2020 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)

Uncertainty-based Object Detector for Autonomous Driving Embedded Platforms

Jiwoong Choi, Dayoung Chun, Hyuk-Jae Lee
Department of Electrical and Computer Engineering and
Inter-university Semiconductor Research Center (ISRC)
Seoul National University, Seoul, Korea
{jwchoi, jjeonda, hyuk_jae_lee}@capp.snu.ac.kr

Hyun Kim
Department of Electrical and Information Engineering and
Research Center for Electrical and Information Technology,
Seoul National University of Science and Technology,
Seoul, Korea
hyunkim@seoultech.ac.kr

***Abstract*—For self-driving cars that operate based on battery-generated power, detection and control are commonly performed in embedded systems to reduce power consumption. To drive safely without driver intervention, it is essential to operate object detection algorithms with high accuracy and fast detection speed within autonomous driving embedded systems. This paper proposes new methods to predict the localization uncertainty by applying Gaussian modeling to the DNN-based tiny YOLOv3 algorithm and consequently, to drastically improve accuracy at the expense of a slight penalty of detection speed by using it in post-processing. Compared to the baseline algorithm (*i.e.,* tiny YOLOv3), the proposed algorithm, tiny Gaussian YOLOv3, improves the mean average precision (mAP) by 2.62 and 4.6 on the Berkeley deep drive (BDD) and KITTI datasets, respectively. Nevertheless, the proposed algorithm is capable of performing real-time detection at 55.56 frames per second (fps) on the BDD dataset and 69.74 fps on the KITTI dataset, respectively, under the mode 0 of the autonomous driving embedded platform, Jetson AGX Xavier.**

Keywords—Uncertainty, object detection, tiny YOLOv3, autonomous driving, Jetson AGX Xavier, post-processing

I. INTRODUCTION

Recently, the development of deep learning algorithms and graphics processing unit (GPU) has accelerated the research in the field of artificial intelligence-based self-driving cars. Self-driving cars are vehicles that can drive safely without driver intervention. Therefore, it is necessary to accurately detect pedestrians, cars, etc. in real time to establish the appropriate control decision making to ensure safety [1]. To detect such objects, it is common for autonomous driving cars to use various sensors such as cameras, LiDARs, and radars [2]. Among these various sensors, the camera sensors are able not only to accurately identify a type of an object based on texture and color features, but are also the most cost-effective among other types of sensors [3]. In recent years, deep learning-based object detection using camera sensors has become more important in autonomous vehicles owing to the higher detection accuracy comparing against humans. Consequently, it has become an indispensable method in autonomous driving systems [4].

In self-driving cars, ensuring the appropriate real-time detection speed while maintaining high accuracy is essential for reducing latency and obtaining a rapid response from vehicle control units. However, battery-based autonomous vehicles typically operate in embedded systems, because they have to reduce power consumption. As a result, there is a difficulty in detecting objects using limited resources. Although the two-stage object detector shows high accuracy, it is unsuitable for autonomous driving applications, because it has very low detection speed within embedded platforms due to its high complexity. Therefore, a one-stage object detector with fast detection speed and high efficiency is actively used in the research on self-driving vehicles based on embedded systems [1]. A one-stage object detection algorithm, which is, however, less accurate than the two-stage method, generally uses deep network layer structures to improve accuracy, which results in a drastic increase in computation cost. Therefore, considering an embedded platform for a self-driving vehicle with limited resources, it is difficult to support real-time operation even though deep learning-based one-stage detector is faster and more efficient than the two-stage method. To mitigate this problem, lightweight deep learning-based object detection algorithms that can be supported on embedded platforms have been proposed [5]. These algorithms focus on significantly improving the detection speed by reducing the computational complexity, thereby enabling the use of deep learning algorithms in embedded platforms. However, there is a problem of accuracy loss comparing to the conventional object detection algorithms. Therefore, it is important to improve accuracy to utilize these lightweight algorithms in autonomous driving systems, where stability ensured by the accurate object detection is critically important.

This paper proposes a new object detection algorithm suitable for the autonomous driving embedded platforms, which is based on the lightweight version of YOLOv3 (*i.e.,* tiny YOLOv3 [5]), which is known to be the most suitable algorithm for autonomous driving [6]. Tiny YOLOv3 can detect multiple objects with single inference similarly to YOLOv3, and it is lighter than the YOLOv3 network, which allows significantly reducing the network of 75 convolutional layers to 13, and therefore, enabling the fast detection speed. However, as the accuracy is relatively lower than that of the conventional YOLOv3, it is essential to improve accuracy while maintaining the fast detection speed.

This paper proposes the method for predicting localization uncertainty through Gaussian modeling based on tiny YOLOv3. Through this method, the accuracy can be improved owing to the loss attenuation [7]. Furthermore, this paper

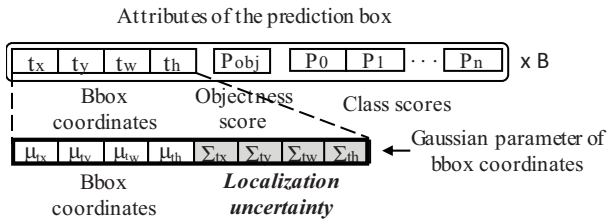

Figure. 1. Attributes of the prediction box of the proposed algorithm. B refers to the number of anchor boxes.

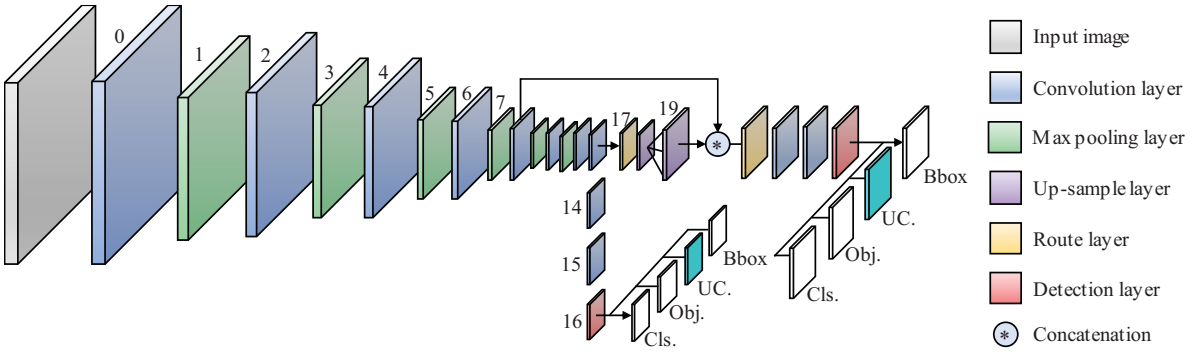

Figure. 2. Network architecture of the proposed algorithm. UC., Obj., and Cls. refer to uncertainty, objectness score, and class scores, respectively.

proposes a method for utilizing the localization uncertainty during post-processing. By using the predicted localization uncertainty at the confidence calculation step and non-maximum suppression (NMS) step, the object bounding box (bbox) candidates, which are of high confidence for all three factors (*i.e.,* objectness, class, and bbox), can be detected. Through this method, the false positive (FP) is reduced, and the true positive (TP) is increased, and thereby, the overall accuracy is improved. The proposed method applies Gaussian modeling only to the bbox of the detection layer (*i.e.,* the output layer) in tiny YOLOv3, so the computation complexity of the hidden layer is the same as that of the original tiny YOLOv3. In addition, the localization uncertainty predicted through Gaussian modeling is used at the original post-processing step; thus, the computational increase is small, and the penalty related to the detection speed can be alleviated.

As a result, the proposed algorithm, the tiny Gaussian YOLOv3, provides the detection speed of 55.56 fps for the BDD dataset, and 69.74 fps for KITTI dataset under the mode 0 of the Jetson AGX Xavier, an autonomous driving embedded platform. Therefore, real-time detection for autonomous driving is still possible. Compared to the tiny YOLOv3, the proposed tiny Gaussian YOLOv3 allows improving accuracy by 2.62 mAP and 4.6 mAP for the BDD and KITTI datasets, respectively, and therefore, it can considerably mitigate the low accuracy, which is a deficiency of the tiny YOLOv3.

II. Gaussian Modeling on Tiny YOLOv3

Using the structural similarities of the object detection mechanisms of YOLOv3 and tiny YOLOv3, it is possible to apply the method proposed in Choi *et al.* [6] to perform prediction of the bbox uncertainty in tiny YOLOv3. A prediction feature map of the tiny YOLOv3 has three prediction boxes per a grid, and each prediction box is composed of objectness score, class scores, and bbox coordinates. As shown in Fig. 1, each bbox coordinate in the prediction feature map of tiny YOLOv3 is modeled as a Gaussian parameter (*i.e.,* the mean (μ) and the variance (Σ)) to predict the uncertainty of bbox. That is, the outputs of bbox are μ_{tx}, μ_{ty}, μ_{tw}, μ_{th}, Σ_{tx}, Σ_{ty}, Σ_{tw}, and Σ_{th}. Considering the mechanism of the detection layer in tiny YOLOv3, μ_{tx}, μ_{ty}, and the variance of all coordinates are preprocessed using a sigmoid function. As a result, the mean of each bbox coordinate of the detection layer becomes the bbox coordinate of the predicted bbox, and each variance represents the uncertainty of each coordinate. Uncertainty information in each bbox coordinate that could not be known by tiny YOLOv3 can be used to cope with the mislocalization, because it can represent the confidence of the bbox itself.

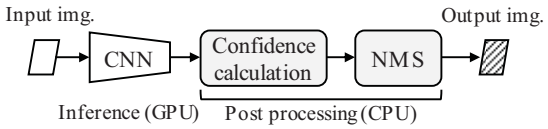

Figure. 3. Detection process of the proposed algorithm.

The tiny YOLOv3 uses the sum of squared error loss for the bbox coordinates. However, the proposed algorithm redesigns the loss function of the bbox coordinates to negative log likelihood (NLL) loss [6], because the bbox coordinates are produced as the outputs as Gaussian parameters. The loss function of objectness and class are binary cross-entropy loss, which is same as tiny YOLOv3. During the training process, the reconstructed loss function of bbox can provide a penalty to the loss through the observation noise parameter (*i.e.,* uncertainty) for the inconsistent data. Through this loss attenuation [7], the accuracy of the bbox regression can be improved.

The network of the proposed algorithm follows the tiny YOLOv3 structure as shown in Fig. 2, and Gaussian modeling for the uncertainty prediction of the bbox is applied only to the bbox information of the detection layer as presented in Fig. 2. Therefore, the computation cost of the algorithm is increased insignificantly. In the BDD and KITTI datasets, the computation cost increases only by 0.1% as compared against that of before Gaussian modeling. Therefore, the penalty of the detection speed due to Gaussian modeling is extremely low. The detailed results are shown in Section IV-B.

III. Utilization of the Localization Uncertainty during Post Processing

A. Confidence Calculation

As shown in Fig. 3, the detection process of the proposed algorithm is the same as that of tiny YOLOv3. After the GPU performs a convolution operation on the input image, the CPU executes post-processing on the result of the GPU operation and generates the final detection result as the output. During the confidence calculation, as shown in Fig. 3, tiny YOLOv3 filters the predicted candidates based on the product of the objectness score, which determines whether an object exists in the bbox, and the class score, which determines the category of the object. However, as the objectness score that indicates the existence of an object in a bbox does not represent the confidence of the bbox well, it is difficult to cope with the mislocalization of the bbox using these two criteria alone. The uncertainty of the bbox effectively represents the reliability of the bbox [6], and the proposed algorithm can predict the uncertainty of the predicted bbox as described in Section II,

Algorithm 1 Proposed post-processing method using localization uncertainty

Input: $B = \{\}$, $S = \{\}$, $U = \{\}$
B: Predicted boxes
S: Predicted scores
U: Uncertainty of each coordinate
Output: D, S, U
$D = \{\} \rightarrow$ Final detection boxes
———— **Confidence calculation step** ————
for *Every grid* **do**
  $s_i \leftarrow obj_i \times cls_i^m \times (1 - u_i^{max})$
  If $s_i > thres.$ **then**
    $B \leftarrow B \cup b_i$
    $S \leftarrow S \cup s_i$
    $U \leftarrow U \cup u_i$
  end if
end for
———— **Non-maximum suppression step** ————
for $B \neq empty$ **do**
  $m \leftarrow$ argmax S
  $M \leftarrow b_m$
  $D \leftarrow D \cup M$
  $B \leftarrow B - M$
  for b_i *in* B **do**
    if $IOU(M, b_i) > N_t$ **then**
      $s_i \leftarrow s_i \times (1 - IOU(M, b_i)) \times (1 - u_i^{remain.})$
    end if
  end for
end for

TABLE I. RESULTS OF ANCHOR BOX FOR TRAINING SETS

	Anchor 0	*Anchor 1*	*Anchor 2*
BDD training set			
First Detection Layer	(32,97)	(57,64)	(92,109)
Second Detection Layer	(7,10)	(14,24)	(27,43)
KITTI training set			
First Detection Layer	(45,76)	(27,172)	(67,116)
Second Detection Layer	(13,30)	(23,53)	(17,102)

consequently, it is possible to use the localization uncertainty in the confidence calculation. In this process, only the maximum value of the uncertainty of each coordinate is used in the confidence calculation, and if any of the coordinates has a low reliability, it is excluded from the prediction candidate. Through this method, the predicted candidates, which are of high reliability in terms of all three factors (*i.e.,* objectness, class, and bbox), can be selected, thereby, reducing FP and improving accuracy. This process is shown in the confidence calculation step of Algorithm 1, which describes the proposed post-processing technique. As it can be seen from this, the computational overhead is very small, because only one multiplication operation is added in the existing loop.

B. Non-Maximum Suppression

The NMS, which is an essential post-processing method for object detection, is a process for selecting only one bbox among several predicted bboxes of an object. NMS of tiny YOLOv3 is a greedy NMS method, which calculates an intersection over union (IOU) value of the bbox with the highest confidence score and the rest bboxes, and excludes the rest bboxes from the final candidate list, if the IOU value is greater than a specific overlap threshold (*i.e.,* N_t). In other words, when being higher than a certain threshold, the confidence score of the neighboring bbox is set to zero. Therefore, if two objects actually overlap, and the IOU value of bboxes for these two objects is higher than the overlap threshold, the greedy NMS method is not able to handle such situation, resulting in reducing TP and consequently, in accuracy loss.

TABLE II. COMPARISON OF ACCURACY AND FLOPS

	mAP (%)	FLOPs (GPU)	Input size
BDD test set			
Tiny YOLOv3 [5]	5.94	8.26×10^9	512×512
Gaussian modeling	7.70	8.27×10^9	512×512
Post-processing	8.56	8.27×10^9	512×512
KITTI validation set			
Tiny YOLOv3 [5]	64.1	8.25×10^9	512×512
Gaussian modeling	65.1	8.26×10^9	512×512
Post-processing	68.7	8.26×10^9	512×512

To address this problem, the proposed method utilizes IOU values and the uncertainty of the current bbox and the neighboring bboxes in the NMS process. When calculating the confidence score in NMS, a penalty is imposed on the confidence score according to the IOU value. If the IOU exceeds a certain overlap threshold, the adjacent candidates are not completely eliminated, and the confidence score of the adjacent candidate is readjusted according to the degree of overlap [8]. Therefore, it is possible to cope with overlapping objects. In addition, when calculating the confidence score in NMS, the confidence score is multiplied by the average of the uncertainty except the maximum uncertainty (*i.e.,* $u^{remain.}$), so that the bbox with the lowest uncertainty among the neighboring bboxes is boosted in the ranked list. This method can increase TP, thereby, improving the accuracy. This process appears at the NMS step of Algorithm 1, and as it can be seen from this, the computational overhead is very small, because only the multiplication operation for the IOU and uncertainty values is added to the existing loop.

IV. EXPERIMENTAL RESULTS

A. Experimental Environment

To demonstrate the superiority of the proposed algorithm in autonomous driving, the BDD [9] and KITTI [10] datasets are used in the experiment. For accuracy evaluation, an official IOU threshold of each dataset is applied. The IOU threshold of the BDD dataset is 0.75 for all classes [9]; and the IOU threshold of the KITTI dataset is 0.7 for cars, and 0.5 for cyclists and pedestrians [10]. In general, the mAP is low in BDD dataset because the IOU threshold is relatively high, making the dataset difficult. For training of the tiny YOLOv3 and the proposed algorithm, the learning rate is 0.0001, and the batch size is 64. Anchor boxes for training and testing are extracted through k-means clustering for each training set of BDD and KITTI as shown in Table I. The experiment is conducted on an NVIDIA Jetson AGX Xavier with CUDA 10.0 and cuDNN v7.

B. Performance Evaluation

Table II shows the resulting mAP and FLOPs values obtained for the baseline algorithm and proposed methods using the BDD test set and KITTI validation set. To perform the accuracy comparison, the official evaluation method for each dataset is used, and the IOU threshold is set to the value previously mentioned. The input size is set to 512×512 as in [6]. One of the proposed methods, Gaussian modeling, results

TABLE III. COMPARISON OF PROCESSING TIME REQUIRED IN VARIOUS OPERATION MODES OF JETSON AGX XAVIER

Mode (ID)	0			1			2			3			4			5			6		
Time (ms)	Infer.	Post.	Total	Infer.	Post.	Total	Infer.	Post.	Total	Infer.	Post.	Total	Infer.	Post.	Total	Infer.	Post.	Total	Infer.	Post.	Total
BDD test set																					
Tiny YOLOv3 [5]	14.38	0.82	15.20	61.03	1.59	62.62	26.94	1.56	28.50	20.54	1.45	21.99	20.41	1.26	21.67	20.36	1.05	21.41	20.70	0.95	21.65
Proposed	14.98	3.02	18.00	61.94	6.54	68.48	27.05	8.00	35.05	22.23	7.55	29.78	22.06	6.60	28.66	22.18	5.29	27.47	21.75	4.58	26.33
△	+0.60	+2.20	+2.80	+0.91	+4.95	+5.86	+0.11	+6.44	+6.55	+1.69	+6.10	+7.79	+1.65	+5.34	+6.99	+1.82	+4.24	+6.06	+1.05	+3.63	+4.68
KITTI valid set																					
Tiny YOLOv3 [5]	13.90	0.09	13.99	60.80	0.18	60.98	26.04	0.15	26.19	20.13	0.15	20.28	19.98	0.13	20.11	20.06	0.14	20.2	20.19	0.12	20.31
Proposed	14.16	0.18	14.34	61.02	0.38	61.40	26.31	0.32	26.63	20.65	0.30	20.95	23.81	0.24	24.05	20.49	0.21	20.7	20.41	0.20	20.61
△	+0.26	+0.09	+0.35	+0.22	+0.20	+0.42	+0.27	+0.17	+0.44	+0.52	+0.15	+0.67	+3.83	+0.11	+3.94	+0.43	+0.07	+0.50	+0.22	+0.08	+0.30

in improving accuracy by 1.76 mAP for the BDD test set, and 1 mAP for the KITTI validation set compared against the tiny YOLOv3. In addition, applying localization uncertainty during post-processing step (*i.e.,* confidence calculation and NMS), accuracy is improved further by 2.62 mAP and 4.6 mAP in the BDD and KITTI datasets, respectively. Nevertheless, GPU FLOPs for both BDD and KITTI datasets only increase by 0.12%.

Table III shows the inference, post-processing, and total detection time of the tiny YOLOv3 and the proposed algorithm under the various operating modes of Jetson AGX Xavier using the BDD and KITTI datasets. Jetson AGX Xavier provides various operation modes according to the operating frequency and the power budget, so that resources can be used efficiently according to the specific application. Under the mode 0, which optimally uses resources among several different modes, the proposed algorithm requires only a very small amount of additional time of 0.6 ms for BDD and 0.26 ms for KITTI, respectively, considering the GPU inference compared against tiny YOLOv3. Even if the post-processing time of CPU is included, only a small amount of time is added, namely, 2.8 ms for BDD, and 0.35 ms for KITTI. As a result, the total detection speed of the proposed algorithm is 55.56 fps and 69.74 fps for the BDD and KITTI datasets, respectively. That is, the proposed algorithm is still capable of performing real-time detection to support autonomous driving similarly to the baseline algorithm. It takes more time in the case of the BDD dataset than for that of KITTI, because the number of classes is larger for BDD. Based on these experimental results, it can be concluded that the proposed tiny Gaussian YOLOv3 can significantly improve accuracy at the expense of minor detection speed increase compared to the tiny YOLOv3 under autonomous driving embedded platforms.

C. *Visual Evaluation*

Fig. 4 shows the detection results of the baseline and proposed algorithms on the BDD test set. The detection threshold is 0.5, which is the default test threshold for tiny YOLOv3. The results presented in the first and second columns of Fig. 4 show that tiny Gaussian YOLOv3 can identify an object that tiny YOLOv3 is not able to find (TP), and thereafter, it can perform bbox regression to fit the objects. In addition, the third column of Fig. 4 shows that tiny Gaussian YOLOv3 can compensate for an object that tiny YOLOv3 identifies incorrectly (FP).

V. CONCLUSION

To compensate the fact that the previous object detection studies have not considered the embedded platforms for self-driving vehicles, this paper proposes the method that can significantly improve accuracy at the minor expense of the detection speed compared to the baseline algorithm (*i.e.,* tiny YOLOv3) within the autonomous driving embedded platform, Jetson AGX Xavier. The proposed method applies Gaussian modeling onto tiny YOLOv3 and utilizes the predicted localization uncertainty during the post-processing step. As a result, the proposed algorithm, tiny Gaussian YOLOv3, allows improving the mAP by 2.62 and 4.6 on the BDD and KITTI datasets, respectively, comparing against the baseline algorithm, with only minor additional latencies of several milliseconds. Therefore, the proposed algorithm is still capable of performing accurate real-time detection for autonomous driving in the Jetson AGX Xavier. In conclusion, the proposed algorithm can enable significant improvement in camera-based object detection algorithms considering autonomous driving embedded platforms.

Figure. 4. Detection results of the baseline and proposed algorithms on the BDD test set. The first and second rows show the detection results of tiny YOLOv3 and tiny Gaussian YOLOv3, respectively.

ACKNOWLEDGMENT

This work was supported in part by "The Project of Industrial Technology Innovation" through the Ministry of Trade, Industry and Energy (MOTIE) (10082585,2017) 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] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2017, pp. 129-137.

[2] C. Zhang, Y. Liu, D. Zhao, and Y. Su, "RoadView: A traffic scene simulator for autonomous vehicle simulation testing," in *Proc. Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2014, pp. 1160-1165.

[3] J. Wei *et al.*, "Towards a viable autonomous driving research platform," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2013, pp. 763-770.

[4] X. Hu *et al.*, "SINet: A scale-insensitive convolutional neural network for fast vehicle detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 8, pp. 1010-1019, Mar. 2019.

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

[6] 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.*, Oct. 2019, pp. 502-511.

[7] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2017, pp. 5574-5584.

[8] N. Bodla *et al.*, "Soft-NMS – Improving object detection with one line of code," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 5561-5569.

[9] F. Yu *et al.*, "Bdd100k: A diverse driving video database with scalable annotation tooling," *arXiv preprint*, arXiv:1805.04687, 2018.

[10] 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.*, Jun. 2012, pp. 3354-3361.