Robust object detection under harsh autonomous-driving environments

Youngjun Kim | Hyekyoung Hwang | Jitae Shin

Abstract
In the autonomous driving environment, object instances in an image can be affected by various factors such as camera, driving state, weather, and system component. However, the deep learning-based vision systems are vulnerable to perturbation, which contains noise. Thus, robust object detection under harsh autonomous-driving environments is a more difficult than the generic situation. In this paper, it is found that not only the accuracy, but also the speed of the non-maximum suppression-based detector can be degraded under harsh environments. Therefore, object detection is handled under a harsh situation with adversarial mechanisms such as adversarial training and adversarial defence. Adversarial defence modules are designed to improve robustness in feature extraction level and define perturbations under a harsh environment for training object detectors to improve the robustness of the model’s decision boundary. The proposed adversarial defence and training mechanisms improve the object detector in both accuracy and speed. The proposed method shows a 43.7% mean average precision for the COCO2015 dataset in generic object detection and 39.0% mean average precision for the BDD100K dataset in a driving environment. Furthermore, it achieves a real-time capability of 23 frames per second.

1 INTRODUCTION

Autonomous driving is applied in various situations based on several factors such as the weather, camera sensors and systemic components etc. Most of previous studies on object detection for autonomous drive [1–3] had been conducted under clear and low-noise conditions and few of them [4–7] considered difficult situations, simply in terms of weather or illumination differentiation. However, the actual autonomous driving environment has more obstacles than weather and illumination. For example, noises from the camera, motion blur based on driving state, object characteristics in the acquired images [8] and so on. Therefore, our study classifies the obstacles of autonomous-driving situations into four categories and designs an object detection model that maintains a robust performance even under such circumstances.

There are three well-known pre-requisites to build a robust object detector for autonomous driving. First, the model should exhibit high performance so that it can recognise the scene (e.g. pedestrian, cars, and traffic lights) correctly. Second, the model requires real-time speed monitoring and stable accuracy. Third, to enhance the model’s stability, a consistent performance is required under various driving conditions. Owing to the trade-off between speed and accuracy, improving both simultaneously has been one of the most challenging tasks. Although deep learning-based object detectors exhibit high performance in an autonomous-driving environment, applying them under various noise types is another challenge [9]. These difficulties are mainly attributed to the fact that deep learning-based computer vision systems are vulnerable to noise. Thus, we propose an object detector that is accurate, fast, and robust to perturbation. Our contributions are listed as follows:

- We define four categories for distinguishing the difficult situations of a driving environment—noise from camera, physical characteristics of the object itself, driving conditions, and the system limitations of image acquisition—and apply them to object detection.
We propose an adversarial defence module (ADM) and apply adversarial training (AT) to improve both speed and accuracy of the object detection network in a harsh driving environment.

We evaluate the proposed method with state-of-the-art networks in terms of both speed and accuracy, using COCO2015 and BDD100K datasets.

We propose a new metric to evaluate the model's robustness in terms of speed.

The rest of this paper is organised as follows. Section 2 introduces the concepts of feature extraction, harsh environment, robust deep-learning system, and fast object detection as related work. Section 3 presents our robust object detector in a harsh environment by utilising the proposed modules and the proposed training method. Section 4 presents the experimental results, and Section 5 concludes the study.

2 | RELATED WORK

2.1 | Object detection

Deep learning based object detectors are divided into two main streams: (1) two-stage detectors which produce object candidate regions and rectify them to detect objects, and (2) one-stage detectors, which omit the phase of the candidate region formation. During the early stage of object detection, two-stage detectors exhibited high accuracy with slow speed, while one-stage detectors showed high speed with moderate accuracy.

However, with extensive research, one-stage object detectors [10–12] have begun to perform as well as two-stage detectors [13]. The related studies focused on enhancing the features based on the traditional single-shot object detector (SSD) or changing the heuristics parameter settings in SSD into automated ones. However, although the modifications obtained high accuracy, they suffered a loss in speed and had limited scalability. As one of the greatest strengths of one-stage detectors was fast speed, losing it was fatal. Furthermore, the lack of scalability disturbed the generalization of the model. The authors in [14] proposed a scale transferable detection network (STDN), which shows high speed and good accuracy while maintaining the model's scalability. The backbone network of STDN is DenseNet, which uses successive feature concatenation so that it has high dimensional features and requires larger computation cost. However, the authors applied pixel shuffle to lighten the features and built a detection network in the form of SSD. Therefore, it is similar with the basic SSD, but the use of DenseNet, rather than ResNet or VGG, as the backbone network utilises the advantage of preserving low-level features. However, as it is a model that simply lightens the heavy features, there is still much room for performance improvement. Therefore, we choose STDN as our baseline model.

2.2 | Feature extraction

Feature extraction has been an important topic in deep learning-based computer vision, because extracting a suitable feature map for the model is an essential part of designing a high-performance network. There are three methods for extracting features from input images.

The first method is to sample features through the convolution operation on input image. For examples, Inception v3 [15] and Atrous Spatial Pyramid Pooling [16] had been commonly used in the early-stage research in this field. Deformable convolution [17] appeared later followed by the most recent receptive field block [18]. Another approach is to reinforce the sampled feature. This method is divided into two sub-methods: pre-feature reinforcement and post-feature reinforcement. The former enhances the features before they are fed into the backbone network of the model. Stem block (SB) [15] was used in the early stage of this research field, followed by AutoEncoder [19] and attention mechanism [20]. The latter enhances the feature maps extracted from the backbone network. The representatives of this approach include pixel shuffle [14], deconvolution [21] and multi-level feature pyramid [10].

Finally, an alternative approach of feature extraction is applying SB, which plays a completely different role from the above-described one. The purpose of SB [15] was to strengthen the features, but SB can be utilized to reduce the size of the entire network. Allowing the input to go through a small network, SB, before it enters the backbone network can revise the size of the network itself. Therefore, it is possible to save computation cost of the overall network size as achieved in [22].

Based on the above mechanisms, we propose a robust object detector for various driving situations.

2.3 | Harsh environment and robust deep-learning system

There have been many attempts to solve the robustness problem of deep learning-based object detectors such as using an adversarial defence [9, 23], improving the object detection performance itself [2], or revising learning strategies to consider a noisy environment [7, 24]. In particular, modifying training strategy for considering the noisy environment has mainly focused on weather-related situations. For example, snow, rain, fog, frost can disturb object detection in autonomous driving.

Conversely, adversarial defence [8, 9] has been used to study the vulnerability of deep learning systems and safety-sensitivity in a driving environment that is found in the physical world. Authors of [8, 9] insisted that spatial conditions, physical limitations, and fabrication errors should be considered along with the environmental conditions. Here, the spatial condition is the scene understanding obtained by the camera, and the physical limitation is the performance of the sensor used in the autonomous-driving system or the physical characteristics based on the movement of object [4]. Fabrication errors include incomplete conditions such as worn or bent traffic signs or...
lights. In addition, it has been claimed that obtaining information from target objects and backgrounds is important for safe self-driving system. Because such information keep varies with environmental changes, such as variation of angle, distance, and illumination contrast [6]. Finally, if diverse sensors are used in the deep learning system, each sensor would have different conditions for achieving optimal performance [25, 26]. Therefore, such physical limitations have to be considered if the system is not built with uniform sensors.

However, the previous defined categories in [8] mainly focused on traffic sign recognition. Thus, the criteria for category division was more related to traffic sign recognition rather than object detection. Thus, new criteria are required to divide the environment related to object detection in the overall autonomous-driving situation. In addition, the classification of previous studies had several ambiguous situations. For example, weather and illumination are included in the environment condition, but they also can be included in the spatial condition for the scene understanding. Therefore, we propose a definition for a harsh environment in autonomous driving without ambiguity, including all environments considered in the previous studies. In addition, most previous studies related to object detection with harsh environments only considered accuracy to evaluate the model performance. However, unlike previous studies, we conducted an experiment to show that harsh environments can affect both speed and accuracy, and we proposed an evaluation metric for those characteristics.

3 | PROPOSED ROBUST OBJECT DETECTION IN A HARSH ENVIRONMENT

This section describes our proposed system for solving the object detection problems under a harsh autonomous-driving environment. The architecture of the proposed object detector has two phases. The first phase is adversarial defence module (ADM), which is composed of adversarial defence mechanism and reduces the computation cost and provides a robust multi-scale feature extraction in harsh environments. The other phase is object detection with two-way DenseNet based on the features obtained from the first phase. The technique for optimization of computation cost for deep learning is included in both ADM and two-way DenseNet. Figure 1 shows the overall architecture of the proposed object detection model.

In this section, after explaining the details about ADM and two-way DenseNet, we will discuss the concept of adversarial mechanism to explain how ADM includes such factors. In addition, adversarial training (AT) will be presented to provide additional robustness on harsh self-driving situations. Moreover, we redefine the rough conditions of autonomous driving, which include the characteristics of the physical world, and combine them with the proposed object detector to achieve more robustness in a real-world situation.
Two-way DenseNet in object detection

is directly followed by a decoder without any enlargement of dilation ratio makes output features sparser. As the Branch 1 of batch normalization-dilated convolution) and an increase of enlarged with the application of the BC-D block (the block is known to work well in relatively deep networks [27].

ber of dimensions, followed by ReLU-batch normalization-Branch 1, exhibits point-wise convolution to reduce the num-

In addition, we design a real-time object detector by setting the SB output. It has five branches, i.e. Branch 1 to

3.1 | ADM phase

The ADM phase, the first phase of the proposed object detector, is divided into two parts: Stem block and multi-scale auto-

encoding (MSA) system with feature fusion (FF). The former reduces the network size and extracts the feature maps that are robust to the harsh environment. The latter creates the weighted sum (WS) feature, which is a weighted sum of the decoded multi-

scale features and combines the WS feature with the SB output via feature fusion. This allows the model to create more complex and informative features and Figure 2 shows a detailed structure of the module.

SB is combination of convolution-batch normalization-ReLU (CBR) blocks and pooling. It is well known that using an SB reduces the input size and overall amount of model computation. Furthermore, when utilising the SB, the model can produce abundant features with less resources. Therefore, this technique has been applied to various deep convolution models such as Inception and PeleeNet. However, the detailed implementation (e.g., the number of CBR blocks and the order of CBR) varies across different studies. Thus, we implement an SB with “CBR-CBR-Pooling” since CBR is a post-activation structure that improves the performance of shallow networks [27].

In addition, we design a real-time object detector by setting the resolution of the SB output to one-fourth of the input resolution.

The MSA system performs multi-scale feature extraction on the SB output. It has five branches, i.e. Branch 1 to Branch 5 with different settings. Each branch, except for Branch 1, exhibits point-wise convolution to reduce the number of dimensions, followed by ReLU-batch normalization-convolution (RBC) blocks. RBC is a pre-activation structure that is known to work well in relatively deep networks [27].

Then, from Branch 1 to Branch 5, the view field keep enlarged with the application of the BC-D block (the block of batch normalization-dilated convolution) and an increase of dilation ratio makes output features sparser. As the Branch 1 is directly followed by a decoder without any enlargement of the view field, it extracts the most fine-grained feature. Sub-

sequently, as the features are closer to Branch 5, the extracted features are sparser. Thus, the sparsest features are formed in Branch 5. The number in the name of the BC-D block indicates the dilation rate of convolution. For example, in Figure 2, BC-D3 3 × 3 indicates a BC-D block with kernel size 3 × 3 and dilation rate 3, and BC-DS 3 × 3 indicates that with dilation rate 5. There are two additional techniques for guaranteeing the real-time ability of the model in each branch: dilated convolution and depth-wise separable convolution. Owing to the process of dilation convolution, which proceeds convolution while skipping pixels according to the dilation rate, a wider field of view and capacity-saving are possible. This allows extraction of sparser features to improve accuracy and increment of model’s speed. We also apply depth-wise separable convolution, which reduces the computational capacity by performing spatial convolution and point-wise convolution sequentially on an N × N matrix. At the end of Branch 1, the decoder is applied directly to tune the channel. Then, the outputs of Branch 2 to Branch 5 are concatenated and applied to the decoder to obtain the same number of channels as the decoder output in Branch 1. Then, since the number of channels in both feature maps is same, weighted sum can be performed between them. At this time, the weight of the feature that originates from the decoded Branch 1 is 1 and that of another feature that originates from decoding the concatenation of Branch 2 to Branch 5 is 0.1. The output of this weighted sum is called the WS feature, which has an input feature map of the same size as that of the MSA system. Finally, the MSA system do feature fusion of the SB output and WS feature, so that the model can preserve the original information.

3.2 | Two-way DenseNet in object detection phase

Once the ADM phase is completed, the output feature moves to the object detection phase. While designing the detection phase, we focus on the fact that as the number of channels in the deep learning model increases, amount of computation of the model increases exponentially. Therefore, we apply two-way DenseNet to refine the backbone network.

The two-way DenseNet separates one dense block with a large channel into two parallel dense blocks with smaller channels. This technique can improve the model’s speed almost without loss of accuracy [22]. However, an excessive use of this technique can cause performance degradation. Therefore, unlike [22], which applied the technique to each dense block, we apply it only to the first dense block of each layer to avoid accuracy degradation. The feature maps created through the backbone network are refined with channel-wise pooling and pixel shuffling.

Finally, the refined feature maps, so called final feature maps, are the input features for object detection layer that has the same structure with SSD. Our baseline network, STDN, also utilises the detection structure of SSD but modifies the output features from the backbone network. The resolutions of the output features of backbone network are 1, 3, 5, 9, 18, and 36 for each detection stage, and the corresponding numbers of channels
are 800, 960, 1120, 1280, 360, and 104 respectively. However, we increase the channels of the last two stages of final feature maps to 1440, 1664 and fix the resolution of all stages to 16, to improve the speed and accuracy. Finally, Soft-NMS [28] is applied to determine the final detection results. The architecture of the object detection phase is also shown in Figure 1.

3.3 Adversarial mechanism

Figure 3 shows an example of an adversarial attack, which disturbs the deep learning system by adding perturbation delta to a clear input. As the deep learning system is vulnerable to adversarial attacks, many studies [9, 29–31] have been conducted to tackle it. There are two main areas of this research: adversarial training (AT) and adversarial defence. The former modifies the training data with the perturbation delta, and then, uses them to train the model according to the original purpose, allowing it to learn various perturbations. This approach is intuitive, simple, and effective but the model will only learn the situations defined in advance so that it is impossible to learn unexpected situations. This limitation can be a major weakness in autonomous driving conditions where there are many unexpected situations.

The latter technique is proposed to compensate the limitation of the former. It trains the model to remove the perturbation delta in the input image and return it as a clear input. Therefore, we adopt both mechanisms to build a robust object detector for a harsh autonomous-driving environment. We set such environments as a perturbation delta and train the ADM to remove the noisy features. The detailed methods of using these mechanisms are covered in the following subsections.

3.3.1 Adversarial defence

Adversarial defence aims to erase the perturbation delta from the input image, which can be considered as noise removal. It is known that an autoencoder and GAN perform well on noise removal because they can recognise a normal input as a result of reconstruction of the input with the perturbation delta [32–35]. We design the MSA system by combining the requirement of the high-performance deep learning model with the concept of adversarial defence. The requirement of a well-performing deep learning is to extract a fine latent distribution that represents the observation distribution in the smallest dimension. Thus, the dimension reduction, such as channel reduction, can also be a way to determine it. Therefore, reducing the channel by removing the noisy features is a combination of the adversarial defence concept and the requirement of a well-performing deep learning system. Thus, we build each branch of the MSA system in the form of a stacked autoencoder to erase the noisy channel through the features and extract fine latent distributions. Furthermore, to improve the performance of the stacked autoencoder, we apply FF to the MSA system. As shown in Figure 4, the difference between each feature created from the latent distribution of the normal input and noisy input is learned through cross entropy. Therefore, our system implementation works end-to-end without any network.

3.3.2 Adversarial training for harsh autonomous-driving environments

Adversarial training (AT) is a method of adding a perturbation delta to the training data, so that the model can learn as many perturbations as possible [7, 36]. As mentioned before, this approach has clear pros and cons. It is simple and effective but cannot handle unexpected situations. Therefore, to take advantage of AT to improve the detector’s performance, a detailed situation classification is required.

Thus, we define harsh autonomous-driving situations that can occur in the real world. Among the situations that often occur while driving, we categorise situations that have a sensitive effect on driving quality. In addition, by observing the results of applying our defined situations to AT, the situations are found to directly affect the performance of the object detector. The qualitative results can be found in Appendix.

To obtain an optimal effect from AT, we classify the harsh situations of autonomous driving into four categories without ambiguity. The proposed categories and the causes of their occurrence and several phenomena are listed in Table 1.

The first category is a camera environment. The noise originating from environmental factors in a camera affects the image even when the person is not aware. As the small-noise (e.g. salt and pepper noise) that humans seldom recognise, can affect the performance of the deep learning system. So, such kind of noises should be considered. These can be usually classified as
dark current noise, readout noise, and photon shot noise. The equation for signal-to-noise ratio (SNR) that indicates camera noise parameter which evaluates the image quality is shown below:

$$\text{SNR} = \frac{PQ \cdot t}{\sqrt{PQ \cdot t + D\epsilon + N_r^2}}.$$  \hspace{1cm} (1)

Here, \( P \) indicates the incident photon flux (photons per second in the area of a pixel), \( Q \) is the quantum efficiency, and \( t \) is the exposure time. \( D \) represents the dark current of pixels, in electrons per second, and \( N_r \) is the pixel readout noise in the electrons. According to the formula, the variables that can be used as a phenomenon are \( P \), \( D \), and \( N_r \), therefore, we define them as the phenomenon in the camera category. This indicates that shot noise and dark noise are neglected if the exposure time is very short, but the readout noise still affects SNR. It means that readout noise has a relatively significant effect on the low-light condition compared to the other two noises. However, in the real world, the photon shot noise has a significant impact on SNR because the photons are not uniformly spread on each pixel of the image sensor. The degree of exposure to the pixels is expressed as Poisson distribution. This case usually occurs when the light condition is not uniform, such as night or inside of a tunnel. In addition, dark current noise is also generated by heat, which may result from long-term camera usage due to long-term driving or in high-temperature climates. Read-out noise, usually occurs in case of an analogue-to-digital conversion. According to Sony [25], a CMOS sensor in autonomous driving should be ready to work under 125 °C. However, it is very difficult to prepare this environment from the sensor itself [26], because all cameras use different sensors and each sensor has a different optimal condition. Therefore, we will solve this problem by using the deep learning technique.

The second category is a driving state. Some harsh autonomous-driving situations that occur in the driving state include motion jitter, zoom blur, and out-focus, which are mainly caused by change of object’s position. Zoom blur appears when the camera approaches the object quickly, while motion jitter originates from the rapid movement of objects, such as that in high-speed driving situation. Out-focus includes any situation that can ruin the focus of images. The third category is weather, which originates from the condition of nature, for example, snow, frost, fog, light, and moisture.

The last category is the system itself, which consists of the following three major phenomena: resolution, data format, and whitening. With the development of deep learning-based object detectors, the required image resolution specifications have also increased. Particularly, recently announced object detectors such as [11, 12], require high-definition (HD) resolution. However, camera sensors in self-driving cars are built with low resolution. Thus, up-sampling is needed to apply the image to recently proposed models. Therefore, interpolation problems appear when up-scaling the images. In addition, the file format can create a problematic situation. If the file formats for training and inference are different, the deep learning system will undergo performance degradation. For example, the most common compression formats, such as JPEG, cause performance degradation for deep learning systems that have been learned with RAW file. The other phenomenon is whitening, which originates from the moisture that fills inside of the lens and makes the screen blurry. This moisture is different from that generated from the weather. The former occurs inside of the lens, while the latter appears outside of the lens. This can also cause problems for vision systems.

Thus, it is very complicated to create optimal settings in all four categories while designing an autonomous-driving system. The reason is that the sensors used for the camera are different, so the optimal condition changes accordingly. Furthermore, the effects driving state and weather cannot be reflected exactly through physical manipulation of the system. So, the simulation method may have a limitation of the similarity gap between simulation data and real-world data. However, our applied AT enables us to learn as many perturbations as possible so that the model’s decision boundary can be extended. Furthermore, our applied ADM can extract actual features of target object under the perturbations, which can reduce the gap. Therefore, ADM and AT together can complement the limitations.

### 4 EXPERIMENT RESULTS

The COCO2015 [37] and BDD100k [1] datasets are used to train and validate the performance of the proposed object detector. The former dataset is used for general object detection, while the latter is used for object detection for driving environment. An ablation study conducted using the COCO2015 dataset shows step-by-step performance improvement of general object detection. The robustness of the proposed model’s performance in various autonomous-driving environments.
is measured using the BDD100k dataset. The experiment is conducted on Ubuntu 18.04, CUDA 10.1, Pytorch 1.1 or higher with the GPU of RTX 2080Ti and Intel i7-9600 CPU with Python version 3.6.

4.1 Experimental setting

For data augmentation, flip, rotate, and crop are used, and all images are normalised with RGB(104,117,123). Xavier initialization is performed to initialise the model. For anchors, the scale parameters are [10,3,2,4] and [16,3,2,4] for input resolutions of 321×321 and 513×513, respectively. The aspect ratios of the anchor are same as those of STDN. Due to the GPU capacity problems, the batch size is 16 in case of resolution for the 321 model and 6 for the 513 model. The minimum and maximum sizes of the anchor are 15 and 90, respectively. 

The initial learning rate is $10^{-2}$ and the gamma value is 0.9. Then, the rate is reduced to $10^{-6}$ for every 90, 200, 300, and 350 epochs. However, after the 200th epoch, the model is overfitted, and thus, we pause the training around this epoch. The training is completed around the 200th epoch for COCO2015 and 180th epoch for BDD100k. The stochastic gradient descent (SGD) is used as an optimiser, with momentum and weight decay values set to 0.9 and 0.0005, respectively. For the final detection, in Soft-NMS, 50 candidate boxes are used for each class to prevent reduction of accuracy and peak speed. The score threshold is 0.1 and the IoU threshold is 0.75; the attention ratio for each branch of the MSA system is as follows: from Branch 2 to Branch 4, the ratio of the output channel of each RBC block to the input is 0.25, and for Branch 5, this value is 0.125 for the first RBC block, 0.1875 for the second, and 0.25 for the last one. As Branch 1 does not contain any RBC block, the ratio for Branch 1 is not required.

4.2 General object detection

To observe the effect of each module of the model, we conduct an ablation study on the COCO2015 test dataset. In addition, we compare the benchmark result of COCO2015 with that of other state-of-the-art networks, to show the performance of the proposed model.

4.2.1 Ablation study

The ablation study of the proposed modules is shown in Table 2, which shows step-by-step performance of each proposed module. We choose the model size with 321, as Ours(321) (Table 3). The ablation study starts from the baseline network STDN and three modules are added to the sequence: two-way dense layer, ADM, and FF between the SB output and MSA system. The two-way dense layer aims to speed up the network while maintaining accuracy. Thus, its application to the baseline increases the speed by 2 frame-per-second (FPS) with loss of 0.2% accuracy.

Subsequently, by attaching ADM with the MSA system without FF, the mean average precision (mAP) increases by 13.1% with a little FPS loss of 0.6. The reason for this small FPS reduction is the depthwise-separable and dilation convolution, which can save the computation cost effectively while maintaining the purpose of the convolution operation. This indicates that our ADM phase successfully removes the noisy feature, and thus extracts a fine latent distribution for object detection. Furthermore, adding FF between the output of the SB and the MSA system in ADM shows 1.2% increase in mAP without any loss of FPS. Thus, FF does not require additional computation in our model.

The final model, ADM with FF, is faster and has 14.3% higher mAP compared to the baseline. The average precision at 0.75 threshold (AP@0.75) is increased by 17.5%, which indicates a large improvement in the localization ability of the network. In addition, there is 7.6% improvement in size for a small object and more than 19% improvement for medium and large objects, compared to the baseline.

4.2.2 COCO2015 benchmark

To compare the object detection performance in general situations, we evaluate the proposed model to the COCO2015 benchmark dataset. The evaluation result is shown in Table 3. The information regarding the GPU and the input size of each model is provided for a fair comparison, as the GPU and input-image size affect the speed and accuracy, respectively. In addition, the comparison between AP@0.75 and mAP provides a comparison of the localization performance among the models. Additionally, the detection performance based on the object size is divided into three sizes, where information about the usage of multi-scale testing is shown in the last column.

According to Table 3, the proposed model is comparable to other networks from the perspectives of speed, accuracy, and localization. In particular, if the object size is set to medium or large, the proposed model outperforms other state-of-the-art networks. Additionally, our model is the fastest, but has a GPU with the weakest computation power. In the case of mAP for a large object, the proposed model shows more than 4.0% better performance than state-of-the-art networks. It also has the smallest input size and is tested without multi-scale testing. Furthermore, the proposed method shows 4.3% higher inference speed than AutoFocus, which uses a multi-scale test to improve the performance and has an input size almost 13 times smaller.
TABLE 3  Benchmark results on COCO2015 with one-stage state-of-the-art networks

| Model                        | mAP  | FPS | GPU            | Input Size   | AP@0.75 | Small | Medium | Large | Multi-Scale Test |
|------------------------------|------|-----|----------------|--------------|---------|-------|--------|-------|------------------|
| SSD_Resnet101 [38]           | 31.2 | -   | -              | 513 × 513    | 33.3    | 10.2  | 34.5   | 49.8  | false            |
| Retinanet [39]               | 39.1 | 5   | TeslaM40       | 800 × 800    | 42.3    | 21.8  | 42.7   | 50.2  | false            |
| M2Det [10]                  | 37.6 | 18  | Titan XP       | 512 × 512    | 40.5    | 18.4  | 43.4   | 51.2  | false            |
| Fcos_ResNet64x_4D_101_FPN [40] | 46.6 | 7   | Tesla V100     | 800 × 1333   | 48.4    | 27.6  | 47.5   | 56.6  | false            |
| Fcos_ResNet50_FPN_1X [40]   | 38.6 | 22  | Tesla V100     | 800 × 1333   | 41.4    | 22.3  | 42.5   | 49.8  | false            |
| Fcos_ResNet50_FPN_1X_Dcnv2 [40] | 42.3 | 18  | Tesla V100     | 800 × 1333   | -       | -     | -      | -     | false            |
| AutoFocus [12]              | 47.9 | 6   | Tesla V100     | 1175 × 1175  | -       | 31.5  | 50.5   | 60.3  | true             |
| Ours(321)                   | 42.3 | 29  | Rtx2080 Ti     | 321 × 321    | 47.1    | 15.5  | 49.6   | 64.6  | false            |
| Ours(513)                   | 43.7 | 24  | Rtx2080 Ti     | 513 × 513    | 49.0    | 19.4  | 52.6   | 62.3  | false            |

times larger than that of ours. Compared to M2Det, which has a similar input size to Ours(513), the proposed model’s mAP is 6.1% higher, and even though Ours(321) has a smaller input size, its accuracy is 4.7% higher than that of M2Det.

AP@0.75 measures amount of the detected objects which overlap with the annotation box is more than 0.75%. Thus, a higher AP@0.75 indicates that the model can produce detection boxes that are more similar to the labels. Therefore, this can measure the localization capability of object detection networks. For the AP@0.75 shown in Table 3, Ours(513), which is the proposed model, shows the best performance with 49% accuracy, although it is input size is about a quarter than that of the second-best FCOS_ResNet64x_4D_101_FPN. The variation of the input size of the proposed model from 321 to 513 shows an increased performance for small and medium objects and decreased performance for large objects. Therefore, our study achieves both a fast and accurate object detector, which is compatible to state-of-the-art detectors.

4.3 Performance and robustness of proposed object detector in harsh autonomous-driving environment

To check the robustness of the proposed object detector on self-driving environments, we compare the mAP and FPS of the proposed model and state-of-the-art models for the BDD100k dataset. Table 4 summarised the performance of the proposed model in BDD100k.

| Model                        | mAP  | FPS | AT  |
|------------------------------|------|-----|-----|
| SSD512                       | 14.1 | 23.1| Off |
| Gaussian Yolo512             | 18.4 | 42.5| Off |
| Gaussian Yolo736             | 20.8 | 22.5| Off |
| RFB513                       | 14.5 | 39.0| Off |
| M2Det520                     | 20.4 | 23.1| Off |
| STDN321                      | 24.1 | 28.6| Off |
| Ours(321)                    | 36.8 | 29.7| Off |
| Ours(321)                    | 36.4 | 29.5| On  |
| Ours(513)                    | 39.0 | 24.6| Off |
| Ours(513)                    | 38.8 | 24.3| On  |

4.3.1 Object detection under driving environment

The performance on the BDD100k dataset is shown in Table 4. As the real-time capability is necessary for application of autonomous-driving environment, both mAP and FPS are used to indicate the performance of each model. Ours(513), which is the proposed model, shows the best mAP with comparable real-time speed of 24FPS.

Once AT is applied, both accuracy and speed slightly decrease, which might affect the performance in a clear (e.g. no noise) situation when using AT. The performance reduction is less than 0.4% for both mAP and FPS. Considering the effectiveness of AT for harsh autonomous-driving situations, this amount of performance reduction in a clear environment is affordable. The deceleration of the deep learning system is caused from the deceleration in the non-maximum suppression (NMS) process owing to an increased number of bounding boxes when learning the perturbation delta rather than the clean input only. For this purpose, we use the official code [41] to apply our result of M2Det to BDD100k. The STDN is implemented by us based on the paper [14]. Other detectors, such as RFB and SSD, can be referred from the Gaussian Yolo paper [2].
4.3.2 Evaluation method for robustness of object detectors on harsh environments

\[ mPC = \frac{1}{N_c} \sum_{c=1}^{N_c} \frac{1}{N_l} \sum_{l=1}^{N_l} P_{c,l}, \quad rPC = \frac{mPC}{P_{clear}} \quad (2) \]

This research aims to develop a robust object detector for a difficult autonomous-driving environment. Thus, we need metrics to measure the robustness of the object detector under various situations. Equation (2) shows the evaluation metric proposed in [8], which captures the robustness of accuracy.

The first term in the above equation is \( mPC \), which measures the accuracy in noisy environments, where \( N_c \) is the set of categories for each harsh environment and \( N_l \) is the noise level within this category. \( P_{c,l} \) is the mAP performance of the object detector at \( l \) level in \( c \) category. The second term for accuracy robustness is \( rPC \), which measures how much performance can be derived from the harsh environment compared to the performance in a clear situation. Therefore, it is calculated as a fraction between \( mPC \) and accuracy under a clear condition \( P_{clear} \).

We use these measures to evaluate accuracy robustness of the object detector. The performance according to the noise characteristics defined in this paper is identified and measured separately by each category. Furthermore, to take a closer look at the changes depending on the severity of the noise, we measure the performance based on the five noise levels. The criterion of each noise level is the same as those mentioned in [42], and the details of the five levels of each category and the corresponding phenomena are listed in Appendix. Our simulation is based on that provided in [42].

\[ mFPS = \frac{1}{N_c} \sum_{c=1}^{N_c} \frac{1}{N_l} \sum_{l=1}^{N_l} FPS_{c,l}, \quad rFPS = \frac{mFPS}{FPS_{clear}} \quad (3) \]

However, the harsh environment affects both accuracy and speed of the object detector. The robust real-time capability of the object detector is one of the most important measures in an autonomous-driving situation, which means that if it fluctuates depending on the environment, and harms both model stability and driver safety.

Therefore, we additionally propose \( rFPS \) as an evaluation metric for measuring the robustness of the model in terms of speed. It is calculated as the ratio between the model speed in a noisy environment and that in a clear environment.
Equation (3) shows the formula for rFPS similar to that for rPC, where \( F_{\text{FPS}} \) indicates the FPS of the object detector at the \( l \) level in the \( c \) category.

Figure 5, Table 5 and Table 6 show the experimental results of object detection in a harsh autonomous-driving environment. The experiment is conducted on the BDD100k dataset for a clear situation, and we create a harsh environment by simulating BDD100k to our proposed categories. In Figure 5 (A)–(D), the changes in mAP and FPS according to the noise level are drawn separately depending on whether AT is used. The \( x \)-axis indicates the noise level, and the \( y \)-axis indicates the robustness evaluated in terms of mAP and FPS. The square mark indicates mAP and the triangle indicates FPS. Notation A.T. in the figure shows the results of the model trained with the AT method. The results of (A)–(D) show different harsh conditions, mentioned in Table 1. The case of (A) indicates camera noise, (B) indicates the driving state, (C) indicates the weather condition, and (D) indicates system noise. Figure 5, shows that the harsh environment of self-driving affects both speed and accuracy of the detector and worsens as the noise level increases. Clearly, the noise can let the model be oblivious to the object by disturbing the input image. However, the reason for the decrease in speed is the increased number of bounding boxes produced per class. Then, the NMS process requires additional time to delete the incorrect bounding boxes.

Table 5 shows the rPC result of the model for the four proposed categories with information obtained with and without AT, and Table 6 shows the rFPS result of the same model for the same situations as those listed in Table 5. According to both tables, using AT provides better performance for both rPC and rFPS.

Another interesting observation is made in perspective of accuracy. Performance shows little difference, except for the camera-noise environment. This can be observed in both Figure 5 and Table 5. The difference in rPC is at least more than 3% in camera noise but at most 1.6% in other noise environments listed in Table 5. In addition, in Figure 5, only situation (A) shows a with and without AT. This indicates that our model that is trained with only a clear situation performs well with those of harsh environments in terms of accuracy. This means that our proposed adversarial noise defence module performs well even without AT. However, as most of the experimental results with AT are better than those obtained without it, it is obvious that AT has a positive effect on the model’s robustness under harsh environments. In addition, the effect of AT on the weather condition shows different aspects for the speed of proposed model. The speed is decreased by a small amount, which can be observed in environment (C) in Figure 5 and the column “Weather” in Table 6. However, this is not a huge difference, and we can still see less false detection from the demonstration of our model, as shown in the next subsection.

4.3.3 Qualitative results of the Harsh environment on BDD100k dataset

As there is no dataset containing all proposed categories and phenomena in Table 1, we have to choose several ways to simulate each phenomenon from a clear image. For the camera noise phenomenon, the noise model was already built, therefore, we put it into a clear image. In the case of driving states and difficult weather, we chose the conventional simulators used in [42]. To apply the system noise phenomenon, we chose RAW and JPEG as the different data formats, which can cause problems because of information loss during the compression of RAW to JPEG. To simulate the resolution problem, we reduced the resolution of existing data, and then up-sampled them again. For the whitening phenomenon, a mixture of fog, contrast, and glass blur effects was applied to the clear images. Figure 6 shows simulation demonstrations of our proposed harsh environment on the BDD100k dataset. We propose four categories with 14 phenomena, where each row matches with each category. The first row shows the simulation with camera noise, and the second row shows that for the driving state. The weather category appears in the third row, and system aspect appears in the last row. The phenomenon for each category listed in Table 1 appears in order from left in each row. Therefore, the first row shows three phenomena included in the camera noise in this order. In the second, third, and fourth rows, the phenomena indicated in each of the mentioned categories are displayed in order.

Figure 7 shows the qualitative result of the detectors for each category of harsh environments. From left column to right, each column shows the result obtained from STDN, Ours(513), and Ours(513) with AT. The environments applied in this demonstration are of the order of “category-phenomenon” with respect to the level of noise. The first row indicates “camera-readout noise” and second row indicates “driving
FIGURE 6 Demonstration of simulation on harsh driving environments: the first row shows three phenomena included in the camera noise in order. In the second, third, and fourth rows, the phenomena indicated in each category mentioned in Table 1 are displayed in order.

FIGURE 7 Detection results on representative harsh driving environments in Table 1: For each row, from left to right, each column shows the results obtained from STDN, Ours(513) without AT, and Ours(513) with AT, respectively. The first row indicates “state-OutFocus”, both having noise level 3. The third row indicates “weather-snow”, and the last row indicates “system-whitening” with noise level 4 for these cases. The results show a significant difference in detection between STDN and Ours(513). The first column shows that STDN only detects cars in all harsh environments and detects few pedestrians or traffic lights. By comparing the first and second column, it can be easily recognised that the results of Ours(513) captures pedestrians and traffic lights that are missed in the detection result of STDN. This is a sufficiently predictable result because the proposed method shows performance accuracy of 13% or more, regardless of whether AT is used or not. However, in the comparison between our models, the one with AT shows better detection. Comparing the second and third columns, the latter shows better detection quality, which is the result of Ours(513) with AT. Ours(513) with AT detects two traffic lights for the first three environments, whereas the Ours(513) without AT model can only detect one of them and detects pedestrians better than the former in all environments and avoids false positives that occurred in the snow environment in Ours(513). In designing the model for autonomous driving, precise detection of pedestrians and traffic lights is the most important part for the safety of drivers and non-drivers. In addition AT makes this possible in cases of more diverse and harsh situations. Therefore, even though the application of AT to the weather condition can cause a slight deceleration, its application is still suitable from the perspectives of safety and the model stability.

FIGURE 8 Detection result (upper picture) of Ours(513) with AT under clear environment. The lower picture is a ground truth. From the qualitative result of Ours(513) model with AT in a clear situation, it is obvious that the proposed model finds almost all objects accurately, except for one pedestrian on the left side of the image. This again shows that there is no significant performance degradation, even in a clear situation, when we apply the AT.

5 CONCLUSION

This paper defines four categories of harsh autonomous-driving environments, with 14 phenomena, as follows: noise from camera, driving state, weather, and system perspective. In addition, we construct a robust object detector to solve the problem of object detection in these difficult environments by using the ADM method. The proposed model achieves mAP gain of more than 14% compared to the baseline and a real-time speed performance of 24 FPS. Additionally, AT is applied to improve
the performance and speed of the proposed detector in harsh driving situations. Thus, the proposed system is comparable to state-of-the-art networks and work consistently even in difficult situations for object detection. Furthermore, we propose the rFPS metric, which evaluates the robustness of network speed irrespective of whether it is interfered with noise. In addition, we observe that the harsh environments affect the inference time of the object detector through deceleration of the NMS process, which can harm the model stability. Therefore, we suggest including the speed measurements to the robustness measure of the object detector under harsh or various other environments. In the future, we will conduct experiments to check whether the performance changes in the proposed noisy environments can be generalised via applying them to other object detectors.

ACKNOWLEDGMENTS
This research was partly supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03031752) and partly by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2018-0-01798) supervised by the IITP(Institute for Information & Communications Technology Promotion).

ORCID
Youngjun Kim https://orcid.org/0000-0002-1470-3220

REFERENCES
1. Yu, F. et al.: Bdd100k: A diverse driving video database with scalable annotation tooling. arXiv:1805.04687, (2018)
2. Choi, J. et al.: Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 502–511. IEEE, Piscataway, NJ (2019)
3. Geiger, A., et al.: Vision meets robotics: The kitti dataset. Int. J. Rob. Res. 32(11), 1231–1237 (2013)
4. Boukhiss, R.R., Fendri, E., Hammami, M.: Moving object detection under different weather conditions using full-spectrum light sources. Pattern Recognit. Lett. 129, 205–212 (2020)
5. Liu, Z., et al.: Analysis of the influence of foggy weather environment on the detection effect of machine vision obstacles. Sensors 20(2), 349 (2020)
6. Boloor, A. et al.: Simple physical adversarial examples against end-to-end autonomous driving models. In: 2019 IEEE International Conference on Embedded Software and Systems, pp. 1–7. IEEE, Piscataway, NJ (2019)
7. Michaelis, C. et al.: Benchmarking robustness in object detection: Autonomous driving when winter is coming. arXiv:1907.07484, (2019)
8. Eykholt, K. et al.: Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition, pp. 1625–1634. IEEE, Piscataway, NJ (2018)
9. Yuan, X., et al.: Adversarial examples: Attacks and defenses for deep learning. IEEE Trans. Neural Networks Learn. Syst. 30(9), 2805–2824 (2019)
10. Zhao, Q. et al.: M2det: A single-shot object detector based on multi-level feature pyramid network. In: Proc. of the AAAI Conf. on Artificial Intelligence vol. 33, pp. 9259–9266. AAAI Press, Palo Alto, CA (2019)
11. Tian, Z. et al.: Feos: Fully convolutional one-stage object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 9627–9636. IEEE, Piscataway, NJ (2019)
12. Najibi, M., Singh, B., Davis, L.S.: Autofocus: Efficient multi-scale inference. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 9745–9755. IEEE, Piscataway, NJ (2019)
13. He, K. et al.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp. 2061–2069. IEEE, Piscataway, NJ (2017)
14. Zhou, P. et al.: Scale-transferable object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 528–537. IEEE, Piscataway, NJ (2018)
15. Szegedy, C. et al.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826. IEEE, Piscataway, NJ (2016)
16. Chen, L.-C. et al.: Rethinking atrous convolution for semantic image segmentation. arXiv:1706.05387, (2017)
17. Dai, J. et al.: Deformable convolutional networks. In: Proc. of the IEEE Int. Conf. on Computer Vision, pp. 764–773, (2017)
18. Liu, S. et al., Receptive field block net for accurate and fast object detection. In: Proc. of the European Conf. on Computer Vision, pp. 385–400. Springer, Berlin (2018)
19. Baldi, P.: Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML workshop on unsupervised and transfer learning, pp. 37–49. MIT Press, Cambridge, MA (2012)
20. Wang, F. et al.: Residual attention network for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3156–3164. IEEE, Piscataway, NJ (2017)
21. Fu, C.-Y. et al.: DSSD: Deconvolutional single shot detector. arXiv:1701.06659, (2017)
22. Wang, R.J., Li, X., Ling, C.X.: Pelee: A real-time object detection system on mobile devices. In: Advances in Neural Information Processing Systems, pp. 1963–1972. MIT Press, Cambridge, MA (2018)
23. Dhillon, G.S. et al.: Stochastic activation pruning for robust adversarial defense. arXiv:1805.01442, (2018)
24. Bijelic, M. et al.: Seeing through fog without seeing fog: Deep sensor fusion in the absence of labeled training data. arXiv:1902.08913, (2019)
25. Technology, Nov 2019. [Online]. https://www.sony.net/SonyInfo/537. IEEE, Piscataway, NJ (2018)
26. Wang, F. et al.: Receptive field block net for accurate and fast object detection. In: Proceedings of the IEEE Int. Conf. on Computer Vision, pp. 5361–5569. IEEE, Piscataway, NJ (2017)
27. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access 6, 14410–14430, (2018)
28. Sinha, A., Namkoong, H., Duchi, J.: Certifying some distributional robustness with principled adversarial training. arXiv:1710.10571, (2017)
29. Chakraborty, A. et al.: Adversarial attacks and defences: A survey. arXiv:1810.00069, (2018)
30. Vincent, P., et al.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Machine Learning Res. 11, 3371–3408 (2010)
31. Su, H. et al.: Robust cell detection and segmentation in histopathological images using sparse reconstruction and stacked denoising autoencoders. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 383–390. Springer, Berlin (2015)
32. Samangouei, P., Kabkab, M., Chellappa, R.: Defense-gan: Protecting classifiers against adversarial attacks using generative models. arXiv:1805.06605, (2018)
33. Lee, H., Han, S., Lee, J.: Generative adversarial trainer: Defense to adversarial perturbations with gan. arXiv:1705.03387, (2017)
34. Tramèr, F. et al.: Ensemble adversarial training: Attacks and defenses. arXiv:1705.07204, (2017)
35. Lin, T.-Y. et al.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision, pp. 740–755. Springer, Berlin (2014)
36. Samangouei, P., Kabkab, M., Chellappa, R.: Defense-gan: Protecting classifiers against adversarial attacks using generative models. arXiv:1805.06605, (2018)
APPENDIX A

The detailed simulations of the proposed difficult autonomous-driving environments with different noise levels based on BDD100k dataset are shown in Figure A.1 through Figure A.4. They are visualized in the same order with the categories presented in the Table 1 and from the leftmost column to the right, the level of noise subsequently increases.

**FIGURE A.1** Simulation results of the camera environment. From top to bottom, each row represents the phenomenon in the same order with Table 1: Dark current noise, Readout noise, and Photon shot noise

**FIGURE A.2** Simulation results of the driving-state environment. From top to bottom, each row represents the phenomenon in the same order with Table 1: Zoom, Motion, and OutFocus
FIGURE A.3  Simulation results of the weather environment. From top to bottom, each row represents the phenomenon in the same order with Table 1: Snow, Frost, Fog, Light, and Moisture.

FIGURE A.4  Simulation results of the system environment. From top to bottom, each row represents the phenomenon in the same order with Table 1: Whiteness, Data format, and Resolution.