A Real-time 17-Scale Object Detection Accelerator with Adaptive 2000-Stage Classification in 65nm CMOS

Minkyu Kim, *Student Member, IEEE*, Abinash Mohanty, *Member, IEEE,* Deepak Kadetotad, *Student Member, IEEE*, Luning Wei, *Student Member, IEEE*, Xiaofei He, *Senior Member, IEEE*, Yu Cao, *Fellow, IEEE*, Jae-sun Seo, *Senior Member, IEEE*

***Abstract*—Machine learning has become ubiquitous in applications including object detection, image/video classification, and natural language processing. While machine learning algorithms have been successfully used in many practical applications, accurate, fast, and low-power hardware implementations of such algorithms is still a challenging task, especially for mobile systems such as Internet of Things (IoT), autonomous vehicles, and smart drones. This paper presents an energy-efficient programmable ASIC accelerator for object detection. Our ASIC accelerator supports multi-class (e.g., face, traffic sign, car license plate, and pedestrian) that are programmable, many-object (up to 50) in one image with different sizes (17-scale support with 6 down-/11 up-scaling), and high accuracy (AP of 0.87/0.81/0.72/0.76 for FDDB/AFW/BTSD/ Caltech datasets). We designed an integral channel detector with 2,000 classifiers for rigid boosted templates, where the number of stages used for classification can be adaptively controlled depending on the content of the search window. This can be implemented with a more modular hardware, compared to support vector machine (SVM) and deformable parts model (DPM) designs. By jointly optimizing the algorithm and the efficient hardware architecture, the prototype chip implemented in 65nm CMOS demonstrates real-time object detection of 20-50 frames/s with low power consumption of 22.5-181.7mW (0.54-1.75 nJ/pixel) at 0.58-1.1V supply.**

***Index Terms*—object detection, machine learning, classification, real-time, low-power, special-purpose accelerator**

I. Introduction

OBJECT detection is essential for intelligent computer vision applications such as augmented reality (AR), advanced driver assistant systems (ADAS), autonomous control in unmanned aerial vehicles (UAV), smart drones, surveillance systems, and Internet of Things (IoT). Real-time, high accurate and energy-efficient object detection is an essential task for these applications. While significant improvement has recently been made in algorithms [1-6], hardware designs using general-purpose processors such as CPUs, GPUs [7], and FPGAs [8] do not provide satisfactory energy efficiency and speed in order to make real-time decisions within the power envelope of embedded systems. This is due to high computational complexity that varies with algorithms and the large memory/communication requirement independent of input, which generates significant data movement that can be as energy consuming as computation.

Special-purpose ASICs for object detection have been previously proposed [9-12]. A real-time object detection engine using a Histogram of Oriented Gradients (HOG) feature extraction in Support Vector Machine (SVM) was presented in [9]. However, the implementation only supported one scale factor, limiting the detection accuracy and robustness. The authors of [10] designed a specialized engine for face detection and recognition with low power consumption of 23mW, but was not able to support multi-scale factors or multiple faces. Multi-scale pedestrian detection was achieved in [11] with 12 scale factors, but only down-scaling was used, limiting the detection of objects with small number of pixels. A multi-object detection accelerator with Deformable Parts Model (DPM) was implemented in [12] with two programmable object classification engines for 58.6mW power consumption, but still only supported down-scaling.

In this paper, we propose an energy-efficient programmable ASIC accelerator [13] for object detection that overcomes the above limitations:

- Multiple classes (e.g., face, traffic sign, car license plate, pedestrian) that are programmable in the accelerator
- Many objects (up to 50) in one image with multiple scales (17-scale support with 6 down-scaling and 11 up-scaling)
- High accuracy (average precision of 0.87/0.81/0.72/0.76/ 0.53 in FDDB/AFW/BTSD/Caltech plate/INRIA Person datasets) comparable to state-of-the-art algorithms
- Energy-efficient hardware architecture based on rigid boosted templates for low power of 22.5mW and low energy per pixel of 0.54 nJ/pixel

M. Kim, D. Kadetotad, Y. Cao, and J. Seo are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281 USA (e-mail: mkim152@asu.edu).

A. Mohanty was with Arizona State University, Tempe, AZ 85281 USA. He is now with FABU America, Inc., Tempe, AZ 85284 USA.

L. Wei and X. He are with the College of Computer Science, Zhejiang University, Hangzhou, China.

This work was supported in part by NSF grant NSF-CCF-1652866, and C-BRIC, one of six centers in JUMP, a SRC program sponsored by DARPA.

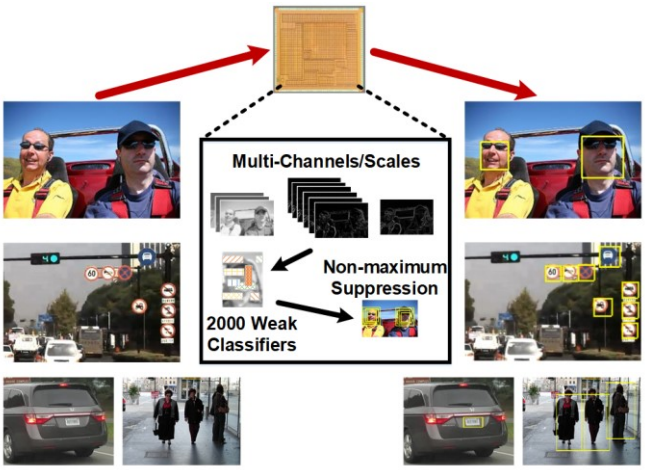


Fig. 1. Illustration of multi-class object detection (e.g., face, traffic sign, car license plate, pedestrian) with 10 channels, 17 scales, 2000 weak classifiers, and non-maximum suppression.

Many object detection algorithms have been using the classification models that are trained on features instead of pixels [1-6]. Hand-crafted features such as the well-known HOG have been traditionally used in object detection including the Viola-Jones algorithm [1], DPM [2], and HeadHunter model [3]. Recently, learned features such as convolutional neural networks (CNNs) have been widely used [4-6]. In general, the CNN learned features outperform the hand-crafted features for object detection accuracy and the hand-crafted features are more energy-efficient than the learned features for hardware implementations. Reference [14] shows the comparison results between two chips: [15] implements the hand-crafted feature using HOG, and [16] implements the learned feature using CNN. Although learned features can reportedly achieve more than 2× average precision (~30 vs. ~65), the accompanying energy consumption per pixel becomes four orders of magnitude higher than that using HOG features.

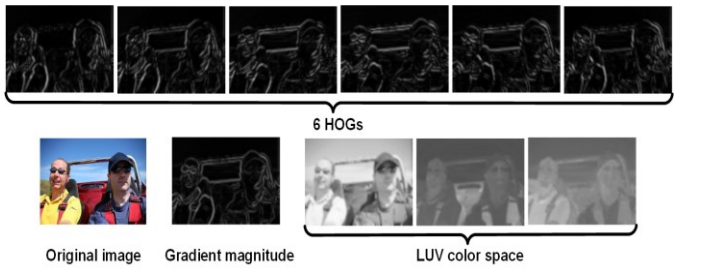
In this work, we employ the HeadHunter model based on rigid templates [3], which achieves state-of-the-art face detection accuracies on AFW [17], FDDB [18], and Pascal VOC [19] datasets compared to other works [2, 4].

Our ASIC accelerator is based on a strong multi-channel including 6 HOGs and 3 LUV and multi-scale model with rigid boosted templates [3], which can detect objects by performing integral of random rectangular regions based on the trained models. We designed a 2,000-stage classifier, where the number of stages used for classification can be adaptively controlled depending on the content of the search window, and can be implemented with a more modular hardware, compared to classification with SVM and DPM [9-12]. Embodying these unique features for comprehensive object detection, an integrated accelerator chip was fabricated in 65nm CMOS to demonstrate real-time programmable object detection. Multi-class object detection is illustrated in Fig. 1, including the measurement results (localized objects) from the prototype chip. Power consumption is further optimized through configurable search stride and re-use of integral computation results for overlapping search windows.

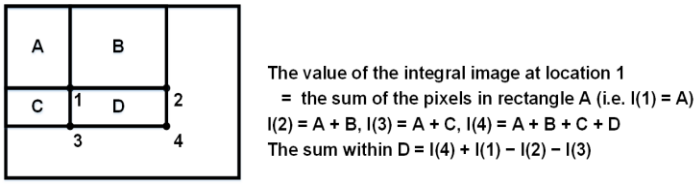
This paper is organized as follows. Section II explains the HeadHunter algorithm in detail. We introduce the system architecture of the proposed hardware accelerator including detailed features of main modules and hardware optimization techniques in Section III. Section IV presents the proposed algorithm adaptations that were employed to improve the hardware efficiency. The chip implementation and evaluation results are described in Section V. We conclude this paper in Section VI.

II. Overview of the Object Detection Algorithm (HeadHunter Model)

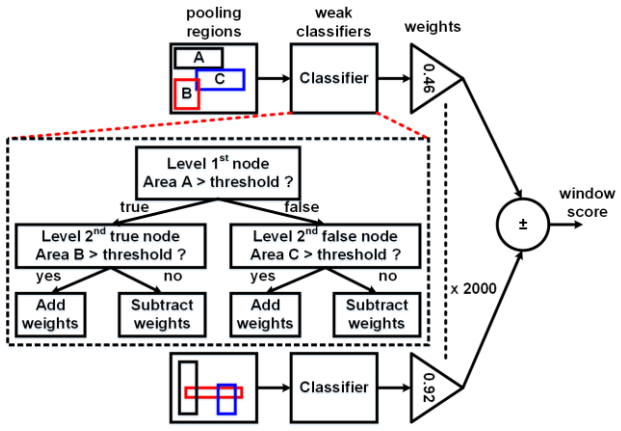
A HeadHunter model is proposed in [3] using a small set of rigid templates (i.e., without deformable parts), which reported state-of-the-art face detection accuracies on AFW [17], FDDB



(a)



(b)



(c)

Fig. 2. (a) Multiple channel features with six HOGs, a gradient magnitude, and LUV color space. (b) The concept of fast computation for an area sum using integral image. The figure is inspired by [1]. (c) Conceptual operation of 2,000 weak classifiers.

[18], and Pascal VOC [19] datasets. This model has four main features: (1) using multiple channels including 7 HOG channels and LUV color channels, (2) employing integral channel detector for fast feature computation, (3) 2,000 Adaboost weak classifiers containing shallow boosted trees of depth two (three stumps per tree), and (4) combining a set of rigid templates instead of using a single template per object category.

1) *Multi-Channel features:* Fig. 2(a) shows the multiple channel features employed in the HeadHunter algorithm, including LUV color channels and 7 HOG features (1 gradient magnitude and 6 quantized orientations). Features are extracted from the input image using integral pixel computation, as shown in Fig. 2(b). Reference [3] reported that the color channel information improves detection accuracy compared to the case of only using HOG channels, since certain objects (especially faces) have a discriminative color distribution. In addition, [20] showed that LUV color channels improved better accuracy comparing to other color channels such as grayscale, RGB, and YUV.
2) *Integral channel detector:* The use of an integral image as summed area table was first proposed in Viola-Jones algorithm [1]. This idea is examined by the integral channel feature framework in [20]. Integral data at (x,y) represents the sum of all the pixels above and to the left and then any rectangle features can be computed very rapidly using an intermediate representation for the image, as shown in Fig. 2(b).
3) *Adaboost weak classifiers:* A number of weak classifiers can be boosted to build a strong classifier. In this work, we employ 2,000 Adaboost weak classifiers for a robust system inspired by [3]. Fig. 2(c) shows the concept of the classifier operation. The 2,000 weak classifiers use pooling over rectangular regions as features. Each weak classifier computes this pooling operation and the 1st node compares with a given threshold to decide which of the two 2nd nodes should be computed. Depending on the 2nd node result, the weight corresponding to the classifier is either added or subtracted from the final score. After computing 2,000 weak classifiers, the final score is compared with a configurable threshold to determine if the search window has an object.
4) *Rigid boosted templates:* A rigid template approach can achieve high-speed object detection, but less detection accuracy, compared to DPM which has high computational cost [15]. HeadHunter model combined a small set of rigid templates that are separately used to capture intra-class diversity of objects, which can be boosted to build a strong detector. In our proposed hardware accelerator, we can use up to five different templates due to a limit on the on-chip memory size.

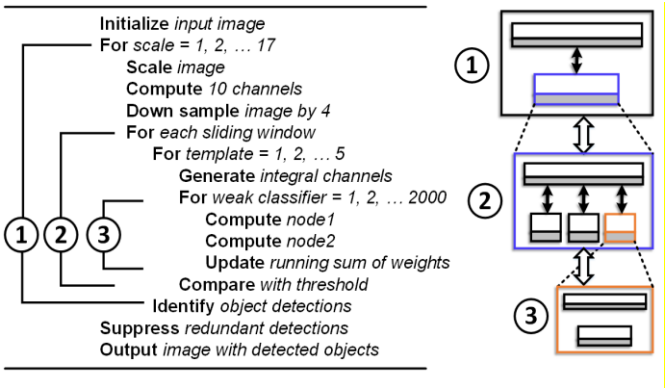


Fig. 3. High-level pseudo-code of the overall object detection operation (left) and corresponding modular nested structure on hardware (right).

The training dataset employed for face detector is the AFLW dataset [21], from which cropped faces are used as positive samples. For negative samples, random images from the Pascal VOC dataset [19] that do not have any person were used. The other training datasets such as traffic sign data, car license plate, and pedestrian are collected and labeled by the authors in a custom manner. During the training procedure, the object detection model first randomly generates a large feature pool and selects the best weak classifier on samples, and then increases the weight for difficult samples in each round. After all the stages of the detector are generated, it further collects the difficult negative samples to perform bootstrap training.

Each weak classifier contains a two-level decision tree for each of the five trained models: one frontal object model, two side views and two mirrored models. The input image is first scaled with scaling factors ranging from 0.2× to 3× to enable detection of various sizes of objects. All five trained models are evaluated separately for a sliding window that sweeps the entire image. The outputs of all weak classifiers are combined and compared with a threshold to allocate the bounding box for an object along with a score. The bounding boxes from all the scales are passed through a non-maximum suppression (NMS) stage, which selects one box with the highest score, and removes other redundant overlapping ones. High-level pseudo-codes of the object detection algorithm that we implement and the modular hardware structure are shown in Fig. 3.

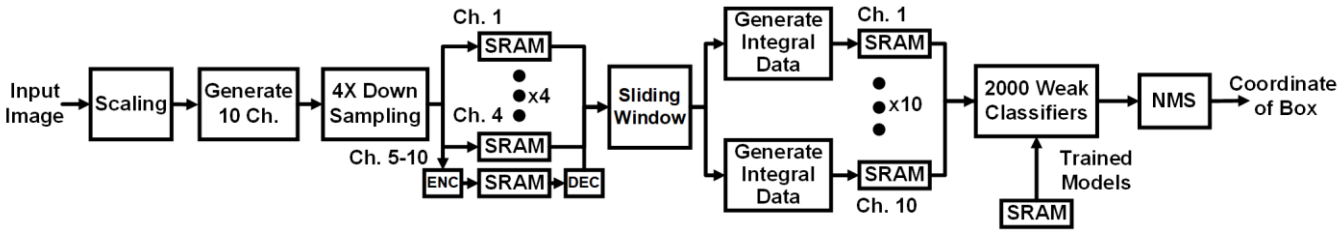


Fig. 4. Top-level block diagram and the end-to-end data flow of proposed object detection accelerator.

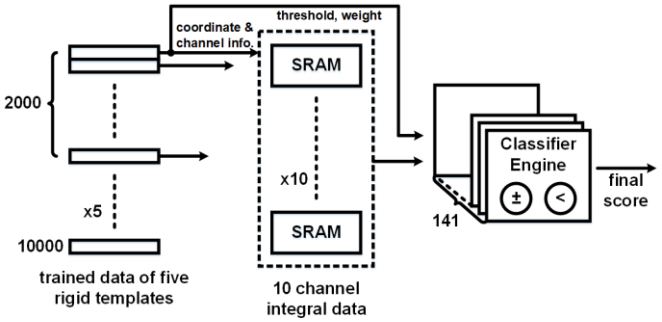


Fig. 5. Block diagram and data flow of classifier operation.

III. Energy Efficient Hardware Architecture Based on Rigid Boosted Templates

A. Hardware Architecture and Operation

Fig. 4 shows the top-level block diagram and data flow for the model architecture in Fig. 3. To achieve high accuracy, the classifier has five trained models, each with 2,000 weak classifiers, which can consume significant time and energy in the model evaluation.

1) Scale function

We use the search window size of 80×80 pixels, and detect objects of various sizes by scaling the input image from 0.4× to 2.0×, with a step size of 0.1×. Bilinear interpolation method is used to cover such wide range of scales. Each pixel in the scaled image is computed from four pixel values in the input image, which are stored into on-chip frame buffer. A 3×3 Gaussian smoothing filter is applied on the scaled image using three line buffers. Note that we support up-scaling up to 2.0× for robust detection, which makes the SRAM size to be 186.5KB, a 3.7× increase compared to the case of only supporting down-scaling.

2) Channel generation

This method uses 10 feature maps consisting of seven HOG channels (1 gradient magnitude and 6 quantized orientations) and LUV color space channels. The quantized orientation of HOG is a weighted histogram where the gradient angle and magnitude determine the bin index and the weight, respectively, as shown in the following equation:

$$Q_\theta(x,y) = G(x,y) \cdot 1[\Theta(x,y) = \theta], \quad (1)$$

where $G(x,y)$ and $\Theta(x,y)$ are the gradient magnitude and quantized gradient angle, respectively, at $I(x,y)$ [20].

Piecewise linear approximation is used for complex non-linear computations such as square and cube root. 7-bit precision is used for channel data. Channels are then down-sampled by 4 and stored in SRAM blocks. To reduce the on-chip memory size, we propose a compression method for six HOG features, such that we reduce the number of SRAMs from 10 to 5 SRAMs (details in Section III-B). Note that all processes such as generating, down-sampling, storing, and loading for 10-channel feature data are executed in parallel.

3) Integral function

Integral images defined over the 10 channels are used for fast summation over random rectangular pooling regions. A key concern of the integral function scheme in terms of hardware implementation is that a huge memory is needed to store integral data. For example, we need a SRAM size of 234.4KB for 8-bit precision data of QVGA (320×240) image to store an entire of integral data. To reduce the memory size, we propose that integration is performed over 12×10 windows and integral data are stored within 160 (whole horizontal pixel)×32 size, instead of an entire size of 160×120 (details in Section III-B).

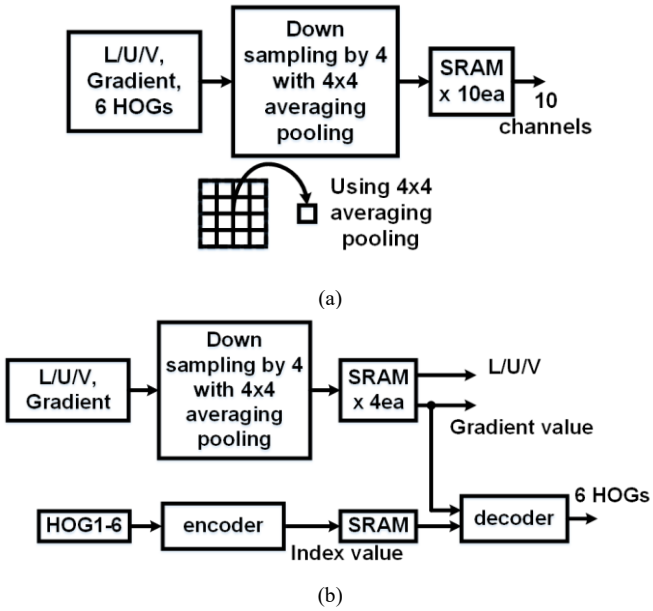


Fig. 6. Illustration of the down-sampling and storage of generated channel data in (a) the baseline scheme and (b) the proposed scheme.

4) Classifier operation

Fig. 5 shows the block diagram and data flow of the classifier operation. The trained data of five different templates, each with 2,000 weak classifiers are stored in SRAM. One of 10 SRAMs that store 10-channel integral data is selected by the channel information given by the trained data, which means that the 10-channel integral data should be ready altogether and be accessible from the 10 SRAMs. The two row data in the selected SRAM are loaded according to the coordinate information from the trained data to use pooling over rectangular regions as the feature. A Classifier Engine (CE) computes the area of the rectangular region, and adds or subtracts weights according to the results by comparing the area with a threshold value given by the trained model. One hundred forty one CE modules compute the weak classifier for all horizontal search windows in parallel. After computing five rigid templates, the classifier operation is iterated over different vertical locations. During the detection process, all five templates are evaluated over each search window and their results are combined using NMS.

5) NMS function

Multiple scales, sliding windows and five different templates result in a cluster of detections around a single object. NMS method is used to select the best detection and remove the redundant ones. In this work, we decided the maximum of detectable objects per image to be 50, balancing the NMS computation time. All detection results are sorted based on their scores. If the overlap is greater than a 0.3 (adopted from [3]),

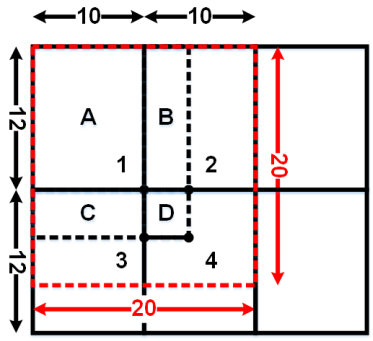


Fig. 7. Illustration of obtaining correct integral data over 20×20 window with 12x10 window of integration.

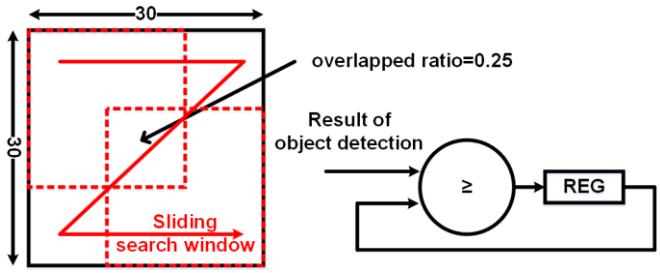


Fig. 8. Pre-processing step for the NMS function is illustrated. The largest detection result is stored at local registers while sliding the search window within 30×30 pixels, such that 120 detection results that have overlap greater than a 0.25 are suppressed.

then the detection was suppressed. After sorting the values from all scales and templates, post-NMS result is used as the final bounding box of the detected object in the image.

B. Hardware Optimization Techniques

We propose an adaptive pooling scheme when we perform down-sampling by 4 after channel generation in order to reduce SRAM size. The baseline algorithm [3] adopted 4×4 average pooling for the down-sampling and 10 channels are stored into SRAM as shown in Fig. 6(a). As illustrated in Fig. 6(b), we proposed a compression technique for six HOG values for the accelerator. Based on (1), the six HOG channel values are the gradient magnitude value or zero according to the quantized gradient angle. In other words, one of six HOGs is non-zero while the rest of the five HOGs are zero at the same pixel location. Based on this HOG feature, the six HOG values can

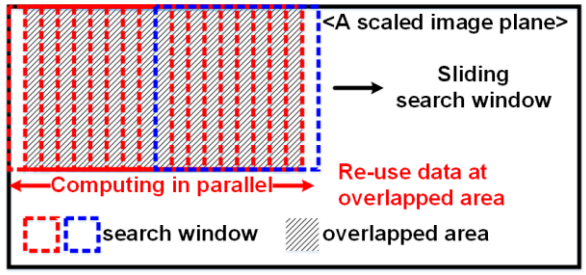


Fig. 9. Data re-use and parallel computing scheme for multiple adjacent search windows.

be replaced to the index value indicating the non-zero HOG channel after down-sampling, following the computation in (2).

$$Index(x,y) = arg\max_{j}\left[\sum_{x,y=1}^{4} HOG_j(x,y), j = 1 \ldots 6\right] \quad (2)$$

The other four channels are down-sampled by 4 with average pooling. The index value and the data of four channels are then stored at SRAM. This reduces the SRAM size for storing channel data by ~2× without any degradation of accuracy. The data of 6 HOG channels can be reproduced through the decoder with index value from SRAM, as described in (3):

$$HOG_j(x,y) = G(x,y) \cdot 1[Index(x,y) = j] \quad (3)$$

where *G(x,y)* and *Index(x,y)* are the gradient magnitude and the index value, respectively, at *I(x,y)*.

In addition, to reduce the number of bits in the integral data, integration is performed over 12×10 windows. When pooling over a 20×20 window, the offset from the previous integral window is added to get the correct result. An example is illustrated in Fig. 7. We can obtain the correct integral data at location 4 with three appropriate offset values at location 1, 2, and 3. The values of the integral image at location 1, 2, 3, and 4 are the sum of the pixels in rectangle A, B, C, and D, respectively. The correct integral data at location 4 for 20×20 window can be computed as A+B+C+D. By using a window size of 12×10 for generating integral channel data, the number of bits used for integral data is reduced to 14 bits (22 bits are

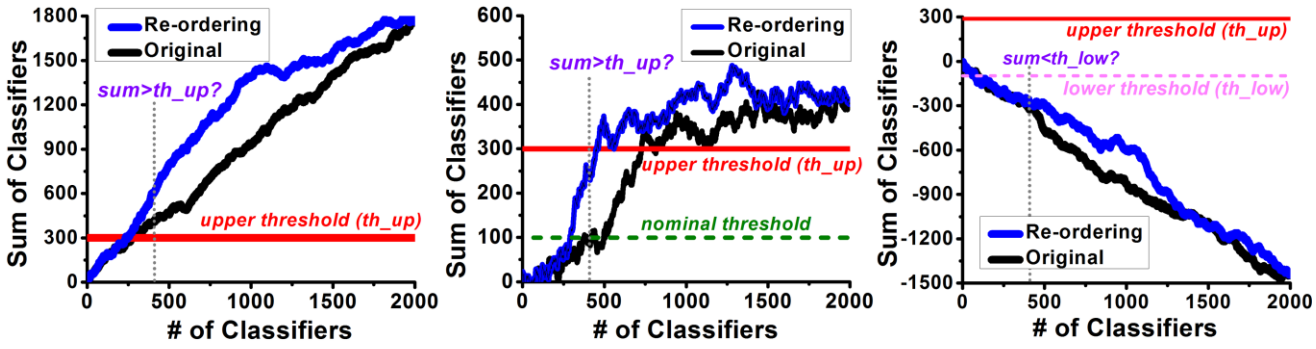
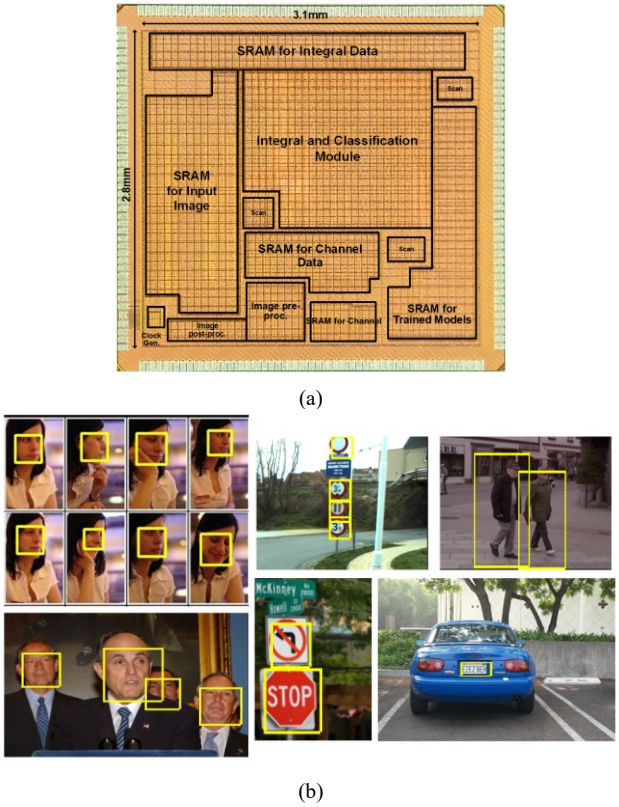


Fig. 10. Weight reordering and adaptive classification. If the intermittent sum is larger than upper threshold (left) or smaller than lower threshold (right), the remaining classifier operations are skipped. Otherwise, 2000 classifiers are computed (middle).



(a)

(b)

Fig. 11. (a) 65nm prototype chip micrograph. (b) Chip measurement results of multi-scale multi-object detection on face and traffic sign images.

required when integrating over the entire image), reducing the SRAM size by 36%.

Furthermore, a pre-processing step for the NMS function was introduced. There are 17 scales to process and each scale has a very large number of search windows that produce object detection results. To alleviate the large memory requirement to store such many results, while sliding the search window in each scaled image, we directly remove redundant boxes of the detected object within specific ranges as shown in Fig. 8. This reduces the computation time and SRAM size for NMS function by 14-89× depending on the pixel stride (1-3). To simplify the computation, we determined the fixed overlap ratio threshold for each scaled image called *intra-scale* overlap threshold to be a value (0.25) that minimally degrades AP based on our experimental results. On the other hand, after completing the pre-processing of NMS for the entire 17 scales, we perform

TABLE I
CHIP SPECIFICATIONS

Technology	65nm CMOS
Chip size	3.6×3.3 mm^2
Core size	3.1×2.8 mm^2
SRAM	339.9KB
Frame buffer	225KB (SRAM)
Input resolution	1920×1080
Supply voltage	0.58 – 1.1 V
Clock frequency	100 – 250 MHz
Frame rate	20 – 50 fps
Power	22.5 – 181.7 mW
Energy	0.54 – 1.75 nJ/pixel

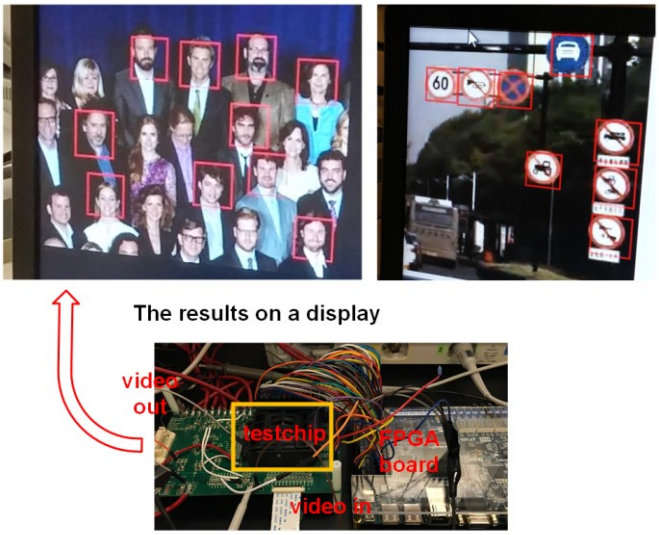


Fig. 12. System test environment.

NMS function to remove overlapping detection boxes with a configurable *inter-scale* overlap threshold parameter.

Finally, instead of computing different weak classifiers in parallel, we compute a single weak classifier across multiple windows in parallel. As shown in Fig. 9, this re-uses data that are overlapped among adjacent search windows, reducing the number of memory access by 77× in average for 17 scales.

IV. ALGORITHM ADAPTATIONS FOR HARDWARE EFFICIENCY

As described in Section II, HeadHunter model based on a set of rigid templates with Adaboost weak classifiers can be implemented with a more modular hardware. We employ five rigid templates in our hardware accelerator and have five trained models for face detection. On the other hand, we only have one trained model for other object classes, such as traffic sign, car license plate, and pedestrian. We propose a multi-class object detection method using five rigid templates. When using five different types of trained models for different object classes through five rigid templates, we can detect up to five different object classes simultaneously. Since we can use five different rigid templates for different types of object classes instead of using a set of rigid templates for single object class, the proposed method can detect multiple object classes at the same time without any hardware redundancy, in contrast to [9, 12]. The architectures in [9, 12] have two classifier engines to detect two object classes. In this work, since we have only four types of trained models for face, traffic sign, car license plate, and pedestrian, we are capable of detecting four object classes at the same time.

In addition, we employ 2,000 Adaboost weak classifiers to build a strong classifier similar to [3]. The experimental results in [3] described that 83.35% and 85.57% average precision were obtained with 200 weak classifiers and 2,000 weak classifiers, respectively. To reduce the computation load from the large number of weak classifiers with less degradation in the detection accuracy, we propose two efficient techniques: adaptive cascading and weight re-ordering, as shown in Fig. 10. Adaptive classifier cascading is proposed to dynamically scale

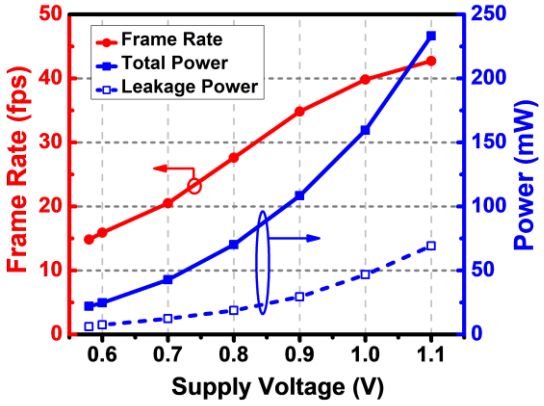


Fig. 13. Measured frame rate and total/leakage power with voltage scaling.

the amount of classifier computation based on input images. We intermittently check the sum of classifiers with a configurable subset of 2,000 classifiers (e.g., 400 as shown in Fig. 10) whether it is higher than a conservative upper threshold or smaller than a lower threshold value, in which case the true or false object detection result is determined without going through 2,000 classifiers. After going through a subset of classifiers, if the intermediate result in a search window is strongly positive or negative compared to the object threshold, the remaining classifier operations are skipped. In weight re-ordering, based on our proposed adaptive classifier cascading scheme, the weak classifiers with higher weight values are computed first. This helps the intermediate result to reach a strongly positive or negative value earlier, and therefore we can expedite the detection of an object. The proposed techniques achieved 5.5× speed-up while having less than 1% degradation in the average precision.

Furthermore, we employed a number of configurable parameters in the algorithm and the implemented hardware, in order to show the trade-offs of performance/accuracy and power. These include (1) the number of different scales (up to 17) and various scale factors (0.4× to 2.0× with as low as 0.1× step), (2) programmable horizontal and vertical stride (1-3 pixels) for the sliding search window, (3) threshold for object classification, and (4) variable *inter-scale* overlap ratio for NMS (0.25-0.55).

TABLE II
POWER BREAKDOWN WITH VARIOUS CONFIGURATIONS

	Config1	Config2	Config3	Config4
Number of scales	17 (0.4-2.0×)	8 (0.4-1.8×)	8 (0.4-1.5×)	8 (0.4-1.5×)
Pixel stride [1]	1,2,3	1,2,3	1,2,3	Max
Adaptive stage [2]	500	500	400	400
Logic power [3] (a)/(b) (mW) @ 1.0V	20/120	20/110	20/93	20/85.5
SRAM power [4] (a)/(b) (mW) @ 1.0V	20/48	20/43	20/37	20/34
Total power (mW) @ 1.0V	215	193	170	159.5
Frame rate (fps)	10.7	22.7	30.3	39.5

[1] (1,2,3): pixel stride pre-configured 1 - 3 based on scale (small→large)
(Max): pixel stride is 2 for horizontal, 3 for vertical
[2] Classification stage when sum is compared with upper/lower threshold
[3,4] (a): pre-processing of image, (b): integral and classification processing

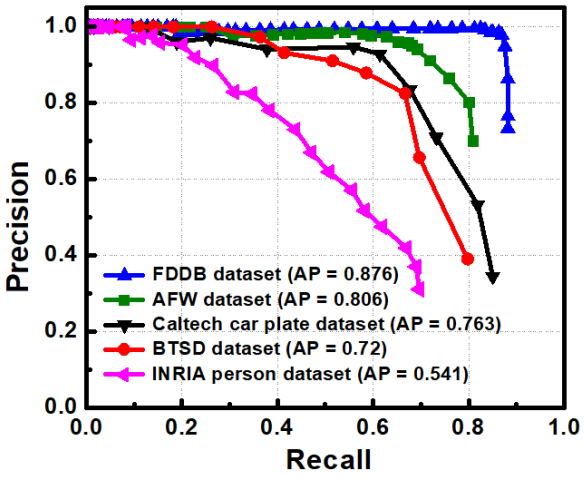


Fig. 14. Measured precision[1] versus recall[2] curve with multiple object classes.
[1]Precision: (true positive) / (true positive + false positive)
[2]Recall: (true positive) / (true positive + false negative)

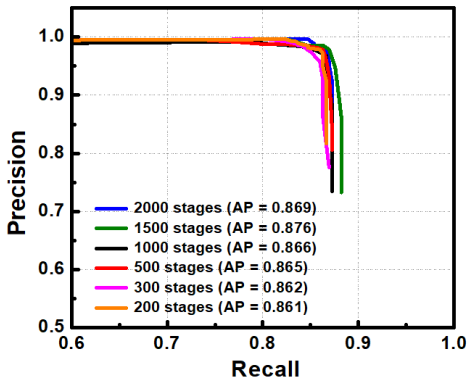


Fig. 15. Precision-recall curves on the FDDB datasets for the different number of weak classifiers in our proposed adaptive classifier cascading

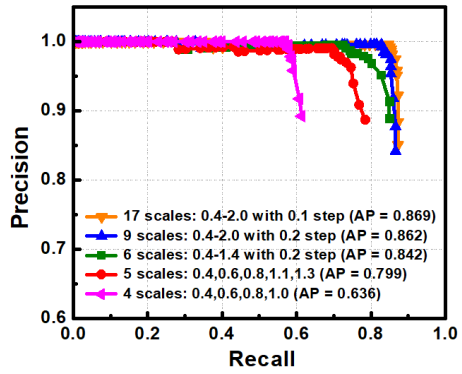


Fig. 16. Precision-recall curves on the FDDB datasets for the various number of scale factors.

V. 65NM IMPLEMENTATION RESULTS

The proposed ASIC accelerator was implemented in 65nm CMOS. The chip micrograph is shown in Fig. 11(a), where the total area is 3.1×2.8 mm^2, including the input image buffer. Fig. 11(b) shows the output of the prototype chip that demonstrates

TABLE III
DELAY TIME, POWER, ENERGY VERSUS DIFFERENT NUMBER OF STAGES IN ADAPTIVE CASCADING

	200 stages	300 stages	500 stages	1000 stages	1500 stages	2000 stages
Delay time (ms)	22.9	28.5	39.8	63.6	98.3	127.1
Power (mW) @ 1.0V	156.6	170.5	172.6	184.3	185.2	187.1
Energy (nJ/pixel)	1.73	2.34	3.31	5.65	8.78	11.47

multi-scale multi-object detection for face, traffic sign, car license plate and pedestrian detection, where bounding boxes (measured chip outputs) are drawn on top of the input image to localize the detected objects. The chip specifications are summarized in Table I.

Fig. 12 shows the prototype chip measurement environment and system that was used to evaluate real-time object detection. It is composed of the custom PCB that mounts the 65nm prototype chip, a FPGA board, a HDMI interface board, and a LCD display. Our prototype chip performs end-to-end object detection, where it takes an input video data and outputs the video data enclosing a detected object with a final bounding box. All the image processing and computations for object detection are done in the prototype chip. We only use the FPGA board to configure the chip and read information of detected object such as coordinate and score to evaluate the accuracy.

Note that we down-sampled higher-resolution images (up to full HD 1920×1080) to QVGA (320×240) to store an entire of input frame image in the on-chip frame buffer instead of using external storage such as DRAM. Then, on-chip QVGA input frame buffer was used to scale images on-the-fly for 17 scales and iteratively compute the same sliding window. In other words, our chip demonstrated object detection for full HD resolution images with down-sampling as a pre-processing step. Since a down-sampled pixel is only read in the pre-processing step while the full HD videos is transmitted in a row raster scan order, no extra process such as interpolation is required. An alternative would be to use a single-size image and scale the sliding window for a number of scales. This method will have a smaller on-chip frame buffer, but will require a larger memory for trained models that increases with the number of scale factors. Performing a fine-grain search on a lower-resolution image is more favorable than a coarse-grain search on a high-resolution image, due to the reduction in image sensor power and data communication.

To characterize the object detection accuracy, performance, and power consumption, we used the AFW and FDDB database [17, 18] for face detection, the BTSD database [22] for traffic sign detection, Caltech database [23] for car license plate detection, and INRIA database [24] for pedestrian detection. The measured chip performance (frames per second) and total/leakage power consumption with dynamic voltage scaling are shown in Fig. 13. Full object detection functionality was verified down to 0.58V, where the chip performs real-time detection at 20.1 fps with 22.5mW power. In Table II, the power breakdown in logic and memory at the nominal voltage as 1.0V is detailed for four different chip configurations, where the number of scales, pixel stride, and the classification stage are

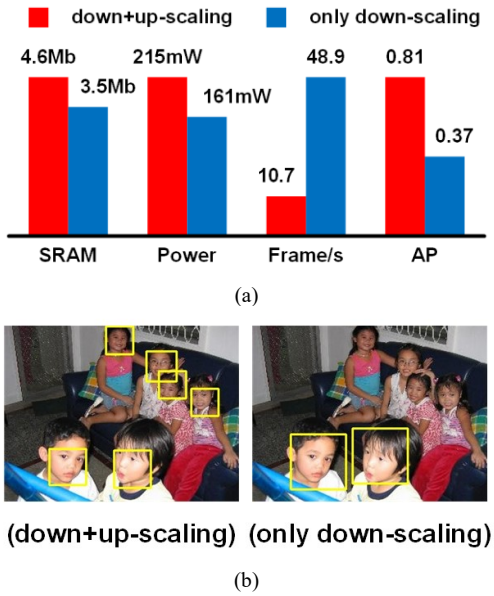


Fig. 17. (a) Design comparison using up-/down-scaling. (b) Measurement results show that smaller faces can be detected through up-scaling.

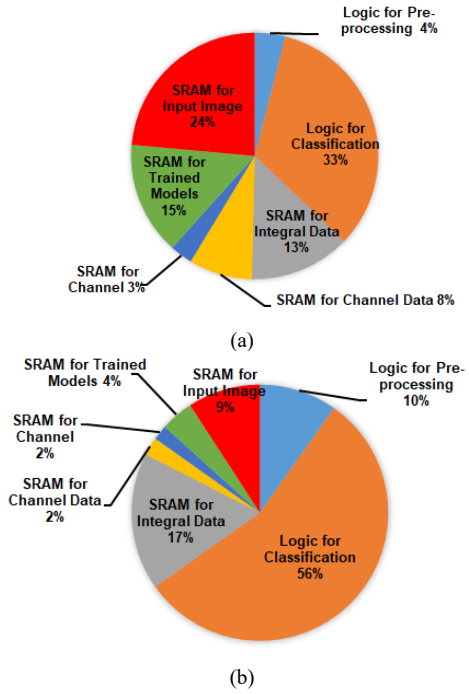


Fig. 18. (a) Area and (b) power breakdown of the overall system.

TABLE IV.
COMPARISON TO PRIOR ASIC WORKS ON OBJECT DETECTION

	Based on hand-crafted features				Based on CNN-learned features			This work
	[9]	[10]	[11]	[12]	[27]	[28]	[29]	
CMOS Tech.	65nm	40nm	45nm SOI	65nm	65nm	65nm	55nm	65nm
Chip size	3.3×1.2 mm^2	2.58×2.27 mm^2	2.8×0.96 mm^2	3.58×3.58 mm^2	4×4 mm^2	3.8×3.8 mm^2	3.3×3.1 mm^2	3.1×2.8 mm^2
Image resolution	Full HD	HD	Full HD	Full HD	224×224	448×448	416×416	Full HD
Channel Feature	9 HOG	1	9 HOG	9 HOG	RNN-FIS	Yolo V2	Yolo V2/ Yolo tiny	7 HOG + 3 LUV
# of scales	single	single	12 (all down)	12 (all down)	single	single	single	17 (6 down, 11 up)
Classifier	SVM	22-stage cascade	SVM	SVM, DPM	RNN	CNN	CNN	2000-stage cascade
Object classes	2	1	1	2	-	-	-	4
Accuracy	F_1=95% (GTI [26])	F_1=93% (custom dataset)	AP=0.37 (INRIA)	AP=0.26 (VOC 2007)	-	-	AP=0.596 (VOC 2007, 2012)	AP=0.88 (FDDB), 0.81 (AFW), 0.72 (BTSD), 0.76 (Caltech), 0.54 (INRIA)
Frame rate	30 fps	5.5 fps	60 fps	30-60 fps	30 fps	12.05 fps	5.5/27.7 fps	20–50 fps
Power	84mW (@0.7V)	23mW (@0.6V)	45.3mW (@0.72V)	58.6-216.5mW (@0.77-1.11V)	330mW (@1.2V)	280mW	68mW (@1.1V)	22.5–181.7mW (@0.58–1.1V)
Energy	1.35 nJ/pixel	4.5 nJ/pixel	0.36 nJ/pixel	0.94-1.74 nJ/pixel	15.3 nJ/pixel	116.6 nJ/pixel	77.37/14.2 nJ/pixel	0.54-1.75 nJ/pixel

varied to check intermediate sum for adaptive cascading. With regards to voltage scaling, the power/energy values in Table II and Table III also scales down in a similar manner that is reported in Fig. 13.

Fig. 14 shows the precision versus recall (PR) curves [25] of the prototype chip measured for AFW, FDDB, BTSD, Caltech car license plate, and INRIA person datasets. The average precision (AP) can be computed as the area under the PR curve. We achieved AP of 0.876 and 0.806 for the FDDB and AFW datasets for face detection, respectively. For traffic sign detection, we achieved AP of 0.72 for the BTSD dataset. We achieved AP of 0.763 for the Caltech dataset for car license plate detection. For pedestrian detection, we achieved AP of 0.541 for the INRIA dataset. Since our proposed system supports input image resolution up to full HD (1920×1080) with down-sampling into QVGA (320×240) as a pre-processing step, images that are over full HD size in the AFW and BTSD datasets are cropped to full HD size. However, note that we used the original annotation data of AFW and BTSD datasets in our AP measurements. In other words, we counted the number of objects that were not detected as false negatives due to truncated or lost objects after cropping the images.

Fig. 15 shows the AP values for the FDDB datasets with various stage number when using our proposed adaptive classifier cascading methods. We achieved AP of 0.862 with 200 stages in the adaptive cascading scheme, which is only 0.85% degradation in the average precision comparing to the AP of 0.869 with 2,000 stages (i.e., without the adaptive cascading scheme). Note that this AP degradation represents a ~3× reduction (0.85% vs. 2.22%) compared to the experimental results of [3]. In addition, the AP measured results with the detection quality versus number of scale factors is shown in the Fig. 16. We achieved the similar accuracy as AP of 0.862 in the nine scale factors from 0.4× to 2.0×, with a step size of 0.2×, comparing to the all (17) scale factors. For the six scale factors, we achieved AP of 0.843 in the FDDB dataset with a small amount of degradation. However, the AP value decreased somehow when using the five scale factors, and especially, there is significant deterioration in the four scale factors using only down-scaling. Table III summarized the measurement results of delay time, power, and energy with the different number of weak classifier stages in our proposed adaptive cascading technique. Comparing to our system without adaptive cascading skill, the proposed adaptive cascading method with 200 weak classifier stages reduced the total delay time of system by 5.5× and achieved 16.3% power reduction. Our proposed accelerator using adaptive cascading method reduces the overall system energy consumption by 6.6×.

Furthermore, through 2× up-scaling, our design can detect objects as small as 40×40 pixels, which is much smaller than the detectable objects in previous works [9-12]. Fig. 17 shows the comparison between the designs when only down-scaling (0.4-1.0×) was used and when both up-scaling and down-scaling (0.4-2.0×) are used. Up-scaling improves the AP significantly at the expense of moderate memory/power increase.

Fig. 18 shows the area and measured power breakdown of the prototype chip. 63% of the total chip area is occupied by on-chip SRAM arrays, due to the requirement to store the trained models, integral data, input image frame buffer, etc. On the other hand, 66% of the total chip power was consumed by logic components due to high activity factors, where the power of the

classifiers (56% of chip power) dominated.

Table IV shows the comparison with hand-crafted features based object detection accelerators [9-12]. The architecture in [10] achieved low power consumption similar to this paper, but the energy per pixel value is much higher than this work due to the lower image resolution and frame rate. The implementation in [11] achieved low energy per pixel number with high frame rate, but it was post-layout results. In addition, the accuracy in [11] is lower than this paper. Two object detection accelerators are presented in [9] and [12]. Both accelerators process full HD videos in real-time and support multiple object detection similar to this work. However, our proposed accelerator employs color based LUV channels and fine-grained up-scaling, which increase the detection accuracy and robustness, while achieving 60% and 42.5% energy/pixel reduction compared to [9] and [12], respectively. Note that our work evaluated AP across multiple datasets for multiple object classes, the most among any prior works [9-12]. In addition, our proposed accelerator is compared with CNN-learned feature based object detection accelerators [27-29]. Note that we calculated the energy per pixel numbers based on the energy efficiency numbers in [27-29]. The reference [27] proposed an advanced driver-assistance system (ADAS) processor that achieved 0.862 TOPS/W with a 4-layer recurrent neural network (RNN) connected to a fuzzy inference system (FIS), but the energy per pixel value is 28× higher than our accelerator. Two CNN processors for object detection are presented in [28-29]. Both processors implemented YOLO CNN [30], which is a representative end-to-end object detection CNN model. As a reconfigurable hybrid-NN processor, Thinker [28] achieved 1.26 TOPS/W for YOLO V2, but the energy consumption per pixel of our work is 216× less than that of Thinker. The CNN design in [29] achieved high energy efficiency of 2.2 TOPS/W and good accuracy of 0.6 mAP for VOC 2007 and VOC 2012 [19] datasets. However, due to the lower image resolution (416×416) the energy per pixel is 143× and 26× higher than our proposed work.

VI. Conclusion

In this paper, we presented a 65nm accelerator for real-time programmable object detection. The accelerator employed HeadHunter model based on a set of five rigid templates with 2,000 Adaboost weak classifiers. A large number of classifiers are used to make a strong object classification, and adaptive cascade is realized for dynamic computation scaling. High average precision of 0.88, 0.81, 0.76, 0.72 and 0.54 was achieved in FDDB, AFW, Caltech car plate, BTSD, and INRIA person datasets, respectively, by using integral channel features on 7 HOG and 3 LUV channels, 17 scale factors with 6 down-scaling and 11 up-scaling, configurable thresholding, adaptive cascading classification, and optimal non-maximum suppression. The accelerator achieved 0.54/1.75 nJ/pixel while consuming 22.5/181.7 mW at 0.58/1.1V with 20/50 fps in full HD videos, respectively. The hardware optimization techniques reduced on-chip SRAM size by overall 2.9×. Our proposed adaptive classifier cascading method achieved an overall 6.6× energy per pixel reduction. The capability of programmable and voltage-/performance-scalable many-object detection will enhance smart vision processors in ubiquitous mobile systems.

References

[1] P. A. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[2] P. Felzenszwalb, et al., "Object detection with discriminatively trained part based models," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627-1645, Sep. 2010.

[3] M. Mathias, et al., "Face detection without bells and whistles," *European Conf. in Computer Vision (ECCV)*, 2014.

[4] H. Li, et al., "A convolutional neural network cascade for face detection," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[5] S. Yang, et al., "Faceness-Net: Face Detection through Deep Facial Part Responses," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 8, pp. 1845-1859, Aug. 2018.

[6] R. Ranjan, V- M. Patel, and R. Chellappa, "HyperFace: A Deep Multi-Task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, pp. 121-135, Jan. 2019.

[7] R. Benenson, et al., "Pedestrian detection at 100 frames per second," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] S. Advanim, et al., "A scalable architecture for multi-class visual object detection," *Int. Conf. Field Programmable Logic and App. (FPL)*, 2015.

[9] K. Takagi, et al.,"A real-time scalable object detection system using low-power HOG accelerator VLSI," *J. Signal Process. Syst.*, vol. 76, no. 3, pp. 261-274, Sep. 2014.

[10] D. Jeon, et al., "A 23mW face recognition accelerator in 40nm CMOS with mostly-read 5T memory," *IEEE Symp. on VLSI Circuits*, 2015.

[11] A. Suleiman and V. Sze, "An energy-efficient hardware implementation of HOG-based object detection at 1080HD 60 fps with multi-scale support," *Journal of Signal Processing Systems*, pp. 1-13, Dec. 2015.

[12] A. Suleiman, et al., "A 58.6 mW 30 Frame/s Real-Time Programmable Multiobject Detection Accelerator With Deformable Parts Models on Full HD 1920x1080 Videos," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 3, pp. 844-855, Mar. 2017.

[13] M. Kim, et al., "A Real-tim 17-Scale Object Detection Accelerator with Adaptive 2000-Stage Classification in 65nm CMOS," *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.

[14] A. Suleiman, et al., "Towards Closing the Energy Gap Between HOG and CNN Features for Embedded Vision," *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.

[15] A. Suleiman, Z. Zhang, and V. Sze, "A 58.6mW real-time programmable object detector with multi-scale multi-object support using deformable parts model on 1920×1080 video at 30 fps," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2016, pp. 1–2.

[16] Y.-H. Chen, et al., "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE International Solid-State Circuits Conference (ISSCC)*, 2016.

[17] X. Zhu and D. Ramanan, "Face detection, pose estimation and landmark localization in the wild," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[18] Jain, V., Learned-Miller, E.: FDDB: A benchmark for face detection in unconstrained settings. Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst (2010).

[19] M. Everingham, et al., "The PASCAL visual object classes challenge 2012 (VOC2012) results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[20] P. Dollar, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," *British Machine Vision Conference (BMVC)*, 2009, pp. 91.1-91.11.

[21] M. Köstinger, et al., "Annotated facial landmarks in the wild: a large-scale, real-world database for facial landmark localization," *IEEE Int. Conf. on Computer Vision Workshops*, 2011.

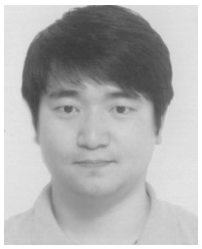
[22] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3D localisation," *Machine Vision and Applications (MVA)*, pp. 633-647, 2014.

[23] Caltech Car Dataset for Car License Plate Detection. http://www.vision.caltech.edu/archive.html

[24] INRIA Person Dataset for Pedestrian Detection. http://pascal.inrialpes.fr/data/human/.

[25] D. Jesse and M. Goadrich, "The relationship between precision-recall and ROC curves," *Int. Conf. on Machine Learning (ICML)*, 2006.

[26] GTI's Vehicle Image Database. http://www.gti.ssr.upm.es/data/Vehicle_database/Vehicle_database.html
[27] K. K. Lee, et al., "A 502GOPS and 0.984mW Dual-Mode ADAS SoC with RNN-FIS Engine for Intention Prediction in Automotive Black-Box System," *IEEE International Solid State Circuits Conference (ISSCC)*, 2016.
[28] S. Yin, et al., "A High Energy Efficient Reconfigurable Hybrid Neural Network Processor for Deep Learning Applications," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 968-982, Apr. 2017.
[29] X. Chen, et al., "A 68 mw 2.2 Tops/w low bit-width and multiplierless DCNN object detection processor for visually impaired people," *IEEE Transactions on Circuits and Systems for Video Technology*, Nov. 2018.
[30] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.



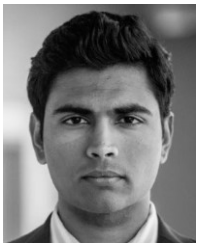
Minkyu Kim (S'15) received the B.E. degree in electronics and electrical engineering from Pusan National University, Busan, South Korea, in 2005, and the M.S. degree in electrical engineering from the Pohang University of Science and Technology, Pohang, South Korea, in 2007. He is currently pursuing the Ph.D. degree in electrical engineering with Arizona State University, Tempe, AZ, USA.

From 2007 to 2013, he was with LG Display Co. Ltd., Paju-si, South Korea, where he worked on the development of timing controller chip and image processing for high-quality display. He worked as a summer intern at Qualcomm Inc., San Diego, CA, USA, in 2017 and 2018. His research interests include efficient hardware design and application of machine learning and neuromorphic algorithms.

Mr. Kim was a recipient of the LG Display Scholarship from 2005 to 2007 and the Qualcomm Roberto Padovani Scholarship Award in 2017.

Abinash Mohanty (S'15) received the B.Tech. degree in electronics and communications engineering from the National Institute of Technology, Rourkela, India, in 2010. He is currently working towards Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA.

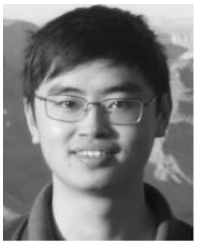
He was with Samsung Research & Development Institute, New Delhi, India, from 2010 to 2013, where he worked on embedded software platform for Smart devices. His research interests include reliability modeling, neuromorphic design for learning and stochastic design with extremely scaled CMOS.



Deepak Kadetotad (S'15) received the B.E. degree in electronics and communication engineering from the M.S. Ramaiah Institute of Technology, Karnataka, India, in 2013. He is currently working towards the Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA.

His research interests include hardware design and application of machine learning and neuromorphic algorithms.

Mr. Kadetotad was a recipient of the LSI Chairman's International Scholarship from 2009 to 2013.



Xiaofei He (SM'10) received the BS degree in computer science from Zhejiang University, China, in 2000 and the PhD degree in computer science from the University of Chicago, in 2005.

He is a professor in the State Key Lab of CAD&CG, Zhejiang University, China. Prior to joining Zhejiang University, he was a research scientist with Yahoo! Research Labs, Burbank, California. His research interests include machine learning, information retrieval, and computer vision.

Yu Cao (S'99-M'02-SM'09-F'17) received the B.S. degree in physics from Peking University, Beijing, China, in 1996, and the M.A. degree in biophysics and the Ph.D. degree in electrical engineering from University of California at Berkeley, Berkeley, CA, USA, in 1999 and 2002, respectively.

He worked as a summer intern at Hewlett-Packard Labs, Palo Alto, CA in 2000, and at IBM Microelectronics Division, East Fishkill, NY, in 2001. After working as a post-doctoral researcher at the Berkeley Wireless Research Center (BWRC), he is now a Professor of Electrical Engineering at Arizona State University, Tempe, Arizona. He has published numerous articles and two books on nano-CMOS modeling and physical design. His research interests include brain-inspired computing, hardware design for on-chip learning, and reliable integration of nanoelectronics.

Dr. Cao was a recipient of the 2012 Best Paper Award at IEEE Computer Society Annual Symposium on VLSI, the 2010, 2012, 2013, 2015 and 2016 Top 5% Teaching Award, Schools of Engineering, Arizona State University, 2009 ACM SIGDA Outstanding New Faculty Award, 2009 Promotion and Tenure Faculty Exemplar, Arizona State University, 2009 Distinguished Lecturer of IEEE Circuits and Systems Society, 2008 Chunhui Award for outstanding oversea Chinese scholars, the 2007 Best Paper Award at International Symposium on Low Power Electronics and Design, the 2006 NSF CAREER Award, the 2006 and 2007 IBM Faculty Award, the 2004 Best Paper Award at International Symposium on Quality Electronic Design, and the 2000 Beatrice Winner Award at International Solid-State Circuits Conference. He served as Associate Editor of the IEEE Transactions on CAD, and on the technical program committee of many conferences.

Jae-sun Seo (S'04–M'10–SM'17) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2006 and 2010, respectively.

From 2010 to 2013, he was with IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, where he worked on cognitive computing chips under the DARPA SyNAPSE Project and energy-efficient integrated circuits for high-performance processors. In 2014, he joined the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA, as an Assistant Professor. In 2015, he was with the Intel Circuits Research Lab as a Visiting Faculty. His current research interests include efficient hardware design of machine learning and neuromorphic algorithms and integrated power management.

Dr. Seo was a recipient of the Samsung Scholarship from 2004 to 2009, the IBM Outstanding Technical Achievement Award in 2012, and the NSF CAREER Award in 2017.