IEIE Transactions on Smart Processing and Computing, vol. 11, no. 4, August 2022
https://doi.org/10.5573/IEIESPC.2022.11.4.255

Efficient Object Detection Acceleration Methods for Autonomous-driving Embedded Platforms

Jiwoong Choi[1], Dayoung Chun[1], Hyuk-Jae Lee[1], and Hyun Kim[2]

[1] Inter-university Semiconductor Research Center (ISRC), Department of Electrical and Computer Engineering, Seoul National University / Seoul 08826, Korea
[2] Research Center for Electrical and Information Technology, Department of Electrical and Information Engineering, Seoul National University of Science and Technology / Seoul 01811, Korea hyunkim@seoultech.ac.kr

* Corresponding Author: Hyun Kim

Received March 24, 2022; Revised May 23, 2022; Accepted June 9, 2022; Published August 30, 2022

** Regular Paper*

***Abstract*:** Object detection in autonomous vehicles is typically operated in an embedded system to reduce power consumption. The use of an object detection algorithm with high accuracy and real-time detection speed in the embedded systems is essential for ensuring safe driving. This study proposes a parallel processing method for GPU and CPU operations to enhance the detection speed of the model. In addition, this study proposes data augmentation and image resize techniques that consider the camera input size of autonomous driving, which increases the accuracy significantly while improving the detection speed. The application of these proposed schemes to a baseline algorithm, tiny Gaussian YOLOv3, improves the mean average precision by 1.14 percent points (pp) for the Berkeley Deep Drive (BDD) dataset and 1.34 pp for the KITTI dataset compared to the baseline. Furthermore, in the NVIDIA Jetson AGX Xavier, which is an embedded platform for autonomous driving, the proposed algorithm improves the detection speed by 22.54 % for the BDD, and 24.67 % for the KITTI compared to the baseline, thereby enabling high-speed real-time detection on both datasets.

***Keywords*:** Object detection, Embedded system, Deep learning, Autonomous driving, NVIDIA Jetson AGX Xavier

1. Introduction

Recently, deep neural network (DNN)-based object detection [1] with camera sensors has shown better detection accuracy than humans, significantly increasing its importance in the object detection part of autonomous vehicles [2, 3]. For autonomous vehicles, the real-time detection speed of object detectors is essential for reducing latency while maintaining a high detection accuracy so that the control system can respond quickly [4]. In addition, reducing power consumption is also essential for autonomous vehicles that operate on battery-generated power. Therefore, autonomous vehicles typically operate based on embedded systems, making it difficult to detect objects in real-time using limited hardware resources, even with a relatively fast and highly efficient DNN-based one-stage detector [5].

To overcome this limitation, lightweight DNN-based object detectors that support a real-time detection speed in embedded platforms and corresponding lightweight and low-power implementation techniques have been proposed in various previous studies [5-12]. These algorithms have focused on improving the detection speed significantly by reducing the computing cost, thereby allowing DNN algorithms to be used in embedded platforms. On the other hand, there is a problem of accuracy loss compared to the conventional object detection algorithms.

Given that improved accuracy is essential for the practical deployment of these lightweight algorithms in autonomous driving, various techniques have been actively studied to enhance the accuracy of lightweight networks [13-15]. Choi *et al.* [13] proposed a model for predicting the localization uncertainty in a lightweight network and used the predicted uncertainty in post-processing to improve the accuracy significantly. On the other hand, the increased computing cost for post-processing leads to a decrease in the overall detection speed of the model. Yi *et al.* [14] and Dong *et al.* [15] enhanced the accuracy by

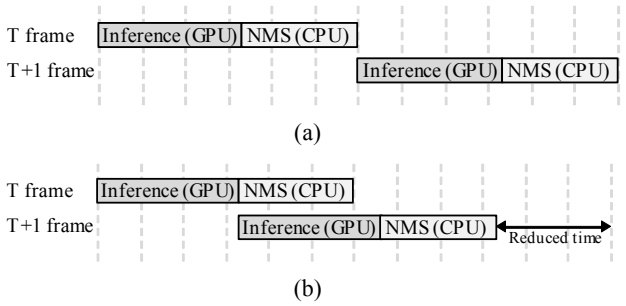


Fig. 1. Examples of the detection process for the conventional algorithm and after applying the proposed technique.

constructing an additional layer in lightweight networks. Unfortunately, these methods also increased the computing cost and decreased the detection speed.

To enhance the detection speed in embedded platforms, this study proposes a parallel processing scheme for CNN operations in a GPU and Non-maximum Suppression (NMS) operations in a CPU, thereby hiding the NMS-processing time in the GPU-processing time while maintaining accuracy. Generally, one-stage object detectors, used widely for autonomous driving, process the input images as square images in the training and inference steps [16]. A preprocessing step is required to convert the inputs to square images, given that all camera sensors employed in recent autonomous-driving applications use a wide-angle camera [17]. Consequently, this conversion damages the original input image. Although CNNs can be well trained to recognize objects in distorted (*i.e.*, square) images [16], the accuracy is significantly degraded for the input of the same ratio as the original image (*i.e.*, wide-angle). To address these problems, this study proposes a new data augmentation technique that considers multiple images and various image ratios in the training step, thereby enabling the model to cope with various input sizes and ratios robustly without the penalty of a detection speed during the inference phase. Furthermore, in the inference phase, the input image is resized to the ratio of the autonomous driving (*i.e.*, wide-angle) camera, thus improving detection speed and further increasing the accuracy in autonomous-driving embedded systems.

By applying all these proposed methods, the mean average precision (mAP) is improved by 1.14 percent points (pp) in the Berkeley deep drive (BDD) dataset and 1.34 pp in the KITTI dataset. The detection speed is also improved by 22.54 % in the BDD and 24.67 % in the KITTI compared to the baseline algorithm, enabling faster and more accurate detection.

2. Proposed Acceleration Methods

2.1 Non-maximum Suppression Hiding

To enhance the detection speed of object detectors in the embedded platforms, this study proposes a parallel processing technique for the convolution operations on the GPU and NMS operations on the CPU, thereby hiding the NMS-processing time into the convolution-processing time. Figs. 1(a) and (b) show the detection process of the conventional algorithm and the process after applying the proposed technique, respectively. As shown in Fig. 1(a), the baseline algorithm does not process the next input image until completing all operations of a currently input image. In autonomous-driving applications, where the images are input through streaming, the processing structure of conventional algorithms is inefficient in terms of hardware utilization. Accordingly, the proposed method employs a pipeline structure to address this issue. As shown in Fig. 1(b), using multi-thread processing, the NMS calculation process of the T frame is hidden by the GPU calculation process of the T+1 frame. Thus, the previous GPU operation result of the T frame stored in the buffer is post-processed simultaneously by the CPU when performing convolution operations of the T+1 frame by the GPU. The processing time of each task is synchronized so that the CPU can process it at the right timing according to the GPU operation. This is possible because there is no data dependency between the GPU inference calculation of the T+1 frame and the CPU post-processing task of the T frame. Therefore, the proposed method is highly efficient in terms of hardware utilization and enables a continuous flow of input images to be efficiently processed.

2.2 Proposed Data Augmentation in Training Step

Among previous augmentation studies, representative Mixup [18] and RICAP [19] enable a one-stage detector to learn various features based on a square input image, but there is a problem in that the detection accuracy is decreased for non-square image ratios. In addition, fully convolutional network (FCN), which is a typical network structure employed in recent object detectors, determines the amount of computation of the entire network according to the input image size. Therefore, the total computational cost of the FCN is sensitive to the size of the input image; the computational cost of the network decreases with decreasing size of the input image, increasing the detection speed. On the other hand, this leads to a decrease in accuracy. To address this problem, this subsection proposes a new data augmentation technique for autonomous driving embedded systems that enhance detection accuracy without compromising the detection speed.

Fig. 2 shows the proposed data augmentation method using a single image. The example on the left side in Fig. 2 shows data augmentation maintaining the ratio of the original image. In the first step, the original image is cropped randomly. This cropped image is inserted into a square training plate while maintaining its ratio. This process prevents the shapes of the objects in the original image from becoming distorted. In the final step, conventional data augmentation schemes, such as flip, saturation, hue, and exposure changes, are applied [20, 21]. The example on the right side in Fig. 2 shows data augmentation without maintaining the ratio of the original image. In the first step, the image is cropped randomly, resized into a square, and inserted into a square training

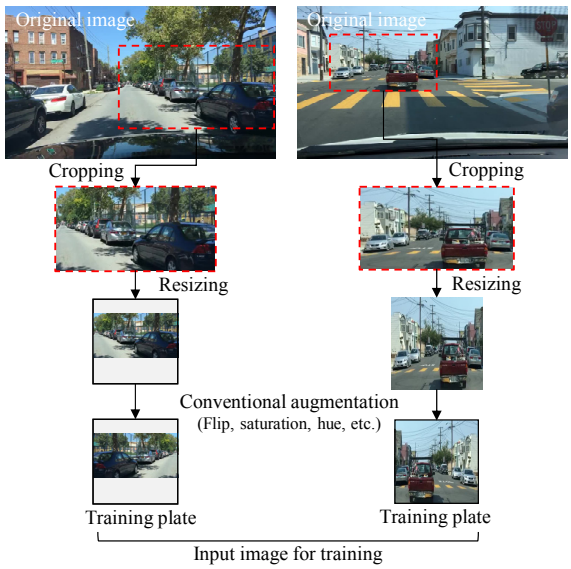


Fig. 2. Example of the proposed data augmentation using single image.

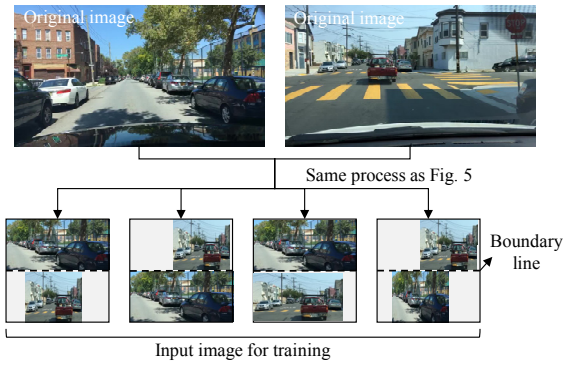


Fig. 3. Example of the proposed data augmentation using two images.

plate. In terms of maintaining the original ratio, an affine transform effect for the objects can be obtained using this method. In the final process, the aforementioned conventional augmentation techniques are also applied to produce the final training image. These two augmentation techniques are used during the training phase to enable the CNN to learn various objects, ratios, and features.

Moreover, the use of multiple images rather than a single image can greatly increase the diversity of the data and prevent the overfitting of the CNN with deep layers [19]. Fig. 3 shows the proposed data augmentation technique using two images. In Fig. 3, the augmentation processes that maintain and do not maintain the original ratio are the same as in Fig. 2. When two images are used, the square training plate area is divided into two. Each preprocessed input image (*i.e.*, image with or without maintaining ratio) is inserted independently into the divided area. The area of the training plate is divided based on the width because the input ratio of the autonomous driving camera is wide-angle. Thus, four training images are generated using two input images, allowing the training data to be expanded. The left side in Fig. 4 shows an

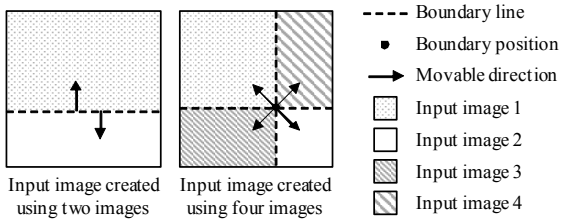


Fig. 4. Example of partitioning a training plate according to various images.

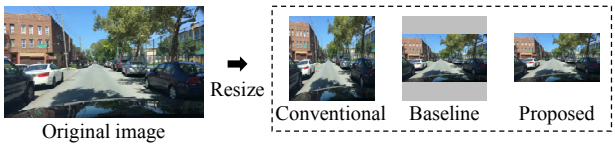


Fig. 5. Examples of the image resize method in the inference phase on the conventional object detector, baseline algorithm, and proposed scheme.

example of forming a training plate using two images. The boundary line on the left side in Fig. 4 is not fixed at the center; it can move in the movable direction (*i.e.*, up and down), further increasing the number of cases of training data.

A training image is produced using four images to expand the diversity of data beyond two images. As shown in the right example in Fig. 4, images with or without the maintained ratio are inserted in each of the four areas partitioned according to the boundary position and line, thus producing the new training data. The boundary position can move within a specific range in the movable direction (*i.e.*, left, right, up, and down), producing more diverse data. Using this proposed augmentation, new global features, which refer to the formation of a new image by combining multiple patches from multiple images [19], can be produced to prevent overfitting. Thus, the accuracy is improved significantly.

2.3 Proposed Image Resize in Inference Step

This subsection proposes a preprocessing scheme that enhances the detection speed by eliminating unnecessary operations in object detectors for an autonomous-driving embedded system. In actual autonomous driving, in general, Full HD (*i.e.*, 1920×1080) is used as the input resolution [22]. However, because this Full HD size causes serious power consumption and speed degradation, it is common to automatically resize the high-resolution image and process it at a lower resolution. Fig. 5 shows the difference in image resizing with the conventional object detectors and after applying the proposed technique in the inference phase. While most previous methods [16] resize the image with a square ratio (*i.e.*, Conventional in Fig. 5), YOLO-based object detectors [6, 13] maintain the actual input ratio when resizing to enhance the accuracy. However, as shown in the Baseline image in Fig. 5, YOLO-based object detectors also execute operations on the square size input by filling in certain pixel values in the

Table 1. Accuracy and FLOPs comparison.

Method	mAP (%)	Diff.	FLOPs ($\times 10^9$)	Input size
BDD test set				
Baseline [13]	8.56		8.27	512×512
+ Proposed augmentation	9.54	+0.98	8.27	512×512
+ Proposed resize	9.70	+1.14	8.14	672×384
KITTI validation set				
Baseline [13]	68.69		8.26	512×512
+ Proposed augmentation	69.65	+0.96	8.26	512×512
+ Proposed resize	70.03	+1.34	6.19	768×256

margin area in the image (*i.e.*, letterbox). In other words, a square input that maintains the input image ratio is generated and processed in the network. Although this provides the advantage of maintaining the input ratio for high accuracy, there is no benefit of reducing the computational cost because the total input still has the same image size of the square ratio.

The proposed inference preprocessing technique can compensate for the inefficient structure by removing unnecessary upper and lower letterbox areas from the baseline method, thereby computing convolution operations only on meaningful pixels and greatly reducing the computational cost. Furthermore, as the algorithm using the proposed data augmentation technique robustly detects diverse input ratios by resizing the image while maintaining the original ratio in the inference phase, the computational cost is reduced while significantly enhancing the detection accuracy. In other words, using the effect of reduced computation, the image can be notably resized again, considering the trade-off between the computing cost and detection speed, thereby improving accuracy.

3. Experimental Results

3.1 Experimental Environment

The superiority of the proposed methods is assessed by experiments using BDD [23] and KITTI [24] datasets, which are widely used in autonomous-driving research. The same datasets, baseline algorithm (*i.e.*, tiny Gaussian YOLOv3), open-source, and experimental settings used in [13] are used for a fair comparison.

3.2 Accuracy Evaluation

Table 1 lists the mAP and floating-point operations per second (FLOPs) results of the baseline algorithm [13] and the algorithms applying the proposed methods for the BDD and KITTI datasets. The input resolution is set to 512×512 as in [13] to compare the accuracy fairly, excluding the proposed image resize method. It is noteworthy that in BDD, the number of classes is larger than that of KITTI, and the data diversity is also higher. In addition, because BDD applies the strict IoU threshold (*i.e.*, IoU>0.75) for all classes in the evaluation, it shows a lower mAP than KITTI [24]. When applying the proposed data augmentation process (+Proposed augmentation in Table 1), the mAP is improved by 0.98 pp and 0.96 pp compared to the baseline algorithm for the BDD and KITTI datasets, respectively. It should be noted that as the proposed data augmentation scheme is applied only to the training phase, the computing cost during the inference phase is not increased. Finally, by applying the proposed resize technique maintaining the original image ratio rather than resizing to a square (+Proposed resize in Table 1), the mAP is improved greatly by 1.14 pp for the BDD and 1.34 pp for the KITTI dataset with respect to the baseline. It is noteworthy that applying the proposed resize technique, which removes unnecessary letterbox operations, causes the input image to become smaller than the square size (*i.e.*, 512×512). Accordingly, considering the trade-off between the accuracy and computing cost, it can be scaled up again while maintaining the original image ratio (*i.e.*, 672×384 in BDD and 768×256 in KITTI. The reason for these values is described in Section 3.3), thereby simultaneously improving the accuracy and reducing the computing cost (*i.e.*, FLOPs).

Table 2 lists the accuracy of the existing data augmentation studies [18, 19] and the proposed techniques. When making a square image size, Mixup [18], RICAP [19], and the proposed method resize the image into a square using the conventional method shown in Fig. 5 in the inference phase. In contrast, when maintaining the original ratio, they resize the image using the proposed scheme shown in Fig. 5. For square images, all previous techniques, Mixup [18] and RICAP [19], improve the accuracy compared to the baseline for the BDD and KITTI datasets; however, for input images that maintain the original ratio, their accuracy is reduced greatly. For the KITTI, which comprises wider images than FULL HD, the previous techniques significantly degrade the accuracy compared to the baseline. Even in this case, the proposed method improves the mAP of 0.25 pp for the BDD and 1.85 pp for the KITTI dataset compared to the baseline. In other words, the proposed augmentation method improves the accuracy for both square and original ratio input images. When comparing the accuracy of Mixup [18] and RICAP [19] for square images with that of the proposed method which maintains the original ratio, the proposed method also shows better result. Nevertheless, the FLOPs of the algorithm applying the proposed method (*i.e.*, 8.14×10^9 on BDD and 6.19×10^9 on KITTI) are smaller

Table 2. Accuracy comparison with previous augmentation studies according to image resize.

Method	mAP (%)	Input size (Square)	mAP (%)	Input size (Resize)
BDD test set				
Baseline [13]	8.56	512×512	9.45	672×384
+ Mixup [18]	8.87	512×512	7.79	672×384
+ RICAP [19]	9.59	512×512	8.78	672×384
+ Proposed Aug.	9.54	512×512	9.70	672×384
KITTI validation set				
Baseline [13]	68.69	512×512	68.18	768×256
+ Mixup [18]	69.59	512×512	38.88	768×256
+ RICAP [19]	69.66	512×512	52.44	768×256
+ Proposed Aug.	69.65	512×512	70.03	768×256

Table 3. Results of processing time required in default mode of NVIDIA Jetson AGX Xavier.

Time (ms)	Inference (GPU)	Post processing (CPU)		Total
		Confidence calculation	NMS	
BDD test set				
Baseline [13]	27.05	0.36	7.64	35.05
+ Prop. NMS hiding	27.05	0.36	-	27.41
+ Prop. Resize	26.81	0.34	-	27.15
KITTI validation set				
Baseline [13]	26.31	0.16	0.16	26.63
+ Prop. NMS hiding	26.31	0.16	-	26.47
+ Prop. Resize	19.91	0.15	-	20.06

than the FLOPs of Mixup [18] and RICAP [19] (*i.e.*, 8.27×10^9 on BDD and 8.26×10^9 on KITTI), which use the square image size.

3.3 Detection Speed Evaluation

Table 3 shows the inference, post-processing, and total processing times of the baseline (*i.e.*, tiny Gaussian YOLOv3 [13]) and the algorithm applying proposed schemes in NVIDIA Jetson AGX Xavier [25] for the BDD and KITTI datasets. The image size for the proposed resize technique is set to match the FLOPs as closely as possible. Hence, it is set to 672×384 for the BDD and 768×256 for the KITTI dataset, while the remaining method is set to 512×512. Specifically, the size of the original BDD image is 1280×720, and that of the original KITTI image is 1242×375. Therefore, the horizontal:vertical ratio is 1.78:1 and 3.3:1, respectively. To match the number of pixels (*i.e.*, 262,144 pixels) of the baseline square image (*i.e.*, 512×512), the image size for the proposed resize technique is set to 672×384 in BDD and 768×256 in KITTI, while matching the ratio of the original image.

The proposed technique shows greater improvements in performance in default mode, which is typically used in NVIDIA Jetson AGX Xavier. The proposed NMS hiding (+Prop. NMS hiding in Table 3) reduces the total processing time by 7.64 ms for the BDD dataset compared to the baseline. Furthermore, the proposed preprocessing technique (+Prop. Resize in Table 3) reduces the processing time further by 7.9 ms (22.54 %). For the KITTI dataset, applying all proposed techniques reduces the total processing time by 6.57 ms (24.67 %) with respect to the baseline. Finally, applying the proposed schemes to the baseline algorithm improves the accuracy while achieving a detection speed of 36.83 fps (=1000 ms/27.15 ms) for the BDD dataset and 49.85 fps (=1000 ms/20.06 ms) for the KITTI dataset, thereby enabling real-time detection to support faster autonomous driving than the baseline algorithm.

4. Conclusion

This study has proposed a method that enhances the detection speed of an object detector while significantly improving the accuracy in NVIDIA Jetson AGX Xavier, an embedded platform for autonomous driving. To improve the detection speed, this study has proposed a parallel processing scheme for the convolution and post-processing operations. This study has also proposed new data augmentation and image resize techniques. Applying the proposed methods to the baseline achieves outstanding performance gains in accuracy and detection speed, enabling accurate and real-time detection for autonomous driving embedded platforms.

Acknowledgment

This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-01304, Development of Self-learnable Mobile Recursive Neural Network Processor Technology) and 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] J. Choi, I. Elezi, H-J. Lee, C. Farabet, and J. M. Alvarez, "Active Learning for Deep Object Detection via Probabilistic Modeling," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10264-10273.

[2] R. Ravindran et al., "Multi-Object Detection and Tracking, Based on DNN, for Autonomous Vehicles: A Review," in *IEEE Sens. J.*, vol. 21, no. 5, pp. 5668-5677, Mar. 2021.

[3] X. Zhao et al., "Fusion of 3D LIDAR and Camera Data for Object Detection in Autonomous Vehicle Applications," in *IEEE Sens. J.*, vol. 20, no. 9, pp. 4901-4913, May 2020.

[4] 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 *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019.

[5] A. Womg, M. J. Shafiee, F. Li and B. Chwyl, "Tiny SSD: A Tiny Single-Shot Detection Deep Convolutional Neural Network for Real-Time Embedded Object Detection," in *Proc. 15th Conf. on Comput. Robot Vision (CRV)*, May 2018, pp. 95-101.

[6] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint*, arXiv:1804.02767, 2018.

[7] 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," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 8, 2019, pp. 1861-1873.

[8] D. T. Nguyen, N. H. Hung, H. Kim, and H. J. Lee, "An Approximate Memory Architecture for Energy Saving in Deep Learning Applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 5, 2020, pp. 1588-1601.

[9] D. T. Nguyen, H. Kim, H. J. Lee, and I. J. Chang, "An approximate memory architecture for a reduction of refresh power consumption in deep learning applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May. 2018, pp. 1-5.

[10] D. Kang, D. Kang, J. Kang, S. Yoo and S. Ha, "Joint optimization of speed, accuracy, and energy for embedded image recognition systems," in *Proc. 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2018, pp. 715-720.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proc. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 4510-4520.

[12] X. T. Nguyen, T. N. Nguyen, H.-J Lee, and H. Kim, "An Accurate Weight Binarization Scheme for CNN Object Detectors with Two Scaling Factors," *IEIE Transactions on Smart Processing & Computing*, Volume: 9 Issue: 6, Pages:497-503, Dec. 2020.

[13] J. Choi, D. Chun, H-J. Lee, and H. Kim, "Uncertainty-based Object Detector for Autonomous Driving Embedded Platforms," in *Proc. IEEE Int. Conf. Artifici. Intell. Circuits Syst. (AICAS)*, Aug. 2020, pp. 16-20.

[14] Y. Zhang, Y. Shen, and J. Zhang, "An improved tiny-yolov3 pedestrian detection algorithm," *Int. J. Light Electron Opt.*, vol. 183, pp. 17-23, Apr. 2019.

[15] D. Xiao et al., "A target detection model based on improved tiny-yolov3 under the environment of mining truck," *IEEE Access*, vol. 7, Jul. 2019.

[16] Q. Zhao et al., "M2Det: A single-shot object detector based on multi-level feature pyramid network," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, Jan. 2019, pp. 9259-9266.

[17] SEKONIX Corp., "SF332X-10X Family Preliminary Datasheet," Feb. 2020. [Online]. Available: http://sekolab.com/products/camera/

[18] H. Zhang et al., "mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Apr. 2018, pp. 1-13.

[19] R. Takahashi, T. Matsubara, and K. Uehara, "Data augmentation using random image cropping and patching for deep CNNs," *IEEE Trans.Circuits Syst. Video Technol.*, vol. 30, no. 9, 2020, pp. 2917-2931.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2012, pp. 1097-1105.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016. pp. 770-778.

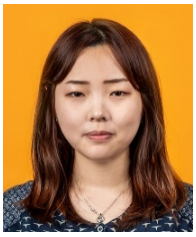
[22] M. Hemmati, M. B-Abhari, and S. Niar, "Adaptive Vehicle Detection for Real-time Autonomous Driving System," in *Proc. Des. Autom. And Test in Eur.Conf. & Exhib.*, 2019.

[23] F. Yu et al., "BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020.

[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3354-3361.

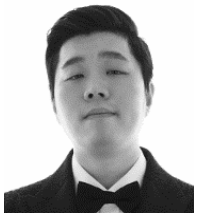
[25] NVIDIA Corp., "NVIDIA Xavier Documentation," Dec. 17, 2018.

Jiwoong Choi received his B.S. degree in electrical and electronics engineering from Chung-ang University, Seoul, South Korea, in 2015, and M.S. and Ph.D. degrees in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2017 and 2021, respectively. He is currently a Deep Learning Research Engineer at NVIDIA, Santa Clara, CA, USA.



Dayoung Chun received her B.S. degree in Electronics Engineering from Sogang University, Seoul, Korea, in 2018. She is working toward Integrated M.S. and Ph.D. degree in Electrical and Computer Engineering at Seoul National University, Seoul. Her research interests include the algorithms and architectures of deep learning, and GPU architecture for computer vision.

Hyuk-Jae Lee received his B.S. and M.S. degrees in electronics engineering from Seoul National University, South Korea, in 1987 and 1989, respectively, and Ph.D. degree in Electrical and Computer Engineering from Purdue University, West Lafayette, IN, in 1996. From 1998 to 2001, he was with the Server and Workstation Chipset Division, Intel Corporation, Hillsboro, OR, as a Senior Component Design Engineer. From 1996 to 1998, he was with the Faculty of the Department of Computer Science, Louisiana Tech University, Ruston, LS. In 2001, he joined the School of Electrical Engineering and Computer Science, Seoul National University, South Korea, where he is currently a Professor. He is a Founder of Mamurian Design, Inc., a fabless SoC design house for multimedia applications. His research interests are in the areas of computer architecture and SoC design for multimedia applications.



Hyun Kim received his B.S., M.S. and Ph.D. degrees in Electrical Engineering and Computer Science from Seoul National University, Seoul, Korea, in 2009, 2011 and 2015, respectively. From 2015 to 2018, he was with the BK21 Creative Research Engineer Development for IT, Seoul National University, Seoul, Korea, as a BK Assistant Professor. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul, Korea, where he is currently working as an Assistant Professor. His research interests are the areas of algorithms, computer architecture, memory, and SoC design for low-complexity multimedia applications and deep neural networks.

Copyrights © 2022 The Institute of Electronics and Information Engineers