IEIE Transactions on Smart Processing and Computing, vol. 9, no. 6, December 2020
https://doi.org/10.5573/IEIESPC.2020.9.6.497

An Accurate Weight Binarization Scheme for CNN Object Detectors with Two Scaling Factors

Xuan Truong Nguyen[1], Tuan Nghia Nguyen[1], Hyuk-Jae Lee[1], and Hyun Kim[2]

[1] Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea
[2] Department of Electrical and Information Engineering; Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul 01811, Korea

* Corresponding Author: Hyun Kim, hyunkim@seoultech.ac.kr

Received May 20, 2020; Revised June 19, 2020; Accepted June 30, 2020; Published December 30, 2020

* *Regular Paper*

* *Extended from a Conference: Preliminary results of this paper were presented at the IEEE International Conference on Electronics, Information, and Communication (ICEIC), 2020. This paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

***Abstract*:** Recently, convolutional neural network (CNN)-based object detectors such as You Only Look Once (YOLO) have been intensively studied for applications in robotics, drones, and autonomous driving. Although YOLO can run in real time by using a graphics processing unit, the YOLO hardware implementation has received a great deal of interest due to its power efficiency and the potential for massive chip production. However, extensive memory access and high computation complexity are widely known as bottlenecks in YOLO hardware implementation. A common and intuitive approach is to apply quantization, especially binarization, to object detectors. However, the existing binarization methods suffer from substantial degradation in detection performance. To address the problem, this study proposes an accurate weight binarization scheme using two scaling factors. Specifically, a new binary weight optimization problem is formulated, and an analytical solution is derived. Experimental results with well-known PASCAL Visual Object Classes show that the proposed method reduces the detection accuracy degradation by up to 32.18% while meeting the memory and computation requirements of state-of-the-art methods.

***Keywords*:** YOLO hardware, Binary weight, Binarization, Quantization, Object detector

1. Introduction

In recent years, object detection has been intensively studied due to emerging applications such as autonomous driving [1-4] and robotics [5]. Especially, with the popularity of the convolutional neural network (CNN), there are many fast and accurate CNN-based object detectors, such as the faster region-based CNN (FRCNN) [6], the single shot multibox detector (SSD) [7], and You Only Look Once (YOLO) [8-10]. It is widely known that YOLOv2 [9] is one of the best detectors to achieve high detection accuracy over large numbers of different object categories in real time on a graphics processing unit (GPU). Because a GPU requires huge amounts of power due to extensive memory accesses and high computational complexity [11-13], the hardware implementation and chip fabrication of CNN-based object detectors (*e.g.* YOLOv2) receive a great deal of interest for their power efficiency and the potential for massive production [14, 15]. However, YOLOv2, with 23 CNN layers, still requires a considerable amount of memory for weights, which is not practical when using on-chip memory. Meanwhile, storing weights in off-chip memory (*i.e.*, dynamic random access memory [DRAM]) causes high latency during memory access. As a result, weight quantization or binarization becomes an intuitive method to efficiently reduce latency and memory size. However, state-of-the-art (SOTA) binarization methods [16, 17] suffer from a substantial drop in accuracy when applied to the object detection task. In particular, being different from a classification task, it is not straightforward to train and achieve accurate binary object detection, so sequential training or knowledge distillation has been used [18]. As shown in Fig. 1, the results of three different binarization methods are quite different when only a single layer (*i.e.*, either a second CNN layer or the last CNN layer) is binarized. To address

the above problem, this paper proposes a new binarization method by using two scaling factors. Experimental results show the proposed method can mitigate the performance drop by up to 32.18% on the PASCAL Visual Object Classes (VOC) dataset [19] while achieving memory and complexity reduction similar to SOTA methods.

To this end, the contributions of this paper are as follows.

1) A new binarization optimization problem with two scaling factors is formulated, and its analytical solution is derived.

2) Experiments are conducted to show the efficiency of the proposed method.

The rest of this paper is organized as follows. Section 2 describes the related works. In Section 3, the proposed optimization scheme for binarization with two scaling factors is formulated, and its solution is derived. Section 4 includes performance evaluations and a discussion of the effect of the proposed scheme. Finally, Section 5 concludes the paper.

2. Related Works

In this section, background regarding YOLOv2 and binarization are presented.

2.1 YOLOv2

The YOLOv2 network comprises two parts. The first takes and extracts features from the input image, while the second (*i.e.*, the last convolutional layer) is in charge of localization and classification. The YOLOv2 network is derived from the original Darknet-19 architecture [9]. It should be noted that the 32-bit weight file consumes 193 MB of memory for storage [20]. Therefore, weight binarization becomes a promising solution that can reduce memory storage by approximately 32 times.

2.2 Binarization

Weight binarization can be formulated as an optimization problem as follows:

$$\min_{W_l^b} \left\| W_l - W_l^b \right\|_2 \tag{1}$$

$$\text{s.t. } W_l^b = \alpha_l b_l,\ \alpha_l > 0,\ b_l \in \{\pm 1\}^{n_l},\ l = 1, \ldots, L,$$

where W_l and W_l^b are full-precision and binary weights for the l-th convolution layer, respectively, and α_l is a scaling factor.

In [16], the scaling factor was not used, and therefore, weights were constrained to two values: -1 and 1. Eventually, binary weights are simply defined via sign function, as follows:

$$W_l^b = sign(W_l). \tag{2}$$

To improve performance, in [17], scaling factor α_l was used, and this factor is computed as follows:

$$\alpha_l = \frac{\|W_l\|_1}{n_l}. \tag{3}$$

However, these weight binarization methods suffer from a substantial drop in detection accuracy [16, 17]. Therefore, this study aims to propose a novel binarization method that significantly mitigates the accuracy drop.

2.3 Block Truncation Coding

Block truncation coding (BTC) [21] is an image compression algorithm for grayscale images. It divides an image into blocks, and uses a quantizer to reduce the number of gray levels for each block while maintaining the same mean and standard deviation. BTC is applied for a small block size (*i.e.*, 4×4), because pixels in a block are usually highly correlated. In this paper, the mathematical derivation of BTC is utilized to find scaling factors in the proposed binarization scheme, and the details are described in Section 3.

3. The Proposed Scheme

3.1 The Optimization Problem of Weight Binarization using Two Scaling Factors

This subsection presents an accurate weight binarization method. First, to use two scaling factors, the binary weight optimization problem in (1) is reformulated as follows:

$$\min_{W_l^b} J_{MSE} = \left\| W_l - W_l^b \right\|_2 \tag{4}$$

$$\text{s.t. } W_l^b \in \{\alpha_l, \beta_l\}^{n_l},\ l = 1, \ldots, L,$$

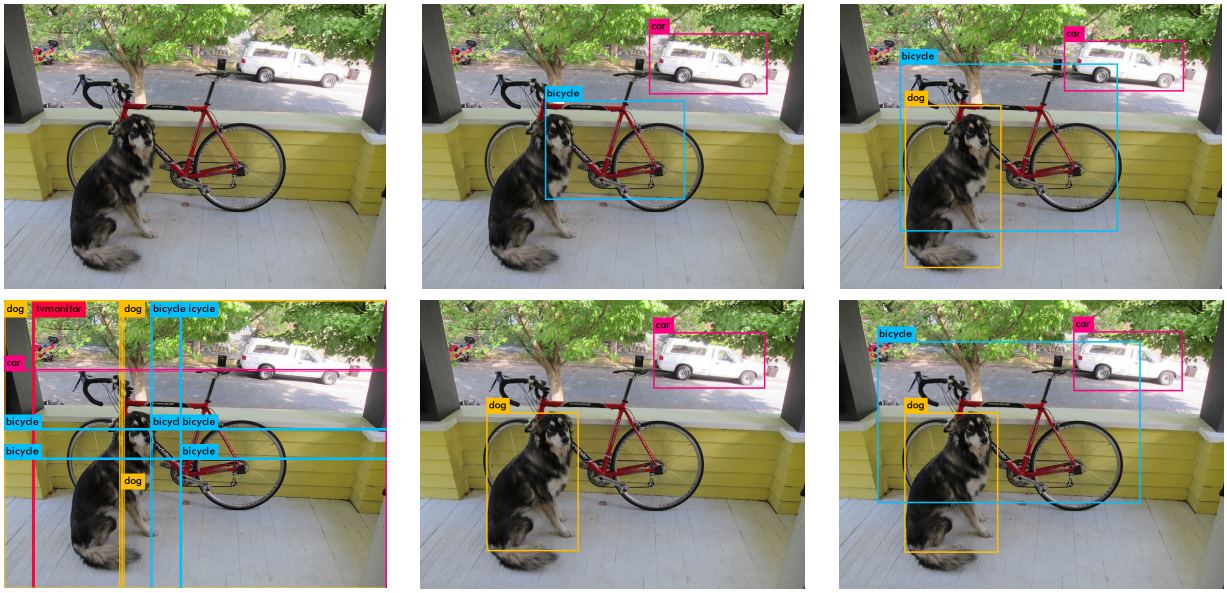
where α_l and β_l are two scaling factors for the l-th layer, and n_l is the number of weights for the l-th layer.

To solve (4), consider the design of a one-bit quantizer that finds a threshold, m_l, and two output levels, α_l and β_l, such that:

$$w_l^{(i)} = \begin{cases} \beta_l, \text{ if } w_l^{(i)} > m_l \\ \alpha_l, \text{ if } w_l^{(i)} \le m_l \end{cases}, \text{ for } i = 1, \ldots, n_l. \tag{5}$$

Without loss of generality, assume that full-precision weights $\{w_l^{(i)}\}_{i=1}^{n_l}$ are sorted (*i.e.*, $w_l^{(1)} \le w_l^{(2)} \le \ldots \le w_l^{(n_l)}$). Then, assume q is a number for $w_l^{(i)}$ that is greater than m_l. Consequently, the objective function in (4) can be rewritten as follows:

$$J_{MSE} = \sum_{i=1}^{n_l - q} \left(w_l^{(i)} - \alpha_l \right)^2 + \sum_{i=n_l - q + 1}^{n_l} \left(w_l^{(i)} - \beta_l \right)^2. \tag{6}$$



(a) (b) (c)

Fig. 1. Detection results of binarized YOLOv2 with three binarization methods (a) Binaryconnect [8], (b) XNORNET, (c) the proposed method. For visualization, the binarization methods are only applied to one single convolution layer (*i.e.*, the second layer or the last layer) of the pre-trained YOLOv2 [20] without retraining the network. In other words, the upper image is generated by applying binarization to the CONV layer 2; and the lower image is generated by applying the binarization to the last CONV layer before the detection layer.

Inspired by BTC [21], this subsection presents a method to derive α_l and β_l. By setting the partial derivatives of J_{MSE} with respect to α_l and β_l to zeros, α_l and β_l are derived with

$$\alpha_l = \frac{1}{n_l - q}\sum_{i=1}^{n_l-q} w_l^{(i)}, \tag{7}$$

$$\beta_l = \frac{1}{q}\sum_{i=n_l-q+1}^{n_l} w_l^{(i)}. \tag{8}$$

Let the first and second moments and the variance of the weights be:

$$\overline{w_l} = \frac{1}{n_l}\sum_{i}^{n_l} w_l^{(i)} \tag{9}$$

$$\overline{w_l^2} = \frac{1}{n_l}\sum_{i}^{n_l}\left(w_l^{(i)}\right)^2 \tag{10}$$

$$\sigma_l = \overline{w_l^2} - \overline{w_l}^2, \tag{11}$$

respectively. Similar to [21], threshold m_l is set to $\overline{w_l}$. In addition, threshold m_l and two output levels, α_l and β_l, are set to preserve $\overline{w_l}$ and $\overline{w_l^2}$ as follows:

$$n_l\overline{w_l} = (n_l - q)\alpha_l + q\beta_l, \tag{12}$$

$$n_l\overline{w_l^2} = (n_l - q)\alpha_l^2 + q\beta_l^2. \tag{13}$$

By solving (12) and (13), α_l and β_l are derived as follows:

$$\alpha_l = m_l - \sigma_l\sqrt{\frac{q}{n_l - q}}, \tag{14}$$

$$\beta_l = m_l + \sigma_l\sqrt{\frac{n_l - q}{q}}. \tag{15}$$

3.2 Training the Binarized YOLOv2

The proposed binarized YOLOv2 is trained by using the stochastic gradient descent (SGD) method as the SOTA methods [16, 17]. The difference is that the binary weight function is changed to (3) with threshold m_l and the two output levels, α_l and β_l, defined in (14) and (15). Binary weights are only used for forward propagation, while full-precision is used for updating gradients in backward propagation.

4. Performance Evaluation

4.1 Performance Comparison

This section compares the proposed method with the SOTA approaches on the popular PASCAL VOC dataset [19], which consists of 20 classes that include cars, persons, bicycles, and buses. The baseline model is a published model of YOLOv2 trained on the PASCAL VOC dataset. The input image of the model was fixed at 416×416 as a baseline. The learning rate was initialized at 0.001, and gradually decreased by a scale of 0.1 at epochs 40,000, 60,000, 80,000, and 100,000. The batch size was 64 with a subdivision of 8. The well-known mean average precision (mAP) was used for the evaluation metric.

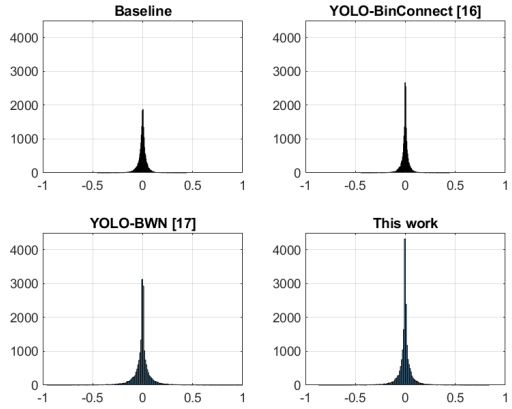
Table 1. Experimental results of binarized YOLOv2 variations on the VOC dataset.

Method	Bits (w-a)	mAP (%)	Diff.
Baseline	32-w 32-a	75.88	-
[14]	1-w 32-a	67.60	-8.28
[15]	1-w 6-a	65.07	-10.81
Yolo-BC [16]	1-w 32-a	61.35	-14.53
Yolo-BWN [17]	1-w 32-a	71.56	-4.32
Proposed (w/ input)	**1-w 32-a**	**72.45**	**-3.43**
Proposed (w/o input)	**1-w 32-a**	**72.95**	**-2.93**

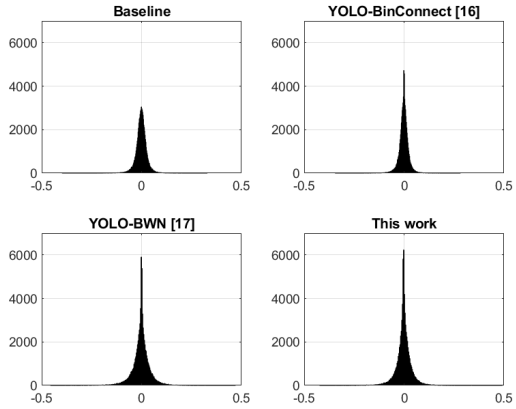
The experimental results are reported in Table 1. In the second column, the number of bits used for weight (w) and activation (a) is reported for each method. For example, [14] uses one-bit weights (1-w) and 32-bit activations (32-a), while [15] uses one-bit weights and six-bit activations. The baseline YOLOv2 was downloaded from a model pre-trained by the author of [20] and marked with 32-bit weights and 32-bit activations. It should be noted that the mAP of the pre-trained model was 75.88%, which is slightly different than in [1]. Note that compared to the baseline, [14] and [15] gave much lower performance, because they use a smaller number of layers for hardware implementation. Meanwhile, YOLO–Binary Connect (YOLO-BC) and YOLO-Binary Weight Network (BWN) were directly obtained by applying the binarization methods in [16] and [17], respectively, re-training from the baseline. It is observed that the binarization method without a scaling factor [16] significantly reduced detection accuracy by about 14.53%, compared to the baseline. One possible reason is that YOLO includes both localization and classification tasks, and then depends highly on the scaling factor. The binarization method in [17] also degraded detection accuracy by about 4.32%. The experimental results with the proposed binarization using two scaling factors are reported in the seventh and eighth rows. It is observed that the proposed method in the seventh row can achieve a mAP of 72.45%, mitigating the performance degradation by about 20.6%, compared to YOLO-BWN [17]. The proposed method can be further improved by not applying weight binarization for the input layer. It achieves a mAP of 72.95%, and therefore, mitigated the accuracy degradation by about 32.18%, compared to YOLO-BWN [17].

4.2 Convergence Speed and Training Time

Table 2 shows the detection accuracy of three categories in autonomous driving (cars, persons, and bicycles) of the models saved at different epochs during the training time. The experimental results demonstrate that the proposed method consistently improves the detection performance during the training, and has the best performance when compared with the previous studies. It is interesting to observe that YOLO-BC [16] can reach a moderate accuracy level, although it starts with a very low performance.



(a) Convolutional Layer 2



(b) Convolutional Layer 3

Fig. 2. Weight distributions of the baseline YOLOv2 and three binarized variations.

In terms of training time, the proposed method requires more time for backpropagation. In particular, when training on a single TITAN X (PASCAL) GPU, it takes about 2.225 seconds per epoch. In comparison, YOLO-BC [16] and YOLO-BWN [17] required 1.095 and 1.843 seconds per epoch, respectively. However, considering both convergence speed and training time, three binarization methods can reach well-trained models after a similar amount of time. In addition, it should be noted that the training phase is less sensitive to processing speed than the inference phase.

4.3 Weight Distributions

Fig. 2 shows the weight distributions of convolutional layers 2 and 3 of the baseline YOLOv2 and three binarized variations. It is interesting to observe that compared to the baseline, all three binarized versions encourage more zero-weights. Among the three versions, the proposed method generated more zero weights, and eventually, enhances detection accuracy.

4.4 Binary Weight Optimization and Block Truncation Coding

The derivation of the proposed binary weights in (5) and scaling factors in (14) and (15) are similar to BTC in

Table 2. Detection accuracy of car, person, and bicycle categories during training time.

		Detection accuracy (mAP) of the models in each epoch								
		10,000	20,000	30,000	40,000	50,000	60,000	70,000	80,000	90,000
Car	Baseline	**0.8003**	**0.8003**	**0.8003**	**0.8003**	**0.8003**	**0.8003**	**0.8003**	**0.8003**	**0.8003**
	BinConnect [16]	0.2737	0.4776	0.623	0.6964	0.6908	0.6962	0.7063	0.7086	0.6922
	BWN [17]	0.7145	0.7315	0.7395	0.7794	0.7830	0.7828	0.7852	0.7870	0.7865
	Proposed	0.7361	0.7579	0.7639	0.786	0.7844	0.7830	0.7905	0.7839	0.7861
Person	Baseline	**0.7787**	**0.7787**	**0.7787**	**0.7787**	**0.7787**	**0.7787**	**0.7787**	**0.7787**	**0.7787**
	BinConnect [16]	0.3362	0.4590	0.6136	0.6670	0.6648	0.6713	0.6792	0.6827	0.6709
	BWN [17]	0.6852	0.6978	0.7039	0.7335	0.7332	0.7369	0.7409	0.7407	0.7426
	Proposed	0.6795	0.7144	0.7110	0.7321	0.7436	0.7399	0.7476	0.7489	0.7521
Bicycle	Baseline	**0.8390**	**0.8390**	**0.8390**	**0.8390**	**0.8390**	**0.8390**	**0.8390**	**0.8390**	**0.8390**
	BinConnect [16]	0.1618	0.3288	0.5522	0.7020	0.7041	0.7149	0.7086	0.7217	0.7127
	BWN [17]	0.7145	0.7315	0.7395	0.7794	0.7830	0.7828	0.7852	0.7870	0.7865
	Proposed	0.7335	0.7706	0.7804	0.8021	0.8087	0.8193	0.8169	0.8169	0.8196

Table 3. Weight sizes of the YOLOv2 baseline and the three binarized methods.

	No. of input channels	No. of output channels	Filter size	No. of weights	Baseline (bits)	BC [16] (bits)	BWN [17] (bits)	This work (bits)
conv01	3	32	3	864	27648	864	1888	2912
conv02	32	64	3	18432	589824	18432	20480	22528
conv03	64	128	3	73728	2359296	73728	77824	81920
conv04	128	64	1	8192	262144	8192	10240	12288
conv05	64	128	3	73728	2359296	73728	77824	81920
conv06	128	256	3	294912	9437184	294912	303104	311296
conv07	256	128	1	32768	1048576	32768	36864	40960
conv08	128	256	3	294912	9437184	294912	303104	311296
conv09	256	512	3	1179648	37748736	1179648	1196032	1212416
conv10	512	256	1	131072	4194304	131072	139264	147456
conv11	256	512	3	1179648	37748736	1179648	1196032	1212416
conv12	512	256	1	131072	4194304	131072	139264	147456
conv13	256	512	3	1179648	37748736	1179648	1196032	1212416
conv14	512	1024	3	4718592	150994944	4718592	4751360	4784128
conv15	1024	512	1	524288	16777216	524288	540672	557056
conv16	512	1024	3	4718592	150994944	4718592	4751360	4784128
conv17	1024	512	1	524288	16777216	524288	540672	557056
conv18	512	1024	3	4718592	150994944	4718592	4751360	4784128
conv19	1024	1024	3	9437184	301989888	9437184	9469952	9502720
conv20	1024	1024	3	9437184	301989888	9437184	9469952	9502720
conv21	512	64	1	32768	1048576	32768	34816	36864
conv22	1280	1024	3	11796480	377487360	11796480	11829248	11862016
conv23	1024	125	1	128000	4096000	4096000	4096000	4096000
Size (MB)					**193.16**	**6.51**	**6.55**	**6.59**

image compression [21]. Despite the fact that they share a similarity in the mathematical formula, they came from different contexts. In other words, BTC comes from the concept that the pixel values in a natural image block are similar, and can be approximated by using two output levels. Meanwhile, there is no clear evidence that the convolutional weights are likely similar, and the problem is directly derived from the concept of quantization. To this end, it is interesting to bridge the binary weight problem and the block-based image compression algorithm.

4.5 Memory Storage for Weights

Table 3 reports the detailed architecture of the YOLOv2 network, which includes the number of input channels, the number of output channels, the convolutional

filters, and the number of weights. There are 23 layers. For simplicity, the number of biases and normalization parameters (i.e., scales, means, and variances) are not included, since they are equal to the number of output channels. For each layer, the weights are grouped based on the number of output channels. Therefore, the proposed method slightly increases the memory size, when compared to BinaryConnect [16] and BWN [17]. The sizes of the baseline and the three models for binarized YOLOv2 are shown in the fifth, sixth, seventh, and eighth columns, respectively. The baseline takes about 193.16 MB, while BinaryConnect [16], BWN [17], and the proposed method take only 6.51 MB, 6.55 MB, and 6.59 MB, respectively, achieving an approximate 30× memory usage reduction. It should be noted that binarization was not applied to the last detection layer.

5. Conclusion

This research proposed a weight binarization optimization problem with two scaling factors and derived its analytical solution. The proposed method outperformed the existing binarization methods in accuracy, while meeting similar memory and computation requirements.

Acknowledgement

This work was supported in part by The Project of Industrial Technology Innovation through the Ministry of Trade, Industry and Energy (MOTIE) under Grant 10082585, and was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2019R1A6A1A03032119.

References

[1] A. Geiger, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), 2012. Article (CrossRef Link)

[2] J. Choi, D. Chun, H. Kim, and H.-J., Lee, “Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving,” in *Proceedings of IEEE International Conference on Computer Vision* (*ICCV*), Oct. 2019, pp. 502-511. Article (CrossRef Link)

[3] J. Choi, D. Chun, H.-J., Lee, and H. Kim, “Uncertainty-Based Object Detector for Autonomous Driving Embedded Platforms,” in *Proceedings of 2nd IEEE International Conference on Artificial Intelligent Circuits Systems* (*AICAS*), Aug. 2020. Article (CrossRef Link)

[4] X. T. Nguyen, K.-T. Nguyen, H. Kim, and H.-J. Lee, “ROI-based LiDAR Sampling Algorithm in On-road Environment for Autonomous Driving,” *IEEE Access*, vol. 7, pp. 90243-90253, Jul. 2019. Article (CrossRef Link)

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *International Journal of Robotics Research* (*IJRR*), vol. 32, no. 11, pp. 1231–1237, 2013. Article (CrossRef Link)

[6] R. B. Girshick, “Fast R-CNN,” *CoRR*, abs/1504.08083, 2015. Article (CrossRef Link)

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single Shot MultiBox Detector,” In *European Conference on Computer Vision* (*ECCV*), pp. 525–542, 2016. Article (CrossRef Link)

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detector,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), 2016. Article (CrossRef Link)

[9] J. Redmon, and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), 2017. Article (CrossRef Link)

[10] J. Redmon, and A. Farhadi, “YOLOv3: An Incremental Improvement,” *Technical Report*, *arXiv preprint*, 2018. Article (CrossRef Link)

[11] D. T. Nguyen, N. H. Hung, H. Kim, and H.-J. Lee, “An Approximate Memory Architecture for Energy Saving in Deep Learning Applications,” *IEEE Transaction on Circuits Systems I*, vol. 67, no. 5, pp. 1861-1873, May 2020. Article (CrossRef Link)

[12] M. Kim, I.-J. Chang, and H.-J. Lee, "Segmented Tag Cache: A Novel Cache Organization for Reducing Dynamic Read Energy," *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1546-1552, Oct. 2019. Article (CrossRef Link)

[13] C. Lee and H.-J. Lee, "Effective Parallelization of a High-Order Graph Matching Algorithm for GPU Execution," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 2, pp. 560-571, Feb. 2019. Article (CrossRef Link)

[14] H. Nakahara, H. Yonekawa, T. Fujii, and S. Sato, “A Lightweight YOLOv2: A Binarized CNN with A Parallel Support Vector Regression for an FPGA,” In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (*FPGA*’18), ACM, pp. 31–40, 2018. Article (CrossRef Link)

[15] D. T. Nguyen, T. N. Nguyen, H. Kim, and H.-J. Lee, "A High-Throughput and Power-Efficient FPGA Implementation of YOLO CNN for Object Detection," in *IEEE Transactions on Very Large Scale Integration* (VLSI) *Systems*, vol. 27, no. 8, pp. 1861-1873, Aug. 2019. Article (CrossRef Link)

[16] M. Courbariaux, Y. Bengio, and J.-P. David. “Binaryconnect: Training deep neural networks with binary weights during propagations,” In *Advances in Neural Information Processing Systems* (*NIPS*), pp. 3123–3131, 2015. Article (CrossRef Link)

[17] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: Imagenet classification using binary convolutional neural networks,” In *European*

Conference on Computer Vision (*ECCV*), pp. 525–542, 2016. Article (CrossRef Link)
[18] J. Xu, P. Wang, H. Yang, and A. M. Lopez, " Training a Binary Weight Object Detector by Knowledge Transfer for Autonomous Driving", in *Proceeding of the International Conference on Robotics and Automation* (*ICRA*), 2018. Article (CrossRef Link)
[19] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *International Journal of Computer Vision*, 88(2):303–338, 2010. Article (CrossRef Link)
[20] https://pjreddie.com/darknet/yolov2/ (Accessed on 2019. 03.31)
[21] E. Delp; and O. Mitchell, "Image Compression Using Block Truncation Coding," *IEEE Transactions on Communications*, vol. 27, no. 9, pp. 1335-1342, 1979. Article (CrossRef Link)

Xuan Truong Nguyen received a BSc in Electrical Engineering from Hanoi University of Science and Technology, Hanoi, Vietnam, in 2011, and an MSc and a PhD in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea, in 2015 and 2019, respectively. He is working as a postdoctoral fellow for BK21+ in the Electrical and Computer Engineering Department of Seoul National University. His research interests are in the areas of algorithm and SoC design for low-complexity computer vision and multimedia applications.

Tuan Nghia Nguyen received a BSc in electronics and telecommunications from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 2017, and an MSc in electrical and computer engineering from Seoul National University, Seoul, Korea, in 2020. He is currently working toward a PhD in electrical and computer engineering at Seoul National University, Seoul, South Korea. His current research interests include computer vision, deep learning applications, and computer architecture.

Hyuk-Jae Lee received BSc and MSc in electronics engineering from Seoul National University, Seoul, South Korea, in 1987 and 1989, respectively, and a PhD in electrical and computer Engineering from Purdue University, West Lafayette, IN, USA, in 1996. From 1998 to 2001, he was a Senior Component Design Engineer at the Server and Workstation Chipset Division, Intel Corporation, Hillsboro, OR, USA. From 1996 to 1998, he was a Faculty Member at the Department of Computer Science, Louisiana Tech University, Ruston, LS, USA. In 2001, he joined the School of Electrical Engineering and Computer Science, Seoul National University, where he is a Professor. He is the Founder of Mamurian Design, Inc., Seoul, a fabless SoC design house for multimedia applications. His current research interests include computer architectures and SoCs for multimedia applications.

Hyun Kim received a BSc, an MSc, and a PhD in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2009, 2011, and 2015, respectively. From 2015 to 2018, he was a BK Assistant Professor of BK21 Creative Research Engineer Development for IT, Seoul National University. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, where he is an Assistant Professor. His current research interests include algorithm, computer architecture, memory, and SoC design for low-complexity multimedia applications, and deep neural networks.