ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.15, No.12, 1–10

VLSI design of a power-efficient object detector using PCANet

Yuteng Zhou[2a)] **and Xinming Huang**[1,2b)]

[1] *School of Electronics and Information, Nantong University, 9 Seyuan Road, Nantong, Jiangsu, CHINA*

[2] *Electrical and Computer Engineering Dept., Worcester Polytechnic Institute, 100 Institute Rd, Worcester, MA, USA*

a) *ytchou@wpi.edu*

b) *xhuang@wpi.edu, Corresponding Author*

Abstract: This paper presents the hardware architecture and VLSI implementations of a PCANet-based object detector. The proposed PCANet model, cascaded with a linear support vector machine, can achieve better classification performance than traditional handcrafted computer vision methods, yet it is significantly more power efficient than multi-layer convolutional neural networks. The proposed pipeline hardware architecture, when implemented using Synopsys 32 nm process technology, results in 27.4 fps while processing 1080P, with only 0.5 watt power consumption. Targeted for the application of advanced driver assistance system, the proposed design is evaluated on road marking and traffic light dataset with an accuracy result of 96.8% and 93.1% respectively. Therefore, the proposed VLSI implementation of PCANet algorithm provides a high-throughput and power-efficient solution for object detection applications.

Keywords: PCANet, ADAS, low power, CNN, object detection, VLSI

Classification: Integrated circuits

References

[1] M. B. Jensen, *et al.*: "Vision for looking at traffic lights: Issues, survey, and perspectives," IEEE Trans. Intell. Transp. Syst. **17** (2016) 1800 (DOI: 10.1109/TITS.2015.2509509).

[2] Y. K. Kim, *et al.*: "Real time traffic light recognition system for color vision deficiencies," International Conference on Mechatronics and Automation (ICMA) (2007) 76 (DOI: 10.1109/ICMA.2007.4303519).

[3] A. De la Escalera, *et al.*: "Traffic sign recognition and analysis for intelligent vehicles," Image Vis. Comput. **21** (2003) 247 (DOI: 10.1016/S0262-8856(02)00156-7).

[4] X. Wen, *et al.*: "Efficient feature selection and classification for vehicle detection," IEEE Trans. Circuits Syst. Video Technol. **25** (2015) 508 (DOI: 10.1109/TCSVT.2014.2358031).

[5] D. M. Gavrila and S. Munder: "Multi-cue pedestrian detection and tracking from a moving vehicle," Int. J. Comput. Vis. **73** (2007) 41 (DOI: 10.1007/s11263-006-9038-7).

[6] M. Mody, *et al.*: "High performance front camera adas applications on ti's

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018

tda3x platform," IEEE 22nd International Conference on High Performance Computing (HiPC) (2015) 456 (DOI: 10.1109/HiPC.2015.56).
[7] T.-H. Chan, *et al.*: "PCANet: A simple deep learning baseline for image classification?" IEEE Trans. Image Process. **24** (2015) 5017 (DOI: 10.1109/TIP.2015.2475625).
[8] W. Qadeer, *et al.*: "Convolution engine: Balancing efficiency & flexibility in specialized computing," ACM SIGARCH Computer Architecture News **41** (2013) 24 (DOI: 10.1145/2485922.2485925).
[9] J. Bruna and S. Mallat: "Invariant scattering convolution networks," IEEE Trans. Pattern Anal. Mach. Intell. **35** (2013) 1872 (DOI: 10.1109/TPAMI.2012.230).
[10] Z. Huang, *et al.*: "Unsupervised domain adaptation for speech emotion recognition using PCANet," Multimedia Tools Appl. **76** (2017) 6785 (DOI: 10.1007/s11042-016-3354-x).
[11] S. Wang, *et al.*: "Human fall detection in surveillance video based on pcanet," Multimedia Tools and Appl. **75** (2016) 11603 (DOI: 10.1007/s11042-015-2698-y).
[12] B. Li, *et al.*: "A pcanet based method for vehicle make recognition," IEEE 19th International Conference on Intelligent Transportation Systems (ITSC) (2016) 2404 (DOI: 10.1109/ITSC.2016.7795943).
[13] Y. Xie, *et al.*: "Fully-parallel area-efficient deep neural net-work design using stochastic computing," IEEE Trans. Circuits Syst. II, Exp. Briefs **64** (2017) 1382 (DOI: 10.1109/TCSII.2017.2746749).
[14] S. Moini, *et al.*: "A resource limited hardware accelerator for convolutional neural networks in embedded vision applications," IEEE Trans. Circuits Syst. II, Exp. Briefs **64** (2017) 1217 (DOI: 10.1109/TCSII.2017.2690919).
[15] Z. Feng, *et al.*: "Dlanet: A manifold-learning-based discriminative feature learning network for scene classification," Neurocomputing **157** (2015) 11 (DOI: 10.1016/j.neucom.2015.01.043).
[16] Y. Gan, *et al.*: "Image classification with a deep network model based on compressive sensing," IEEE 12th International Conference on Signal Processing (ICSP) (2014) 1272 (DOI: 10.1109/ICOSP.2014.7015204).
[17] O. Pina-Ramirez, *et al.*: "An fpga implementation of linear kernel support vector machines," IEEE Internal Conference on Reconfigurable Computing and FPGA's, ReConFig (2006) 1 (DOI: 10.1109/RECONF.2006.307784).
[18] T. Groleat, *et al.*: "Hardware acceleration of svm-based traffic classification on fpga," 8th International Wireless Communications and Mobile Computing Conference (IWCMC) (2012) 443 (DOI: 10.1109/IWCMC.2012.6314245).
[19] V. Gokhale, *et al.*: "A 240 gops/s mobile coprocessor for deep neural networks," IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2014) 696 (DOI: 10.1109/CVPRW.2014.106).
[20] C. Farabet, *et al.*: "Hardware accelerated convolutional neural networks for synthetic vision systems," Proc. IEEE International Symposium on Circuits and Systems (ISCAS) (2010) 257 (DOI: 10.1109/ISCAS.2010.5537908).
[21] C. Farabet, *et al.*: "Neuflow: A runtime reconfigurable dataflow processor for vision," IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2011) 109 (DOI: 10.1109/CVPRW.2011.5981829).
[22] P.-H. Pham, *et al.*: "Neuflow: Dataflow vision processing system-on-a-chip," IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS) (2012) 1044 (DOI: 10.1109/MWSCAS.2012.6292202).
[23] L. Cavigelli and L. Benini: "Origami: A 803-GOp/s/W convolutional network accelerator," IEEE Trans. Circuits Syst. Video Technol. **27** (2017) 2461 (DOI: 10.1109/TCSVT.2016.2592330).

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018

[24] T. Wu and A. Ranganathan: "A practical system for road marking detection and recognition," IEEE Intelligent Vehicles Symposium (IV) (2012) 25 (DOI: 10.1109/IVS.2012.6232144).
[25] M.-M. Cheng, *et al.*: "BING: Binarized normed gradients for objectness estimation at 300 fps," Proc. IEEE Conference on Computer Vision and Pattern Recognition (2014) (DOI: 10.1109/CVPR.2014.414).
[26] T. Chen, *et al.*: "Road marking detection and classification using machine learning algorithms," IEEE Intelligent Vehicles Symposium (IV) (2015) 617 (DOI: 10.1109/IVS.2015.7225753).
[27] Z. Chen and X. Huang: "Accurate and reliable detection of traffic lights using multiclass learning and multiobject tracking," IEEE Intell. Transp. Syst. Mag. **8** (2016) 28 (DOI: 10.1109/MITS.2016.2605381).

1 Introduction

Recently, many industrial and academic research efforts have been focused on ADAS (advanced driver-assistance systems). ADAS usually needs a sophisticated fusion of sensors such as LiDAR, radar, and cameras. Among these sensors, optical cameras are most widely used because of their low costs and easy installation. Also, thanks to the rapid development of deep learning in computer vision, vision based algorithms have become more accurate and more robust in varied driving environments [1].

With the popularity of deep learning, a lot of vision based solutions for intelligent vehicles have been proposed, such as traffic light detection [2], traffic sign recognition [3], vehicle detection [4], and pedestrian detection [5]. However, few of these solutions can work in real time, which is very critical for intelligent vehicles [6]. On the other hand, traditional computer vision algorithms cannot handle the intra-class variability arising from varied lighting conditions, misalignment, occlusion and corruptions, and non-rigid deformations [7]. In our work, considering performance, throughput and power efficiency, we propose the PCANet as the baseline detector for vision based ADAS solutions.

In this work, we designed efficient hardware architecture for PCANet on digital VLSI circuits in 32 nm process technology. We successfully speeded up the PCANet algorithm to 27.4 fps at 1080P while consuming less than 0.5 Watt of power. In this way, we are able to provide a PCANet based single-chip solution for vision based ADAS applications. The main contributions of this paper are as follows:

1. Proposing the PCANet as a potential baseline object detector for computer vision systems where processing speed and performance are desired at the same time. The PCANet outperforms traditional feature based algorithms, and is faster on training and inference than CNNs.
2. Designing an efficient hardware architecture for PCANet, achieving 27.4 fps throughput at 1080P, while consuming only 0.5 watt. Our implementation beats typical CNN implementations such as ConvEngine [8] in both power efficiency and throughput.
3. Proposing a single-chip solution specifically suitable for ADAS applications. System integration has long been a tough task for ADAS. Unlike typical CNN

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018

chips, our chip design doesn't rely on external memory thus can be used as a go-to solution, largely reducing integration efforts.

The rest of this paper is organized as follows. Section 2 presents work related to the PCANet, such as the hardware implementation of a convolutional neural network. Section 3 derives the expression of the PCANet, illustrating how a PCA filter's coefficients are trained. Section 4 presents the hardware architecture of our PCANet implementation. Section 5 presents results of our chip design in Synopsys 32 nm process technology. Section 6 shows the performance of our PCANet chip on road marking detection and traffic light detection. Finally, Section 7 concludes this work and provides some future directions.

2 Related work

The idea of PCANet arises from wavelet scattering networks (ScatNet) [9], in which the convolutional filters are prefixed, needing no training at all. Since convolutional filters in a ScatNet are prefixed, the ScatNet does not generalize very well to tasks where intra-class variability includes illumination change and corruption [7], let alone vision based ADAS tasks. Adopting the simple architecture of ScatNet and the robust performance of multi-layer CNNs, PCANet is fast to train, and invariant to intra-class variability. PCANet has proven its usage in applications like speech emotion recognition [10], human fall detection [11], vehicle make recognition [12], and so on.

The PCANet resembles convolutional neural network (CNN) in many ways. The way which PCA filters extract features is the same as convolutional computation. The binary quantization block in the PCANet mimics the feature pooling layer in the CNN, and the block-wise histogram in the PCANet adds non-linearity to the network. PCANet is relatively new, and this work is, to the best of our knowledge, one of the first hardware implementations of PCANet. This work is related to the existing hardware implementations of CNNs [13, 14].

There are other data processing networks like PCANet as well. Discriminative locality alignment network (DLANet) has been proposed as a strong feature extractor for scene classification [15]. A compressive sensing model (CSNet) is proposed which has a cascaded structure similar to PCANet. Moreover, both CSNet and PCANet use binary hashing, block-wise histogram, and linear SVM as part of their calculations [16].

3 Algorithm of PCANet

In this section, PCANet's structure is described in detail. Sequentially, an input image is processed through several stages including patch-mean removal, PCA filter with 2D convolution, binary quantization, block-wise histogram, and SVM classification. Note that this work is focused on the implementation of the PCANet detector and the training procedure is not implemented in hardware. Coefficients of the PCA filter and SVM classifier are pre-trained offline using datasets from the target applications.

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018

3.1 Patch-mean removal

Prior to applying the PCA filters, all pixels in the patch need to have zero mean. Thus, the process is called "patch-mean removal": calculating the mean of all pixels in the patch and then subtract it from the pixel values. For each pixel in the image, a patch is generated with the pixel at the center. For the pixels around the edges, the image is padded with zeros. After removing its mean, each patch is stored separately and goes through the PCA filter [7].

3.2 Training convolutional filters

The training procedure of PCANet works in this way: for a single training image, a k-by-k sliding window is used to sample the training image to generate a sequence of k-by-k image patches. A single k-by-k image patch can be regarded as a k^2 element vector:

$$X = [x_1 x_2, \ldots, x_{kk}]. \tag{1}$$

Assuming there is a total of m training images, and each training image generates n such k-by-k image patches. The k^2 element vector contains k^2 variance, and then principal component analysis is applied to transform the vector X to a new coordinate where the greatest variance lies on the first coordinate (also called the first principal component), the second greatest variance on the second coordinate, and so on. In network inference, multiplying with trained PCANet filters is like transforming the input image to another coordinate. Generated feature maps are in such an order that the first feature map contains the greatest variance, and the second feature map contains the second greatest variance, and so on. The detailed training procedure of PCANet can be referred to [7].

3.3 Linear support vector machine

After extracting meaningful PCANet features, a classifier also need to be trained to make final decisions on the input image. In this letter, the linear SVM is used as the classifier.

The function of SVM is to quickly separate hyperplanes between distinct categories and to map those extracted features to high-dimensional feature spaces. The advantage of SVM is that it is quick to train. In this paper, linear SVM is applied, since it has better timing performance on hardware and is more efficient on resource usage when compared to kernel-based SVMs [17]. Previous work [18] shows that linear SVM achieves a good recognition rate for image classification. Expression [2] gives the equation of the linear SVM.

$$y = wx^T + b. \tag{2}$$

4 Hardware architecture of PCANet

For the implementation of PCANet on hardware, we chose typical values for input image size, the number of stages of PCANet, and the PCA filter size. All the input image patches are resized to 27-by-27 pixels. We set the convolutional filter size to 7-by-7 and the number of filters at each stage to 8. Our software simulation shows that these settings produce the most accurate classification results on target applications.

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018

4.1 2D convolution

The operation of 2D convolution is to multiply data in a patch piece-wisely with coefficients from each of the 8 different PCA filters and then to obtain the sum of the 49 products. An adder tree is used for the sum up operation. Since there are 8 feature maps extracted at the first layer, we have instantiated 8 such 2D convolution modules for the first layer of the PCANet on the hardware design.

4.2 Second stage computation

The first stage computation consists of one patch generation module, one patch-mean removal module and eight 2D convolution modules. At the second stage, each feature map from the first stage is again expanded by convolutions with 8 PCA filters. Thus, there are a total of 8 patch generation modules, 8 patch-mean removal modules, and 64 2D convolution modules in second layer. The computational steps are almost identical to the first stage computations except for the different bit width of data being processed.

At second stage, there are 64 such modules. A separate analysis shows that the second stage's computation makes up 90% of the chip area and power consumption. To reduce resource usage and power consumption, we reused these modules at the second stage. Additional buffers are inserted between the first stage and the second stage. In this way, only 1 patch generation module, 1 patch-mean removal module and 8 2D convolution modules are used in the second stage. The first stage still uses 1 patch generation module, 1 patch-mean removal module and 8 2D convolution modules. Our later VLSI results show that such an implementation comes with a small chip size and low power consumption, and can still achieve a high throughput.

4.3 Binary hashing module

After two layers of 2D convolutions, each of the 8 feature maps created by the second stage need to be merged together. First, we take the sign of the input, then multiply those signs with different weights of [128,64,32,16,8,4,2,1]. This process is called binary hashing. On hardware, we first set up an 8-bit register, and then put the sign of each data into different slots of this 8-bit register. The resulting 8-bit register is the sum of these 8 weighted values. Such a structure eliminates the need for multiplication operations, reducing power consumption.

4.4 Block-wise histogram

As the last step of the PCANet feature extraction, a histogram is needed to generalize the features. After binary hashing, there are 8 feature maps, each containing a 27-by-27 matrix of 8-bit data. Each single feature map is then divided into 6-by-6 blocks, each block having 7-by-7 pixels. The first block is taken from the up left corner of the feature map, then the block slides to the right by 4 pixels to generate the second block. The block keeps sliding to the right until it slides out of the 27-by-27 area, then it slides down by 4 pixels and starts from the leftmost pixel to obtain the next block. A total of $6 \times 6 = 36$ blocks are generated in this way. For each block, one histogram ranging from 0 to 255 is then generated. A total of 256 comparators and 256 6-bit counters are used, in $7 \times 7 = 49$ clock cycles,

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018

data from one block is sent to such a circuit, all 256 features are generated at the 50th clock cycle. The architecture of the histogram generation module is shown in Fig. 1.

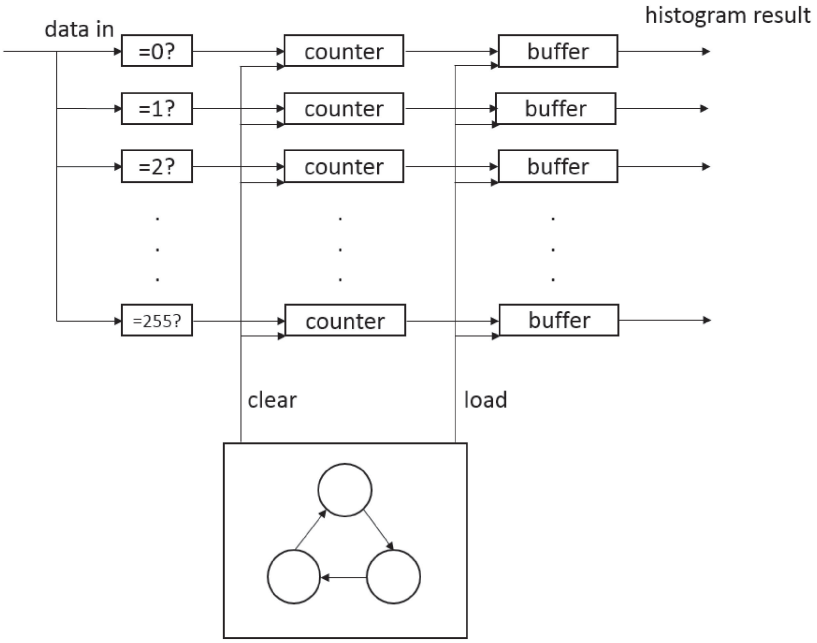

Fig. 1. For one data, 256 comparators are used in parallel to accumulate the right counter. After every data in a 7-by-7 block is processed, the counter results are loaded into buffers.

4.5 Linear support vector machine

Linear SVM makes the classification decision based on the PCANet features. In our work, the input to SVM consists of 8 feature maps, each feature map contains 36 blocks, and each block has 256 histogram values. Therefore, each image patch contains a total of $8 \times 36 \times 256 = 73{,}728$ PCANet features. Creating 73,728 multipliers on hardware will insanely increase power consumption. We need to design an efficient SVM architecture for the such a large vector feature vector. In our work, we decided to use a total of 128 multipliers through experiments. In this way, we divided these features into small pieces and multiplied with coefficients piece-wisely. A state machine based module is introduced to control the status of the multiplications. Once all multiplications are completed, it proceeds to adding with the bias and then providing the final classification results of the PCANet.

As explained in section 4.2, we only process one feature map at a time, and there are 36 units working in parallel to extract histograms from a single feature map. As indicated in Fig. 2, 36 buffers are used to store histogram results, and each buffer contains 256 memory slots. On the other side of the input buffer array, at one clock cycle, 128 features are taken for multiplication. After multiplication, an adder tree is attached to compute the summation of these 128 data. SVM coefficients are taken from 32 separate ROMs, with each ROM size 64-by-1024 bits.

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018

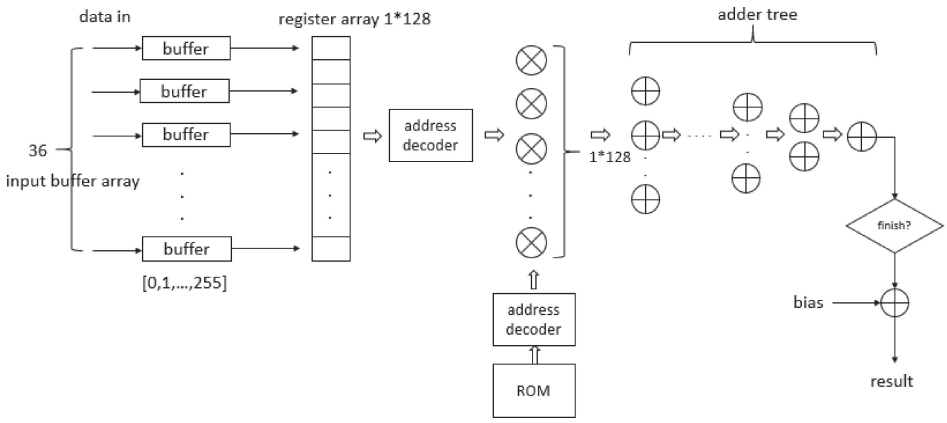

Fig. 2. Hardware architecture of the linear SVM

5 Chip implementation results

5.1 VLSI results

The low power design achieves a highest frequency of 454.5 MHz. Since the PCANet only has a total of $(8 \times 7 \times 7 + 64 \times 7 \times 7) \times 2$ bytes = 6.9 KB weights, we are able to store all these weights to on-chip memory, which is a big difference with other mainstream CNN chips. In this way, there is no need to take in weights from external memory during computation.

The throughput bottleneck of the chip is at the second stage, to process one candidate, $27 \times 27 \times 8/454.5\,\text{M} = 12831.68$ ns is needed. This is equivalent to a processing speed $1 \times 10^9/12831.68 = 77{,}932$ candidates per second.

5.2 Comparison with convolutional neural network

Convolutional neural networks have already been implemented on FPGAs, the implementation on Zynq XC7Z045 SoC FPGA achieves a maximum frequency of 142 MHz, while consuming 8 watts [19]. Another FPGA implementation of CNN is on Xilinx Virtex-4 FPGA, operating at 200 MHz with a power consumption of 15 watts [20]. Implementing a six-layer convolutional neural network requires 10 watts on a Xilinx Virtex-6 VLX240T board [21]. A state-of-the-art FPGA implementation achieves 10 fps with less than 10 watts power consumption [14].

VLSI in nature consumes less power and enables a higher frequency than FPGAs, so we compare our work to other VLSI implementations. Typical CNN chip implementations are shown in Table I.

Table I shows that our implementation achieves the best power efficiency, can process at a higher data precision at the same time. Also note that in the Origami paper, it reports a memory bandwidth of 525 MB/s, which is equivalent to 88.5 frames of 1080P images in one second. We assume their design can exactly catch up with the data streaming, 88.5 fps is their highest throughput.

Compared to other implementations, our implementation can process 77,932 27-by-27 images in one second, which is equivalent to 27.4 fps at 720P, comparable to other CNN chips as shown in Table I. Also, one major difference is that our implementation stands out as the only single-chip solution by storing all the weights on chip, largely reducing integration effort.

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018

Table I. Comparison of PCANet and CNN chip implementations

	PCANet	Neuflow [22]	Origami [23]	ConvEngine [8]
Chip Area/mm^2	3.25	12.5	3.09	2.4
Power/W	0.49	0.6	1.24	0.76
Max Freq./MHz	454.5	400	350	204
VLSI Process/nm	32	45	65	45
Throughput/fps	27.4	24	88.5	30
Image Size	1080P	500 × 375	1080P	1080P
Precision	fixed24	fixed16	fixed12	fixed10

6 Performance evaluation results

In our work, targeting ADAS applications, we evaluated the PCANet detector on road marking detection and traffic light detection.

6.1 Road marking detection

Road marking detection is a very important function for advanced driver assistance systems. Previously, we evaluated the PCANet detector on road marking dataset provided by Wu and Ranganathan [24]. The road marking dataset contains 1,443 street view images, each with a size of 800-by-600. There is a total of 11 road markings in the dataset, we only selected 9 classes because the other 2 classes do not provide enough samples for training. We randomly divided all images into 60/40 with no overlap.

For road marking detection, the BING algorithm [25] is used to proposal potential image candidates. The BING algorithm can run at 300 fps on a single laptop CPU and works well for road marking detection. In this letter, the BING algorithm is used to proposal potential image candidates. Then, each proposed image candidate is resized to be 27-by-27 and sent to the PCANet chip.

On the road marking dataset, we achieved an overall accuracy of 96.8% on 9 classes. As presented in the paper [26], PCANet classification accuracy is more consistent and much better than the original road marking detection work done by Wu and Ranganathan [24], especially on "FORWARD" sign detection, where PCANet achieves an accuracy of 96.8%, far exceeding their achieved value of 23.13%.

6.2 Traffic light detection

Traffic light detection, especially red-light detection is very critical, ignoring a red light can be life-threatening. In our work, we built up our own traffic light dataset [27] around the city of Worcester, Massachusetts, USA. The traffic light dataset contains video data collected during summer and winter. Our work showed that the PCANet outperforms the HoG algorithm on traffic light detection. The comparison of HoG and PCANet performance is shown in Table II.

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018

Table II. Comparison of PCANet and CNN chip implementations

	Precision Rate	Recall Rate	Total True Positives
HoG	80.3%	89.1%	3,586
PCANet	93.1%	93.2%	3,752

7 Conclusion

In this paper, we investigate the PCANet algorithm and its hardware architecture as a single-chip object detector. The PCANet detector achieves satisfactory performance on road marking detection and traffic light detection, and is applicable to many other ADAS applications. The ASIC implementation is able to process 27.4 frames of 1080P images per second, with the power consumption of only 0.5 watt. Compared to other typical CNN based chip implementations, the PCANet implementation achieves better power efficiency and higher throughput. Moreover, the PCANet only has 6.9 K Bytes of weights that is much lesser than that of a CNN. All weights can be stored on chip for fast data access. The proposed PCANet detector is a high-throughput and power-efficient solution for real-time vision applications.

Acknowledgments

This work was supported in part by the U.S. NSF under Grant 1626236 and by The MathWorks. The authors are with Worcester Polytechnic Institute, Massachusetts 01609, USA. Huang is also with Nantong University, Jiangsu 226019, China. The corresponding author is Xinming Huang (e-mail: xhuang@wpi.edu).

DOI: 10.1587/elex.15.20180396
Received April 18, 2018
Accepted May 2, 2018
Publicized May 28, 2018
Copyedited June 25, 2018