Robust Vehicle Detection Under Adverse Weather Conditions Using Auto-encoder Feature

V. D. Nguyen, D. D. Tran, M. M. Tran, N. M. Nguyen, and V. C. Nguyen

***Abstract*—Existing deep learning-based obstacle detection systems are often designed and implemented based on raw input feature. These systems obtain high accuracy under normal driving conditions. But they fail to operate under difficult driving conditions, which are different from their training. Recently, an unsupervised auto-encoder has been successfully applied to produce robust input features for a stereo matching system under difficult driving conditions. Therefore, this paper investigates an auto-encoder feature to improve the performance of existing vehicle detections under adverse weather conditions. Experimental results show that the proposed method obtained better result than existing state-of-the-art object detection methods in term of accuracy.**

***Index Terms*—Vehicle detection, auto-encoder, deep learning, and local binary pattern.**

I. Introduction

Intelligent driving assistance plays an important role to save lives by preventing accident in advance. Nowadays, driving assistance system (DAS) often provides information of traffic and road such as free spaces, obstacles and driving state of preceding vehicles [1]-[4]. Among these, vehicle detection is one of the most fundamental and challenging issues in computer vision. Recently, deep learning techniques have emerged as a powerful strategy for learning feature representation directly from data and have led to remarkable breakthrough in the field of obstacle detection. Many deep learning based obstacle detection methods have been introduced, such as region-based convolutional neural network (RCNN)[5], fast region convolutional neural network (fast RCNN)[6], faster region convolutional neural network (Faster RCNN)[7], You-Only-Look-One (YOLO)[8], DenseNet [9], FPN [10], Mask RCNN [9], RetinaNet [12], and CornetNet [13]. Without loss of generality, the deep learning-based object detections are categorized in two main categories: two-stage based method and single-stage based method. Two stage-based method includes proposal generation and detection steps, while a single-stage based method does not separate the process of the detection proposal. Existing deep learning-based object detection methods work well under normal conditions, but their performance decreases under difficult driving conditions, such as night, tunnel, rain or snow. Recently, Nguyen *et al.* has introduced a 3-channel pattern to detect vehicles under various road conditions [14]. However, this 3-channel pattern was computed by using a hand-design feature-based method, such as Local Binary Pattern (LBP), and Local Ternary Pattern (LTP). Therefore, it cannot work well under hostile driving conditions, where LBP and LTP fail to extract robust features from the input image. Fig. 1 shows experiment results of the proposed system under difficult conditions with low light. In this experiment (Fig. 1) the proposed system shows better performance than Faster R-CNN and YOLO. Recently, Nguyen *et al.* have been successfully to investigate unsupervised deep learning-based auto encoder to improve the performance of stereo matching under various driving conditions [15]. Motivated by the previous research [14], [15], this study introduces a deep learning-based auto-encoder method to compute 3-channel patterns that helps to improve the performance of existing deep learning-based object detections under hostile driving conditions. The main contributions of this research are as follows: (1) This research is the first one to apply unsupervised approach to develop a robust transformation features for a robust deep learning-based obstacle detection. (2) We successfully integrate the proposed transformation feature to improve the performance of Faster RCNN and YOLO under difficult driving conditions. In addition, the proposed method also achieved better performance than the deep learning-based object detection using multiple local patterns [14]. The remainder of this paper is organized as follows. Section II briefly reviews existing deep learning-based object detection techniques and their limitations. Section III describes the proposed deep learning system. Section IV presents experimental results. Finally, the paper concludes in Section V.

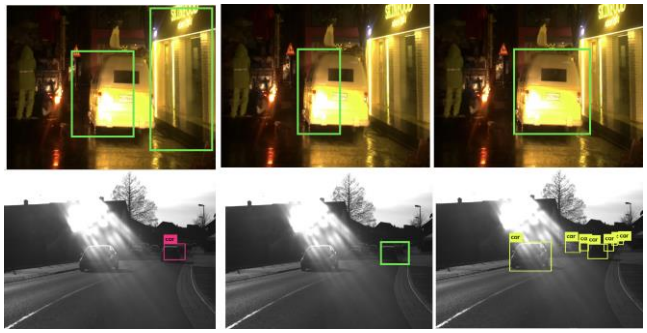

Fig. 1. Experimental results of the proposed system and existing obstacle detection systems. The first row is the results of YOLO, Faster RCNN and the proposed system, respectively, using the CCD Dataset. The second row is the results of YOLO, Faster RCNN and the proposed system, respectively, using the HCI dataset.

Manuscript received November 15, 2019; revised March 11, 2020.

V. D. Nguyen, V. C. Nguyen, D. D. Tran, M. M. Tran, and N. M. Nguyen are with the Department of Software Engineering, School of Computing and Information Technology, Eastern International University, Vietnam (e-mail: vinh.nguyen@eiu.edu.vn, vu.nguyen@eiu.edu.vn, man.tran.set15@eiu.edu.vn, duy.tran.set15@eiu.edu.vn, nhan.nguyen@eiu.edu.vn).

II. Related Work

In two-stage based method, first, a set of candidate regions are generated from the input image, a convolution neural network (CNN) is then used to extract features from these regions [16]. Next, a category-classifier is used to determine the category label of each region. Many CNN-based methods have been proposed, such as DetectorNet [17], OverFeat [18], Multiple box [19] and Region CNN (RCNN) [20]. RCNN integrates a region proposal selective search with AlexNet [21] for a generic object detection. RCNN achieves high object detection accuracy, but it still has several limitations including multiple training stages and time consuming, because CNN features need to be calculated from each object proposal in the input image.

To overcome imitations of the RCNN, many approaches have been proposed such as spatial pyramid pooling (SPPNet) [22], Fast RCNN [6], and Faster RCNN [7]. SPPNet solved the time-consuming issue of the RCNN by introducing a traditional spatial pyramid pooling (SPP) into the CNN architecture. The SPP layer is added on the top of the last convolutional layer. The processing time of RCNN is significantly reduced without decreasing its detection accuracy by using the SPPNet. However, the SPPNet is unable to update the convolutional layers preceding the SPP layer. To improve the accuracy of SPPNet, Fast RCNN is proposed by using a streamline training process that learn simultaneously a SoftMax classifier and bounding box regression. The main idea of Fast RCNN is to share the computation of convolution across region proposals. Fast RCNN is three times faster than SPPNet in training and 10 times faster in testing. The processing time of Fast RCNN is still slow because it still depends on external region proposals. Therefore, Faster RCNN was introduced by replacing the selective search by a CNN for generating region proposals.

Faster RCNN introduced an effective region proposal network (RPN) to generate region proposals. However, the processing time of Faster RCNN still depends on the number of region proposals generated from the RPN. Therefore, Dai *et al.* proposed the Region Based Fully Connect Convolution (RFCN) detector which is fully convolutional (no hidden FC layers) with almost all computations shared over the entire image [23]. RFCN obtains the same accuracy as Faster RCNN while its processing time is faster than that of Faster RCNN. The region-based methods have been successfully applied and achieved leading results on popular benchmark datasets, such as KITTI [24] and COCO [25]. Most of them are designed/improved based on Faster RCNN. However, a region-based method is still time consuming when implementing on limited hardware resources due to the limitation of memory and computational capability. Therefore, a new deep learning approach, called as a single stage method, has been introduced to overcome the limitations of a region-based methods. Single stage methods employ only a single feed-forward CNN to predict class probabilities and bounding box of an object. The stage of generating region proposals is no more need. Many deep learning-based single stage methods have been proposed, such as DetectorNet [26], OverFeat [27], YOLO, SSD [28], and CornerNet [11]. DetectorNet designed an object detection network by using the AlexNet, and substituting the SoftMax classifier with a regression layer. DetectorNet designed one network for predicting the foreground and four networks for predicting objects. DectectorNet is time consuming because it requires multiple networks for each region of interest (ROI) of the input image. To overcome the limitations of DectectorNet, a single stage object detection, called as OverFeat, was introduced based on a fully convolutional neural network. A single feed-forward fully CNN was designed in DetectorNet. DetectorNet is faster than the RCNN, but its accuracy is lower because it is difficult to train the fully CNN at that time as discussed in [4].

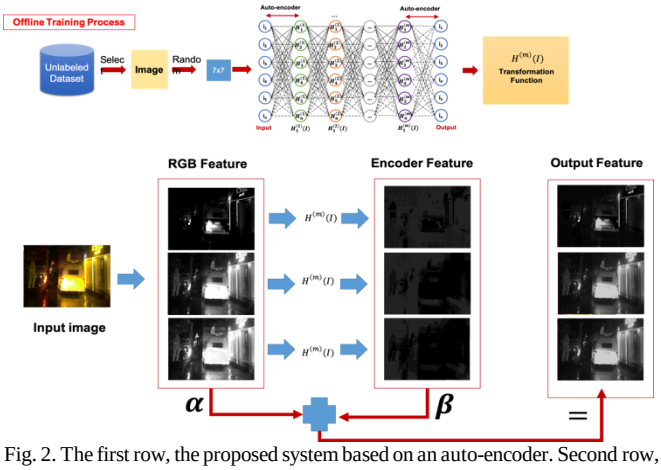

Fig. 2. The first row, the proposed system based on an auto-encoder. Second row, the work flow showing how to use the proposed auto-encoder produce a robust and stable feature in three channels.

Recently, a new object detection approach based on global features from the entire image, called as YOLO, has been proposed. YOLO predicts the object location using a small set of candidate regions (98 regions) rather than a huge number of region proposals in the Selective Search approach (more than 2000 proposals). YOLO splits the image into an SxS grid, and each of grid predict C class probabilities, B bounding locations and confidence scores. The processing time of YOLO is much faster than the existing deep learning-based region proposal systems by removing the stage of generating region proposals. However, YOLO fails to detect small objects because of assuming that each grid cell can only contain one object. More recently, newest version of YOLOV2 [29] and YOLOV3 [28] have been proposed to detect objects at multiple scales. To maintain the same accuracy as Faster RCNN, while preserving the real-time processing, SSD is then introduced. SSD combines the benefits of RPN and YOLO to obtain real-time speed and high accuracy as Faster RCNN. Motivated by the limitations of anchor boxes with a huge imbalance between positive and negative examples, CornerNet was then proposed by using the idea on Associative Embedding in pose estimation [31]. CornetNet was designed to detect object's coordinates by using top-left and bottom-right key points. CornerNet obtained better performance than the previous single stage methods. However, the processing time of CornerNet is slow with 4FPS on Titan X GPU, it is significantly slower than SSD and YOLO.

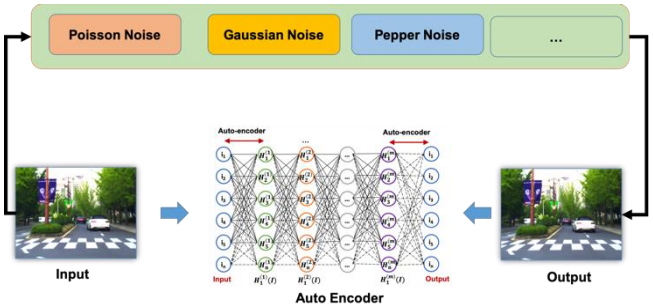

Fig. 3. The proposed auto-encoder network for learning robust features using various kind of noise.

III. Proposed Deep Learning Method

Nguyen *et al*. introduced a new approach by combining multiple local patterns (such as LBP and LTP) into a single pattern for deep learning-based vehicle detection [14]. The performance of Faster RCNN is improved under difficult driving conditions by using 3-channel patterns in [14]. However, expert knowledge is required to develop such local patterns under specific conditions. For example, LBP works well when the gray level changes monotonically, but it fails in uniform regions or with large illumination changes. Thus, the existing local patterns cannot work well under conditions which vary from those for which they are designed. In addition, a combination of unknow factors, such as rain, snow, and sun, is significantly challenging to develop a robust feature transformation function. More recently, Nguyen *et al*. have been successfully applied an unsupervised auto-encoder to design a robust feature transformation function for a robust stereo matching system [15]. Motivated by the work in [15], this study aims to design and develop a deep learning-based auto encoder for improving the performance of existing deep learning-based vehicle detection systems. This section first briefly reviews an unsupervised auto-encoder algorithm, and describes how it can learn abstract features from the input. After that, the proposed deep learning-based vehicle detection is then introduced by using the benefits of an auto-encoder.

A. Auto-encoder

An auto-encoder is an unsupervised algorithm that tries to learn an approximation of a density function by setting the input value equal to the output value [32]. Fig. 2 shows the architecture of a basic auto-encoder neural network including a hidden layer $H^{(1)}(I)$, and an output layer. $I = \{i_1, i_2, \ldots, i_n\}$ are inputs to the training network. The auto-encoder tries to learn a function $H^{(1)}(I) \approx I$, which is computed as follows:

$$H^{(1)}(I) = \frac{1}{1+e^{-(W^{(1)}*I+b(1))}} \quad (1)$$

where $W^{(1)}$ and $b^{(1)}$ are the weight matrix and bias vector, respectively, computed by minizining the cost function Δ_{sparse} with KL divergence [33].

A stacked auto-encoder is constructed by combining several auto-encoder layers in which the output of each layer is the input for the next layer as show in Fig. 2. $H^{(1)}(I)$ becomes the input for computing the next auto-encoder, $H^{(2)}(I)$:

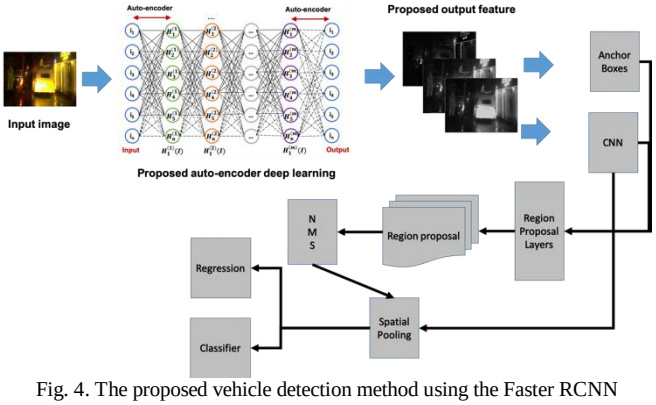

Fig. 4. The proposed vehicle detection method using the Faster RCNN architecture [5].

$$H^{(2)}(I) = \frac{1}{1+e^{-(W^{(2)}*H^{(1)}(I)+b^{(2)})}} \quad (2)$$

In summary, a stack auto-encoder [23] with m layers can be constructed as show in Fig. 2.

$$H^{(m)}(I) = \frac{1}{1+e^{-(W^{(m)}*H^{(m-1)}(I)+b^{(m)})}} \quad (3)$$

B. Proposed Auto-encoder Pattern for Robust Feature Transformation

Existing local patterns, such as LBP and LTP, are designed to work under specific conditions. Therefore, those transformations will fail to handle conditions that are differ from their original design. Motivated by the fact that an auto-encoder can help to learn robust features under various driving condition in stereo matching [15], this study aims to design a stack auto-encoder network to learn and extract robust features that becomes input for training a deep neural network. Similar to [15], this study first designs an auto-encoder network to learn a robust transformation function $H^{(m)}(I)$. However, different from [15], we design an auto-encoder network that intentionally adds some noise to the output as shown in Fig. 3. The noise is added to the output because we aim to learn the relationship between the input under normal conditions and the output under noise condition). Various types of noise, such as the Gaussian noise, salt and pepper noise, the Poisson noise, speckle noise, and illuminance changes, are used to add into the output image as shown in Fig. 3. We design the auto-encoder network with three layers (input, hidden and output layers). the input, hidden, and output units is set to 7×7 gray image patches. The training was conducted off-line to learn the robust transformation function $H^{(m)}(I)$, automatically. Thus, if the dataset used for training is diverse enough, it is easy for the proposed method to learn a robust transformation under various driving conditions.

For normal driving conditions, raw feature inputs are enough to detect and classify obstacles. However, they are not sufficient to detect and classify obstacles under difficult

driving conditions. Therefore, a new approach for generating robust input features is then introduced. Given the center pixel *p* at channel *c*, $z_p^c = (x, y, c)$, this study introduces an approach to investigate benefits of raw features along with proposed auto-encoder features as follows:

$$\Psi_N^c(z_p^c) = \alpha \times \left\lfloor \frac{\Phi\left(H^{(m)}(\{z_1^c, z_2^c, ..., z_N^c\})\right)}{N} \times 255 \right\rfloor + \beta \times I(z_p^c)$$
$$\Phi\left(z_1^c, z_2^c, ..., z_N^c\right) = \sum_{i=0; i \neq N/2}^{N} f\left((z_{N/2}^c, z_i^c)\right) \quad (4)$$
$$f\left((z_{N/2}^c, z_i^c)\right) = \begin{cases} 1, & if \quad I\left(z_i^c\right) > I\left(z_{N/2}^c\right) \\ 0, & else \end{cases}$$

where *N* is the number of neighboring input units, $H^{(m)}(\{z_1^c, z_2^c, ..., z_N^c\})$ is the result of the proposed auto-encoder using *N* neighboring input units. $I(z_p^c)$ is the raw intensity value at pixel *p* and channel c. The proposed auto-encoder patterns and raw input features have the following advantages: (1) The proposed auto-encoder extracts robust and stable features under various road condition for obstacle detection. (2) This approach can be integrated easily into existing deep learning systems to improve their performances. The α and β are designed to control how the raw and the proposed features contribute to the final robust features, respectively. These α and β are set to 0.3 and 0.7 during the training and testing phases.

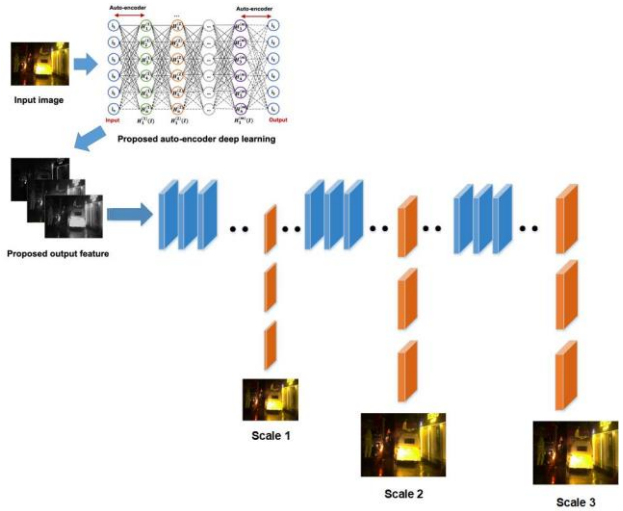

Fig. 5. The proposed vehicle detection system using YOLO on multiple scales [30].

C. Proposed Robust Vehicle Detection

We aim to integrate the proposed features $\Psi_N^c(z_p^c)$ to improve the performance of state-of-the-art deep learning-based object detection methods, including Faster RCNN and YOLO. Fig. 4 and Fig. 5 show the proposed vehicle detection system by integrating the proposed features $\Psi_N^c(z_p^c)$ to Faster RCNN [7] and YOLO [30].

For Faster RCNN, the proposed system was trained end-to-end by back propagation and stochastic gradient descent with a 4-step alternating training strategy [7]. The training process includes two stages: (1) First, for each single input image, the proposed auto-encoder pattern was applied as a pre-processing to compute the 3-channel features map before being input into the region proposal network (RPN) [5] for training. Weights were initialized using a zero-mean Gaussian distribution with a standard deviation of 0.01. The learning rate was set to 0.0001, the momentum was set to 0.9, the weight decay was set to 0.0005, and the number of iterations were set to 100,000. For training the RPN, an anchor is considered as a positive example if it has an Intersection-of-Union (IoU) ratio greater than 0.7 with one ground truth box and is otherwise considered as negative. We use non-maximum suppression (NMS) with a threshold of 0.7 to filter the proposal regions. (2) Second, for Fast R-CNN training, we construct the training set by selecting the top-ranked 20,000 proposals (and ground truths) of each image. For YOLO case, this study trains on original input images with no hard negative as mention in [30]. The multi-scale training with data augment, batch normalization was used in YOLO v3 [30]. The proposed auto-encoder pattern also was applied as a pre-processing step to compute the feature map before being input into YOLO for training. For training YOLO v3, the momentum was set 0.9, the decay was set to 0.0005, the maximum batches were set to 400,000.

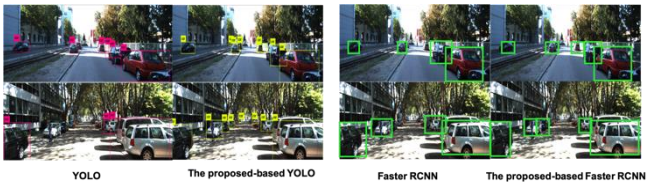

Fig. 6. Experimental results of the proposed system together with faster R-CNN and YOLO on KITTI dataset.

IV. Experimental Results

A. System Configuration and Parameter Setting

To evaluate the processing time of the proposed system, we implemented it on a computer equipped with 8 Intel Xeon 3.5 GHz CPUs, 8GB of RAM, and a Titan X GPU. We compared the performance of the proposed method to various state-of-the-art algorithms such as Faster RCNN, YOLOv3, and 3-channel patterns [14]. Faster R-CNN was re-implemented based on its available source code. In addition, the original Faster R-CNN used both VGG-16 and the Zeiler and Fergus model (ZF) to evaluate its performance. However, to make a fair comparison with regard to real-time processing, we only investigated baseNet from ZF for both Faster R-CNN and our proposed system because the ZF network is more lightweight than VGG-16. For implementation of YOLO, we used the implementation with training on the VOC dataset [30]. The 3-channel pattern was re-implement by using LBP and LTP as mentioned in [14]. Three datasets are investigated using each of these systems in order to evaluate their performance: The Karlsuhe Institute of Technology and Toyota Technology Institute (KITTI) Car

dataset [24], HCI dataset [34] and the comprehensive CCD camera dataset [14].

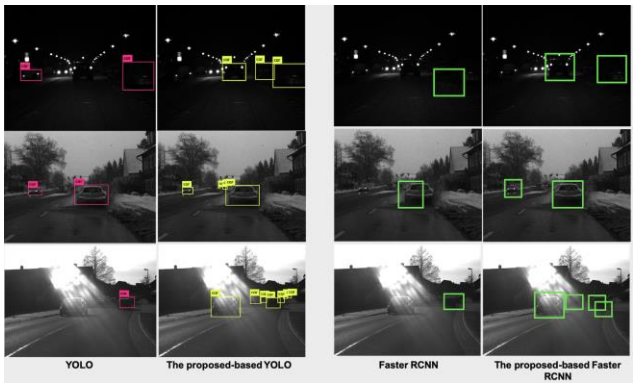

Fig. 7. Experimental results of the proposed method and Faster RCNN [7] and YOLO [30] using the HCI dataset.

B. Evaluation Method

To evaluate the accuracy of the proposed system, we use the detection rate (DR) and false alarm (FA) rate as discussed in [14]:

$$DR = \frac{\psi}{\kappa}$$
$$FA = \frac{\delta}{\psi + \delta} \quad (5)$$

where, κ is the number of real vehicles (ground truth), ψ is the number of detection vehicles, and δ is the number of false detections. A detection result is considered correct detection if the overlap region (intersection region) between its estimation and the ground truth is over 70%. Also note that, this research does not focus on improving the accuracy under various occluded situations, which is another challenging topic in vehicle detection. Therefore, we only evaluated the performance of the proposed system using the easy level of occlusions and truncations (as described in KITTI benchmark).

C. KITTI Dataset

KITTI includes 7,481 images for training and 7,518 images for testing under daylight conditions. The ground-truth is only available for the training set. Similar to [14], we split the training set into train and validation sets. We generated 10,000 images for training, and 10,000 images for testing (validation) by using the augmentation technique as in [14]. Fig. 6 shows the experimental result of the proposed-based Faster RCNN, the proposed-based YOLO, Faster RCNN [5], and YOLO [30]. Most of methods obtained a high accuracy in this test, because the KITTI dataset was captured under normal conditions. The proposed-based YOLO and the proposed-based Faster RCNN obtained better performance than YOLO [30] and Faster RCNN [7] under shadow conditions due to the benefits of the proposed auto-encoder features. Tables I and II show the comprehensive experiments on the KITTI dataset in term of detection rate and false alarm rate. The proposed-based Faster RCNN, the proposed-based YOLO, Faster RCNN [7], YOLO [30], and 3-channel method [14] obtained 98.45%, 92.67%, 92.50%, 90.55%, and 97.25% accuracy, respectively. The proposed methods still obtained the best performance under this experiment.

Fig. 8. Experimental results of the proposed system and Faster R-CNN [7] and YOLO [30] using the CCD Night dataset.

D. HCI Dataset

To evaluate the performance of the proposed method under more difficult driving conditions, we conducted more experiments by using the HCI dataset. Adverse driving conditions were captured by HCI dataset, such as blinking arrow, car truck, crossing cars, flying snow, night and snow, rain blur, rain flares, reflecting car, shadow on truck, sun flare, and wet autobahn. We generated 5,000 images for training and 5000 images for testing using the augmentation technique [14]. It is worth to note that the training and testing data do not overlap. Fig. 7 shows the experimental results of the proposed system and the others. The proposed method obtained much better performance than Faster RCNN [7] and YOLO [30] under night, snow and illumination changing conditions. Tables I and II show the comprehensive experiments on the HCI dataset in term of detection rate and false alarm rate. The proposed-based Faster RCNN, the proposed-based YOLO, Faster RCNN [7], YOLO [30], and 3-channel method [14] obtained 95.15%, 83.75%, 81.28%, 80.50%, and 93.45% accuracy, respectively. Under this experiment, the proposed methods obtained better performance than the others because the proposed unsupervised auto-encoder is able to learn and extract robust features under difficult driving conditions.

E. CCD Dataset

The CCD stereo dataset was captured under various day and night conditions, such as daylight, sunny, snowy, cloudy, rain, and nightlight conditions. This study generated 15,000 images for both training and testing(validation) phase as mentioned in [14]. Most of algorithms performed well under normal conditions because vehicle features are extracted, accurately. Table I and Table II show the comprehensive experiments on the CCD day and night dataset in term of detection rate and false alarm rate, respectively. The proposed-based Faster RCNN, the proposed-based YOLO, Faster RCNN [7], YOLO [30], and 3-channel method [14] obtained 98.75%, 93.45%, 94.07%, 89.68%, and 98.10% accuracy, respectively, under daylight conditions. To verify the performance of the proposed method under more difficult

driving condition, we evaluated the performance of the proposed system under nightlight and rain conditions, as shown in Fig. 8. The proposed-based Faster RCNN and the proposed-based YOLO obtained stable results in such difficult conditions. Table I and Table II show a comprehensive experimental result on the CCD nightlight dataset.

TABLE I: DETECTION RATE OF CAR DETECTION ON VARIOUS DATASETS

Method	CCD Day	CCD Night	KITTI	HCI
Faster RCNN	94.07	84.62	92.50	81.28
YOLO [28]	89.68	80.44	90.55	80.05
3-Channel [12]	98.10	95.58	97.25	93.45
The proposed-based YOLO	93.45	83.65	92.67	83.75
The proposed-based Faster RCNN	98.75	96.86	98.45	95.15

TABLE II: FALSE ALARM RATE OF CAR DETECTION ON VARIOUS DATASETS

Method	CCD Day	CCD Night	KITTI	HCI
Faster RCNN	1.75	2.31	1.88	2.66
YOLO [28]	1.96	2.49	2.55	2.87
3-Channel [12]	1.04	1.28	1.42	2.05
The proposed-based YOLO	1.85	2.06	2.33	2.22
The proposed-based Faster RCNN	1.02	1.07	1.15	1.28

The proposed-based Faster RCNN, the proposed-based YOLO, Faster RCNN [7], YOLO [30], and 3-channel method [14] obtained 96.86%, 93.85%, 84.62%, 80.44%, and 95.10% accuracy, respectively, under night conditions. The proposed obtained the better result than the 3-channel method [14] because its auto-encoder can learn abstract and robust input features, automatically, while the computation of 3-channel patterns has to depend on hand-designed features such as LBP and LTP.

V. CONCLUSION

This paper presents a robust vehicle detection method using the proposed unsupervised auto-encoder deep learning. The proposed system improves the performance of the existing deep learning-based obstacle detection methods, such as Faster RCNN and YOLO under difficult driving conditions. The proposed framework is flexible, easy to apply to improve another deep learning-based object detection accuracy in DAS. In future, we aim to combine both hand-design features (such as LBP and LTP) and deep learning-based features because this combination might yield promise results.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

D. D. Tran, M. M. Tran, and N. M. Nguyen collected and analyzed the data. V. D. Nguyen and V. C. Nguyen worked to develop the new deep learning model and wrote the paper. All authors had approved the final version.

ACKNOWLEDGMENT

The authors would like to express gratitude to Eastern International University (EIU), Vietnam, for their support. A warm thank to all faculty members at the Department of Software Engineering.

REFERENCES

[1] D. C. Tseng, C. T. Hsu, and W. S. Chen, "Blind-spot vehicle detection using motion and static features," *International Journal of Machine Learning and Computing*, vol. 4, no. 6, pp. 516-521, 2014.
[2] M. M. Obaida and A. S. Ma'moun, "Automated pedestrian recognition based on deep convolutional neural networks," *International Journal of Machine Learning and Computing*, vol. 9, no. 5, pp. 662-667, 2019.
[3] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 5, pp. 694–711, May 2006.
[4] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. P. Kaine, "Deep learning for generic object detection: A survey," arXiv:1809.02165 [cs.CV], 2019.
[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014, pp. 580–587.
[6] R. Girshick, "Fast R-CNN," in *Proc. ICCV*, 2015, pp. 1440–1448.
[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards realtime object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst*., pp. 91–99, 2015.
[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit*., 2016, pp. 779–788.
[9] G. Huang, Z. Liu, K. Q. Weinberger, and L. V. Maatern, "Dense connected convolution network," in CVPR, 2017.
[10] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. CVPR*, 2017.
[11] K. He, G. Gkioxari, P. Dollar, and R. Girshick. "Mask RCNN," in *Proc. ICCV*, 2017.
[12] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. ICCV*, 2017.
[13] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *Proc. ECCV*, 2018.
[14] V. D. Nguyen, D. T. Tran, J. Y. Byun, and J. W. Jeon, "Real-time vehicle detection using an effective region proposal-based depth and 3-channel pattern," *IEEE Trans. On Intelligent Transportation System*, vol. 20, no. 10, pp. 3634-3646, 2019.
[15] V. D. Nguyen, H. V. Nguyen, and J. W. Jeon, "Robust stereo matching using learning strategy," *IEEE Trans. On Intelligent Transportation System*, vol. 18, no. 12, pp. 248-258, Feb. 2017.
[16] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
[17] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection" in *Proc*. NIPS, 2013, pp. 2553–2561.
[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. ICLR*, 2014.
[19] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proc. CVPR*, 2014, pp. 2147–2154.
[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2013, pp. 580–587.
[21] A. Krizhevsky, I. Sutskever, and G. Hinton, 2012, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
[22] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. ECCV*, 2014, pp. 346–361.
[23] J. Dai, Y. Li, K. He, and J. Sun, "RFCN: Object detection via region based fully convolutional networks," in *Proc. NIPS*, 2016, pp. 379–387.
[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit*., 2012, pp. 3354-3361.
[25] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, 2014, pp. 740–755.
[26] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Proc. NIPS*, 2013, pp. 2553–2561.

[27] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. ICLR*, 2014.
[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg. "SSD: Single shot multibox detector," in *Proc. ECCV*, 2016, pp. 21–37.
[29] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. CVPR*, 2017.
[30] Redmon and A. Farhadi, "YOLOv3: An incremental improvement," in *Proc. CVPR*, 2018.
[31] A. Newell, Z. Huang, and J. Deng, "Associative embedding: End to end learning for joint detection and grouping," in *Proc. NIPS*, 2017, pp. 2277–2287.
[32] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layerwise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst*., 2007, pp. 153–160.
[33] G. Hinton, "A practical guide to training restricted Boltzmann machine," *Tech. Rep.*, Univ. Toronto, Toronto, ON, Canada, 2010.
[34] S. Meister, B. Jaehne, and D. Kondermann, "Outdoor stereo camera system for the generation of real-world benchmark data sets," *Opt. Eng*., vol. 51, no. 2, p. 021107, 2012.

Vinh Dinh Nguyen received the B.Sc. (*magna cum laude*) degree in computer science from Nong Lam University, Ho Chi Minh City, Vietnam, in 2007, and the M.S. and Ph.D. degrees in electrical and computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2012 and 2015, respectively. From 2015 to 2019, he was a research professor at College of Information and Computer Engineering, Sungkyunkwan University, South Korea. Since 2019, he has been with the Department of Software Engineering, School of Computing and Information Technology, Eastern International University, Vietnam as a lecturer and researcher. His research interests include deep learning, computer vision, image processing, and graphics processing unit computing.

Duy Dinh Tran is currently finishing his B.Sc. degree in software engineering at the Department of Software Engineering, School of Computing and Information Technology, Eastern International University, Vietnam. His interests include software engineering, machine learning, and deep learning.

Man Minh Tran is currently finishing his B.Sc. degree in software engineering at the Department of Software Engineering, School of Computing and Information Technology, Eastern International University, Vietnam. His interests include software engineering, machine learning, and deep learning.

Nhan Minh Nguyen is currently finishing his B.Sc. degree in software engineering at the Department of Software Engineering, School of Computing and Information Technology, Eastern International University, Vietnam. His interests include software engineering, machine learning, and deep learning.

Vu Cong Nguyen received the B.Sc. degree in mathematics and computer science from the University of Science, Ho Chi Minh City, Vietnam, in 1991, and a Ph.D. degree in computational chemistry from Innsbruck University, Austria in 1996. Since 2014 he has been with the Department of Software Engineering, School of Computing and Information Technology, Eastern International University, Vietnam as a lecturer and researcher. His research interests include software engineering, data science, machine learning, and computer vision.