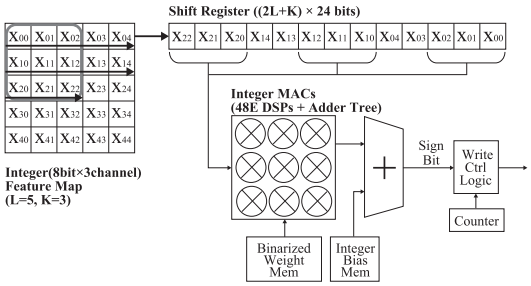

Fig. 5 Pipelined 2D convolutional circuit

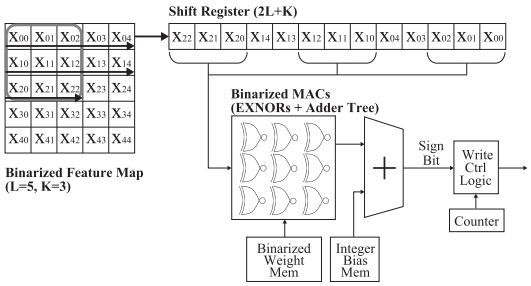

Fig. 6 Pipelined binarized 2D convolutional circuit

the next layer operation. Since the on-chip BRAMs realize the memory part of the architecture, this circuit provides increased power efficiency. As for ABCNN design, since a binarized convolutional circuit can be shared among all convolutional layers, the overall circuit is smaller than BCNN one.

Figure 5 shows a convolutional circuit in integer precision [71]. The convolutional circuit consists of a DSP48E, adder trees, a bias adder, and a write controller. Since convolutional operations access the same value lead to multiple memory accesses, the circuit uses a shift register and store values outside the memory. In this study, the first convolution of the BCNN is realized by this circuit.

Figure 6 shows a binarized convolutional circuit [60]. Figure 5 is similar to Fig. 6 except for using dedicated DSP blocks, since the input of the circuit is binary, and then it allows the DSP48E to be replaced with a bitwise XNOR gate. After multiplication, the architecture is the same as that of the integer convolutional circuit. It realizes all convolutional layers except for the first layer of the BCNN.

For BCNN design, we improved the circuit in order to make it possible to compute MACs in parallel. Figure 7 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 an even number of binarized MACs and can compute in parallel, and then we also confirmed this improvement does not affect the area significantly. The integer convolutional circuit parallelization is four, and the binarized one is two.

Figure 8 shows a binarized max pooling layer circuit. Since a binarized max pooling operation is realized by the logical OR, a binarized max pooling circuit is realized by a line buffer and bitwise OR gates.

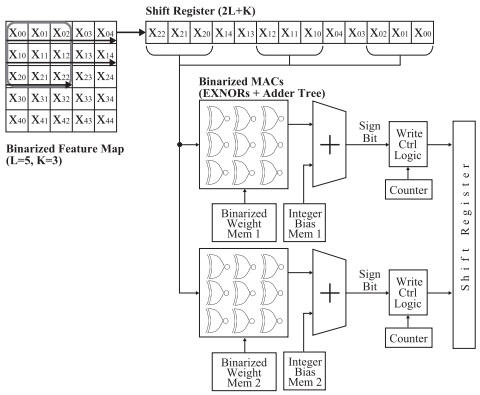

Fig. 7 Parallel pipelined binarized 2D convolutional circuit

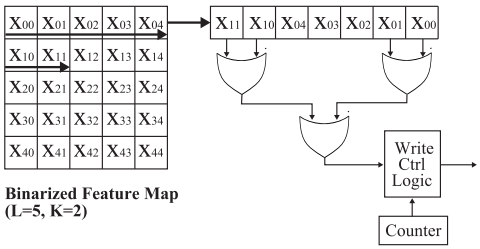

Fig. 8 Binarized max pooling circuit

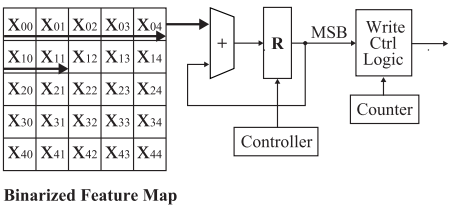

Fig. 9 Binarized average pooling circuit

Figure 9 shows an average pooling layer circuit. A binarized average pooling is realized by a 1's counter and a threshold circuit.

7. Experimental Results

7.1 Comparison of Accuracies

We evaluated our proposed system on PETS 2009 Dataset [72], which consists of videos of campus from Kyushu University including many pedestrians. While the dataset has videos from eight different views, we only use S2L1, and the ground truth labels of MOT 2015 [73] are employed. Since there is currently no dataset available for the output of an event-driven camera, it is necessary to make an imitation dataset from the dataset. Accordingly, We applied background subtraction to the dataset, then extract areas randomly, and classified them. Our used models are shown in Table 2 and Table 3. Table 4 shows precision, recall and F-score for each CNN. Compared with a BCNN-based system, while recall decreased by 0.48, precision increased by 0.29,

Table 2 Used BCNN model

Layer	Type	#Output f.map	Kernel size	Padding
1	conv	64	3	1
2	conv	64	3	1
3	maxp	64	2	0
4	conv	64	3	1
5	conv	64	3	1
6	maxp	64	2	0
7	conv	64	3	1
8	conv	64	3	1
9	maxp	64	2	0
10	avgp	64	4	0
11	fc	3	1	0

Table 3 Used ABCNN model

Layer	Type	#Output f.map	Kernel size	Padding
1	conv	64	3	1
2	conv	64	3	1
3	conv	64	3	1
4	avgp	64	40	0
5	fc	2	1	0

Table 4 Evaluation of precision, recall, and f-score on pets2009 dataset

Models	IoU (0.3)		
	Precision	Recall	F-score
BCNN	0.14	0.93	0.24
ABCNN	0.43	0.45	0.44

Fig. 10 Detection result. While bounding boxes are represented as the detection result of human. This figure shows that our system cannot detect the humans who are behind any obstacles.

and f-score increased by 0.2. Since our system cannot detect the humans who are behind any obstacles such as street lights, as shown in Fig. 10, the decrease is caused. On the other hand, since BCNN-based system proposes many background areas, its precision is dramatically decreased. Our system achieved superior precision thanks to background subtraction.

7.2 Implementation Results

We implemented an ABCNN and a BCNN on FPGAs and investigated their resulting areas and latencies, using the GUINNESS [74] development environment based on Chainer [75] that generates base circuit and Xilinx Inc. Vivado SDSoC 2017.4 with a constraint of 100 MHz. For BCNN implementation we used a Xilinx Inc. Zynq UltraScale+ MPSoC zcu102 evaluation board, which is equipped with a Xilinx Zynq UltraScale+ MPSoC FPGA (ZU9EG, 68,520 Slices, 269,200 FFs, 1,824 18Kb BRAMs, 2,520 DSP48Es), and for ABCNN implementation we use the Xilinx Inc. Zedboard, which has the Xilinx Zynq FPGA (XC7Z020, 53,200 LUTs, 106,400 FFs, 280 18Kb BRAMs, 220 DSP48Es). Table 5 shows an implementation result. Compared with BCNN implementation, The #DSP48E decreased by 135, the #18 Kbit BRAM decreased by 498, the #FF decreased by 9,253, the #LUT decreased by 17,591 and the computation time is about 2 times faster per a proposed area. Thanks to all binarization, our proposed one does not require any DSP48E and big CNN, which means that our system can be realized by the tiny circuit and be more suitable for embedded systems.

Table 5 Implementation on an FPGA (utilization of each platform). Note that, the clock cycle is estimated by Vivado HLS.

	BCNN	ABCNN (proposal)
Platform	ZCU102	Zedboard
18 Kbit BRAM	548 (30.04)	50 (17.86)
DSP48E	135 (5.36)	0 (0.00)
FF	23,852 (4.35)	14,599 (13.72)
LUT	28,892 (10.54)	11,301 (21.24)
Clock cycle	3,206,021	1,568,744

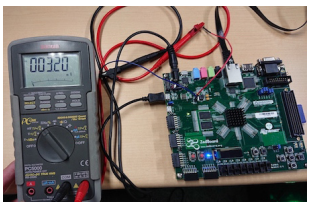

Fig. 11 Measurement of the FPGA power consumption.

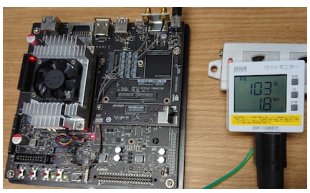

Fig. 12 Measurement of the GPU power consumption using a power monitor.

Next, we realized the proposed system with a software emulated event-driven camera on both the FPGA and a Jetson TX2 GPU (NVIDIA Corp.) to compare FPS and power consumption. In this paper, we consider the dynamic board power consumption as the system power consumption. The power consumption of the FPGA is measured using a tester as shown in Fig. 11, and a power monitor determined the GPU one as shown in Fig. 12. The Chainer framework realizes the GPU system. The results are shown in Table 6.

The static power consumption of the FPGA system is 3.08 W, and the runtime power consumption is 3.20 W. As for the dynamic power consumption, the GPU system power consumption was 2.1 W while the FPGA consumed only 0.14 W, thus showing that our proposal reduced power

Table 6 Evaluate event-based object detection system

	Mobile GPU	FPGA (proposal)
Platform	Jetson TX2	Zedboard
Speed [avg. FPS]	1.8	3.5
Power [W]	2.1	0.14
Efficiency [FPS/W]	0.9	25.0

consumption by 1.96 power reduction. As for performance, the GPU achieved an average of 1.8 FPS, whereas the FPGA achieved an average 3.5 FPS. From these results, it is clear that our proposed system was about 2 times faster than a comparable GPU system, and that in terms of performance per power consumption (FPS/W), the FPGA was 27.8 times higher than the GPU. Therefore, our system provides high power efficiency.

8. Conclusion

We proposed an object detection system using a sliding window with an event-driven camera. Thanks to background subtraction performed in the camera, it is easy for a sliding window to detect object area candidates, and it provides both reduced power consumption and high-speed detection. We evaluated our proposed system in term of detection performance, hardware requirement, and power efficiency. The results showed that our system outperformed BCNN system from the viewpoint of all except for recall, and also showed that FPGA is more suitable for our system than mobile GPU. From these results, under the constraints of stationary camera (such as in the case of a security camera), our proposed system is more suitable than are conventional object detection systems.

Acknowledgments

This research is supported in part by the Grants in Aid for Scientific Research from JSPS, and the New Energy and Industrial Technology Development Organization (NEDO). In addition, thanks are extended to the Xilinx University Program (XUP), the Intel University Program, and NVidia Corp. for their support.

References

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1–9, June 2015.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," International Conference on Learning Representations, 2015.

[3] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems 25, pp.1097–1105, 2012.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp.580–587, June 2014.

[5] M.B. Blaschko and C.H. Lampert, "Learning to localize objects with structured output regression," ECCV '08 Proc. 10th European Conference on Computer Vision: Part I, pp.2–15, 2008.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S.E. Reed, C.-Y. Fu, and A.C. Berg, "Ssd: Single shot multibox detector," European conference on computer vision, pp.21–37, 2016.

[7] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.6517–6525, 2017.

[8] M. Gottardi, N. Massari, and S.A. Jawed, "A 100 μ w 128 $\times$ 64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications," IEEE J. Solid-State Circuits, vol.44, no.5, pp.1582–1592, 2009.

[9] M. Shimoda, S. Sato, and H. Nakahara, Power efficient object detec-tor with an event-driven camera on an FPGA, in Proceedings of the 9th International Symposium on Highly-Efficient Accelerators andRecongurable Technologies, HEART 2018, Toronto, ON, p.10:110:6, Canada, June 2018.

[10] M. Shimoda, S. Sato, and H. Nakahara, "All binarized convolutional neural network and its implementation on an fpga," 2017 International Conference on Field Programmable Technology (ICFPT), pp.291–294, Dec. 2017.

[11] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[12] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2110–2118, 2016.

[13] J. Li and Z. Wang, "Real-time traffic sign recognition based on efficient cnns in the wild," IEEE Trans. Intell. Transp. Syst., pp.1–10, 2018.

[14] E. Peng, F. Chen, and X. Song, "Traffic sign detection with convolutional neural networks," International Conference on Cognitive Systems and Signal Processing, pp.214–224, 2016.

[15] D. Ribeiro, A. Mateus, J.C. Nascimento, and P. Miraldo, "A real-time pedestrian detector using deep learning for human-aware navigation," arXiv: Robotics, 2016.

[16] S. Rujikietgumjorn and N. Watcharapinchai, "Real-time hog-based pedestrian detection in thermal images for an embedded system," 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.1–6, Aug. 2017.

[17] X. Liu, W. Liu, H. Ma, and H. Fu, "Large-scale vehicle re-identification in urban surveillance videos," 2016 IEEE International Conference on Multimedia and Expo (ICME), pp.1–6, July 2016.

[18] Y. Tang, D. Wu, Z. Jin, W. Zou, and X. Li, "Multi-modal metric learning for vehicle re-identification in traffic surveillance environment," 2017 IEEE International Conference on Image Processing (ICIP), pp.2254–2258, Sept. 2017.

[19] X. Liu, W. Liu, T. Mei, and H. Ma, "Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance," IEEE Trans. Multimedia, vol.20, no.3, pp.645–658, March 2018.

[20] R.S. Feris, B. Siddiquie, J. Petterson, Y. Zhai, A. Datta, L.M. Brown, and S. Pankanti, "Large-scale vehicle detection, indexing, and search in urban surveillance videos," IEEE Trans. Multimedia, vol.14, no.1, pp.28–42, Feb. 2012.

[21] R. Feris, R. Bobbitt, S. Pankanti, and M.-T. Sun, "Efficient 24/7 object detection in surveillance videos," 2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.1–6, Aug. 2015.

[22] M. Fabbri, S. Calderara, and R. Cucchiara, "Generative adversarial models for people attribute recognition in surveillance," 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.1–6, Aug. 2017.

[23] C. Bahnsen and T.B. Moeslund, "Detecting road user actions in traffic intersections using rgb and thermal video," 2015 12th IEEE International Conference on Advanced Video and Signal Based

Surveillance (AVSS), pp.1–6, Aug. 2015.
[24] K. Lim, W.-D. Jang, and C.-S. Kim, "Background subtraction using encoder-decoder structured convolutional neural network," 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.1–6, Aug. 2017.
[25] A. Shimada, H. Nagahara, and R. Taniguchi, "Change detection on light field for active video surveillance," 2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.1–6, Aug. 2015.
[26] T. Minematsu, A. Shimada, and R. Taniguchi, "Analytics of deep neural network in change detection," 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.1–6, Aug. 2017.
[27] F. Bousetouane and B. Morris, "Fast cnn surveillance pipeline for fine-grained vessel classification and detection in maritime scenarios," 2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp.242–248, Aug. 2016.
[28] D.K. Prasad, C.K. Prasath, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek, "Object detection in a maritime environment: Performance evaluation of background subtraction methods," IEEE Trans. Intell. Transp. Syst., pp.1–16, 2018.
[29] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol.37, no.9, pp.1904–1916, 2015.
[30] S. Ren, K. He, R.B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," IEEE Trans. Pattern Anal. Mach. Intell., vol.39, no.6, pp.1137–1149, 2017.
[31] J. Redmon, S.K. Divvala, R.B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.779–788, 2016.
[32] Du Shuxin and Wu Tiejun, "Support vector machines for pattern recognition," Journal of Zhejiang University (Engineering Science), 37(5): 521-527, 2003.
[33] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol.1, pp.886–893, June 2005.
[34] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, vol.60, no.2, pp.91–110, 2004.
[35] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp.1–8, June 2008.
[36] P. Dollár, Z. Tu, P. Perona, and S.J. Belongie, "Integral channel features," Proc. British Machine Vision Conference, BMVC 2009, London, UK, Sept. 7-10, 2009, pp.1–11, 2009.
[37] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," IEEE Trans. Pattern Anal. Mach. Intell., vol.36, no.8, pp.1532–1545, Aug. 2014.
[38] W. Nam, P. Dollár, and J.H. Han, "Local decorrelation for improved pedestrian detection," Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Dec. 8-13 2014, Montreal, Quebec, Canada, pp.424–432, 2014.
[39] R.B. Girshick, "Fast r-cnn," 2015 IEEE International Conference on Computer Vision (ICCV), pp.1440–1448, 2015.
[40] M.-M. Cheng, J. Warrell, W.-Y. Lin, S. Zheng, V. Vineet, and N. Crook, "Efficient salient region detection with soft image abstraction," 2013 IEEE International Conference on Computer Vision, pp.1529–1536, Dec. 2013.
[41] V.H.C. de Melo, S. Leão, D. Menotti, and W.R. Schwartz, "An optimized sliding window approach to pedestrian detection," 2014 22nd International Conference on Pattern Recognition, pp.4346–4351, Aug. 2014.
[42] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," European conference on computer vision, pp.525–542, 2016.
[43] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," arXiv preprint arXiv:1602.02830, 2016.
[44] F. Li and B. Liu, "Ternary weight networks," arXiv preprint arXiv:1605.04711, 2016.
[45] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," arXiv preprint arXiv:1606.06160, 2016.
[46] D.A. Gudovskiy and L. Rigazio, "Shiftcnn: Generalized low-precision architecture for inference of convolutional neural networks," arXiv preprint arXiv:1706.02393, 2017.
[47] M. Kim and P. Smaragdis, "Bitwise neural networks," arXiv preprint arXiv:1601.06071, 2016.
[48] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," arXiv preprint arXiv:1511.06488, 2015.
[49] S. Han, H. Mao, and W.J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," International Conference on Learning Representations, 2016.
[50] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: training deep neural networks with binary weights during propagations," Neural Information Processing Systems, pp.3123–3131, 2015.
[51] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented approximation of convolutional neural networks," arXiv preprint arXiv:1604.03168, 2016.
[52] G. Venkatesh, E. Nurvitadhi, and D. Marr, "Accelerating deep convolutional networks using low-precision and sparsity," 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp.2861–2865, 2017.
[53] D. Miyashita, E.H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," arXiv preprint arXiv:1603.01025, 2016.
[54] E. Nurvitadhi, D. Sheffield, J. Sim, A.K. Mishra, G. Venkatesh, and D. Marr, "Accelerating binarized neural networks: Comparison of fpga, cpu, gpu, and asic," 2016 International Conference on Field-Programmable Technology (FPT), pp.77–84, 2016.
[55] D.J.M. Moss, E. Nurvitadhi, J. Sim, A.K. Mishra, D. Marr, S. Subhaschandra, and P.H.W. Leong, "High performance binary neural networks on the xeon+FPGA™ platform," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp.1–4, 2017.
[56] L. Jiao, C. Luo, W. Cao, X. Zhou, and L. Wang, "Accelerating low bit-width convolutional neural networks with embedded fpga," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp.1–4, 2017.
[57] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M.B. Srivastava, R. Gupta, and Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable fpgas," Proc. 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.15–24, 2017.
[58] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going deeper with embedded fpga platform for convolutional neural network," Proc. 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.26–35, 2016.
[59] A. Prost-Boucle, A. Bourge, F. Pétrot, H. Alemdar, N. Caldwell, and V. Leroy, "Scalable high-performance architecture for convolutional ternary neural networks on fpga," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp.1–7, Sept. 2017.
[60] H. Yonekawa and H. Nakahara, "An on-chip memory batch normalization free binarized convolutional deep neural network on an fpga," 24th Reconfigurable Architectures Workshop (RAW 2017),

2017.
[61] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," Proc. 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.161–170, 2015.
[62] C. Farabet, Y. LeCun, K. Kavukcuoglu, E. Culurciello, B. Martini, P. Akselrod, and S. Talay, "Large-scale fpga-based convolutional networks," Scaling up Machine Learning: Parallel and Distributed Approaches, pp.399–419, 2011.
[63] K. Abdelouahab, M. Pelcat, F. Berry, and J. S'erot, "Accelerating cnn inference on fpgas: A survey," arXiv preprint arXiv:1806.01683, 2018.
[64] Y. Umuroglu, N.J. Fraser, G. Gambardella, M. Blott, P.H.W. Leong, M. Jahre, and K.A. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," Proc. 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.65–74, 2017.
[65] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K.A. Vissers, and Z. Zhang, "High-level synthesis for fpgas: From prototyping to deployment," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol.30, no.4, pp.473–491, 2011.
[66] V. Kathail, J. Hwang, W. Sun, Y. Chobe, T. Shui, and J. Carrillo, "Sdsoc: A higher-level programming environment for zynq soc and ultrascale+ mpsoc," Proc. 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, p.4, 2016.
[67] T.S. Czajkowski, U. Aydonat, D. Denisenko, J. Freeman, M. Kinsner, D. Neto, J. Wong, P. Yiannacouras, and D.P. Singh, "From opencl to high-performance hardware on fpgas," 22nd International Conference on Field Programmable Logic and Applications (FPL), pp.531–534, 2012.
[68] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J.O.G. Hock, Y.T. Liew, K. Srivatsan, D.J.M. Moss, S. Subhaschandra, and G. Boudoukh, "Can fpgas beat gpus in accelerating next-generation deep neural networks?," Proc. 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.5–14, 2017.
[69] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," International Conference on Machine Learning, pp.448–456, 2015.
[70] H. Nakahara, T. Fujii, and S. Sato, "A fully connected layer elimination for a binarizec convolutional neural network on an fpga," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), pp.1–4, Sept. 2017.
[71] B. Bosi, G. Bois, and Y. Savaria, "Reconfigurable pipelined 2-d convolvers for fast digital signal processing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.7, no.3, pp.299–308, Sept. 1999.
[72] J. Ferryman and A. Shahrokni, "Pets2009: Dataset and challenge," 2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, pp.1–6, Dec. 2009.
[73] "Multiple object tracking benchmark: 2d mot 2015."
[74] 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," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), p.1, Sept. 2017.
[75] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," in Proceedings of Workshop on Machine Learning Systems (LearningSys) in TheTwenty-ninth Annual Conference on Neural Information Processing Systems (NIPS), 2015.

Masayuki Shimoda received the B.E. degree in engineering from Tokyo Institute of Technology, Tokyo, Japan, in 2018. He is currently a Master Student with the Department of Information and Communications Engineering of Tokyo Institute of Technology. His current research interests include Deep Neural Network and FPGA. He is a member of IEEE and IPSJ.

Shimpei Sato received the B.E., M.E., and Ph.D. degrees in engineering all from Tokyo Institute of Technology, Tokyo, Japan, in 2007, 2009, and 2014, respectively. He is currently an Assistant Professor with the Department of Information and Communications Engineering of Tokyo Institute of Technology. From 2014 to 2016, he worked in high performance computing area as a post doctoral researcher, where he investigated an application performance analysis/tuning method. His current research interests include approximate computing realization by architecture design and circuit design and their applications. He is a member of IEEE, ACM, and IPSJ.

Hiroki Nakahara received the B.E., M.E., and Ph.D. degrees in computer science from Kyushu Institute of Technology, Fukuoka, Japan, in 2003, 2005, and 2007, respectively. He has held research/faculty positions at Kyushu Institute of Technology, Iizuka, Japan, Kagoshima University, Kagoshima, Japan, and Ehime University, Ehime, Japan. Now, he is an associate professor at Tokyo Institute of Technology, Japan. He was the Workshop Chairman for the International Workshop on Post-Binary ULSI Systems (ULSIWS) in 2014, 2015, 2016 and 2017, respectively. He searved the Program Chairman for the International Symposium on 8th Highly-Efficient Accelerators and Reconfigurable Technologies (HEART) in 2017. He received the 8th IEEE/ACM MEMOCODE Design Contest 1st Place Award in 2010, the SASIMI Outstanding Paper Award in 2010, IPSJ Yamashita SIG Research Award in 2011, the 11st FIT Funai Best Paper Award in 2012, the 7th IEEE MCSoC-13 Best Paper Award in 2013, and the ISMVL2013 Kenneth C. Smith Early Career Award in 2014, respectively. His research interests include logic synthesis, reconfigurable architecture, digital signal processing, embedded systems, and machine learning. He is a member of the IEEE, the ACM, and the IEICE.