IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Lane Segmentation Data Augmentation for Heavy Rain Sensor Blockage using Realistically Translated Raindrop Images and CARLA Simulator

Jinu Pahk[1], Seongjeong Park[2], Jungseok Shim[1], Sungho Son[3], Jungki Lee[3],
Jinung An[2], Yongseob Lim[2], and Gyeungho Choi[2]

***Abstract*— Lane segmentation and Lane Keeping Assist System (LKAS) play a vital role in autonomous driving. While deep learning technology has significantly improved the accuracy of lane segmentation, real-world driving scenarios present various challenges. In particular, heavy rainfall not only obscures the road with sheets of rain and fog but also creates water droplets on the windshield or lens of the camera that affects the lane segmentation performance. There may even be a false positive problem in which the algorithm incorrectly recognizes a raindrop as a road lane. Collecting heavy rain data is challenging in real-world settings, and manual annotation of such data is expensive. In this research, we propose a realistic raindrop conversion process that employs a contrastive learning-based Generative Adversarial Network (GAN) model to transform raindrops randomly generated using Python libraries. In addition, we utilize the attention mask of the lane segmentation model to guide the placement of raindrops in training images from the translation target domain (real Rainy-Images). By training the ENet-SAD model using the realistically Translated-Raindrop images and lane ground truth automatically extracted from the CARLA Simulator, we observe an improvement in lane segmentation accuracy in Rainy-Images. This method enables training and testing of the perception model while adjusting the number, size, shape, and direction of raindrops, thereby contributing to future research on autonomous driving in adverse weather conditions.**

I. Introduction

In recent times, self-driving algorithms have witnessed widespread commercialization and integration into everyday life, driven by rapid advancements. However, despite their utility across various applications, these algorithms often encounter challenges when confronted with unusual scenarios, leading to performance degradation. Such degradation can have severe consequences as it results in incorrect driving outcomes [1]. Adverse weather conditions, especially heavy rain, represent a pertinent example of such edge cases [2].

This work was supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (21AMDP-C162419-01), by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (23-DPIC-19) and by Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government(MSIT) (NO.RS-2021-II211343)

[1] Jinu Pahk and Jungseok Shim are (previously with DGIST) respectively with Interdisciplinary Program in Artificial Intelligence and Computer Science and Engineering, Seoul National University, Seoul 08826, Republic of Korea jnpahk@bi.snu.ac.kr; shim2751@snu.ac.kr

[2] Seongjeong Park, Jinung An, Yongseob Lim and Gyeungho Choi are with DGIST, Daegu 42988, Republic of Korea 4or2psj@dgist.ac.kr; robot@dgist.ac.kr; yslim@dgist.ac.kr; ghchoi@dgist.ac.kr

[3] Sungho Son and Jungki Lee are with Korea Automobile Testing and Research Institute (KATRI), Gyeonggi-do 18247, Republic of Korea 1011ssh@kotsa.or.kr; highband@kotsa.or.kr

Numerous studies indicate that heavy rain adversely affects camera vision recognition algorithms [3] [4] [5] [6], with Rainy-Images underperforming compared to Original-Images. Consequently, researchers have endeavored to address this issue through deraining techniques [7] [8] [9]. However, previous research [3] highlights that incorporating deraining processes into autonomous driving algorithms, which require real-time processing, does not always enhance object detection performance and may introduce additional latency, negatively impacting system efficiency.

While existing research on heavy rain's impact on autonomous driving has largely focused on object detection, the scarcity of studies on lane detection is also problematic. Jeon et al. [10] assessed lane-keeping performance under simulated rain using the CARLA Simulator, but their method—simply adding white lines to images—falls short of replicating true heavy rain effects, like fog or lens-obstructed raindrops. Such inaccuracies can cause lane segmentation models to misidentify raindrops as lanes or overlook actual lanes, leading to reduced accuracy, as demonstrated in Section IV.C experiments.

In addition, previous research on rain impacts [3] has overlooked the variability in raindrop types. While typical models generate oval or droplet-shaped raindrops, real scenarios, especially at high speeds or in heavy rain and strong winds, often produce line-shaped raindrops with varying directions and lengths, as shown in Fig. 1. Moreover, external camera placements can lead to water droplets on the lens, causing significant view obstruction with large, blurred shapes.

To address these aforementioned challenges, this paper introduces a novel strategy to enhance cognitive model training by using DCLGAN (Dual Contrastive Learning Generative Adversarial Network) [13] to create new Rainy-Images, improving model performance in adverse weather without the need for deraining or real rainy data [14] [15]. This method eliminates the need for extra processing or inference delays by incorporating these images directly into the existing dataset. Our key contributions include:

- Rather than the end-to-end method (e.g. text-to-image) frequently used in synthetic image creation, we use randomly generated raindrop objects as an intermediate medium, allowing direct adjustment of variables such as the size, morphology, and quantity of raindrops with a high degree of versatility.
- We introduced a segmentation attentive mask-based technique to mitigate artifacts caused by inter-domain distribution inconsistencies in the image background

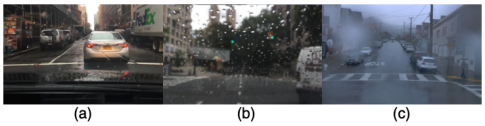

Fig. 1. Examples of real Rainy-Images depicting various raindrop states like linlie, spherical, blur, etc. Images (a) and (b) are from BDD100k [11], and image (c) is from Waymo Open Dataset [12].

during the training of DCLGAN with real Rainy-Images.

- We proposed an automatic ground truth extraction algorithm for training lane segmentation model using the CARLA Simulator. This makes large-scale data augmentation more efficient along with the Rainy-Image synthesis method above.
- Experimental results show that the lane segmentation model trained on our synthetic Rainy-Images outperforms those trained without augmentation or with other methods like ImageNet-C [16], Rain Rendering [17], and CARLA Simulator based Raindrop [10].

II. RELATED WORK

A. Image deraining

Single image deraining is a challenging task that aims to remove the visual effects of rain from the image. Methods for single image deraining can be divided into two subcategories: prior-based algorithms and data-driven methods [18]. Prior-based algorithms utilize a prior for the clean parts of images and rain obstructions to recover the background. Li et al. [19] divide the rainy image into patches, classify them as background or rain streak, and use this classification as the prior for a Gaussian mixture model. Data-driven methods, utilizing deep neural networks to map rainy images to clean ones without explicitly modeling the rain layer, have demonstrated superior performance in visual quality and quantitative metrics. For example, R. Qian et al. [20] show novel deraining performance with an attention map-guided GAN model by combining an LSTM that recurrently learns the position of a raindrop in front of a generator. Wang et al. [21], Quan et al. [22], and Xiao et al. [23] also propose deraining methods with promising performance using rain pattern-aware attention within CNNs or a Vision Transformer.

B. GAN-based unpaired image generation

The existing deraining-based methods exhibit a drawback by not considering the variability in raindrop shapes and imposing additional computational burden during autonomous driving inference. Instead, to enhance the robustness of the detection model, Rainy-Image data augmentation should be considered. Utilizing Generative Adversarial Networks (GANs), which leverage adversarial loss, enables the generation of realistic synthetic images. This approach has been widely applied for data augmentation purposes, particularly for adjusting imbalanced datasets and addressing domain shifts [24] [25].

The generation of rainy images conditioned on input images for diverse augmentations falls within the scope of Image-to-Image (I2I) translation. However, constructing a cross-domain paired dataset for direct application with Generative Adversarial Networks (GANs) presents significant challenges [26]. The CycleGAN [27] framework introduces a cycle consistency to constrain the possible mappings between domains, facilitating I2I translation even for unpaired datasets. Furthermore, DCLGAN [13] expands on CycleGAN's capabilities by incorporating patch-based contrastive learning, allowing it to handle a broad spectrum of raindrop shapes and quantities.

In the process of training DCLGAN for the generation of realistic raindrops, the methodology draws upon the techniques from image deraining methods as referenced in studies [19] and [20]. The prior of the lane segmentation model for the spatial latent representation of obstructions was used as an attentive mask indicating the locations of the raindrop.

III. PROPOSED ALGORITHM

The fabrication of Rainy-Image (RI) involves a sequential implementation of three stages. These stages encompass Generated-Raindrop (GR) generation, Raindrop Translation, and the integration of Original-Image (OI) and Translated-Raindrop (TR). Its formal representation is expressed through the following formulas (1)-(4).

$$RI = OI + \frac{\alpha}{255}TR \tag{1}$$

$$TR = DCLGAN(GR, RI + \frac{\beta}{255}G(\Psi(RI))) \tag{2}$$

$$G(x)_{i,j} = \sum_{m=-k}^{k}\sum_{n=-k}^{k} x_{i+m,j+n}\frac{1}{2\pi\sigma^2}e^{-\frac{m^2+n^2}{2\sigma^2}} \tag{3}$$

$$\Psi(RI) = \Phi(B(\mathcal{G}^2_{sum}(A_{ENet,frozen}(RI;\theta_{ENet,frozen})))) \tag{4}$$

Equation (1) entails the fusion of the Translated-Raindrop onto the Original-Image. α represents the opacity of Translated-Raindrop. Equation (2) describes the transformation of Generated-Raindrop into Translated-Raindrop, employing DCLGAN trained with datasets of Generated-Raindrop and Rainy-Image as input. In this context, the attentive mask $\Psi(RI)$ is added, which indicates the location of the raindrop of Rainy-Image. The parameter β determines the opacity of this attentive mask.

G means convolution with a Gaussian filter for smoothing the attentive mask, which can be found in formula (3). The value of the (i, j)-th pixel of the operation result is multiplied by the value of $x_{i+m,j+n}$ near the original image $x_{i,j}$ and the value of the (m, n)-th pixel of the Gaussian filter. The size of the Gaussian filter is $(2k, 2k)$, and for this reason, convolution is performed by changing m and n from $-k$ to k.

The computation of $\Psi(RI)$ is outlined in (4). Here, $A_{ENet,frozen}$ represents the activation map of the middle layer of frozen ENet-SAD weight for RI, and $\mathcal{G}^2_{sum}$ denotes

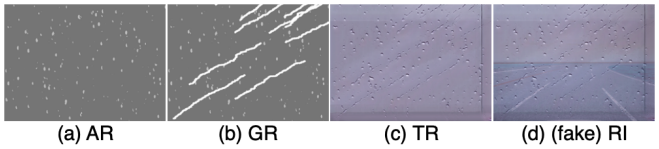

Fig. 2. The fake Rainy-Image generation process: (a) Arc-shaped raindrops (b) Adding line-shaped raindrops (c) Translated-Raindrop (d) Fake Rainy-Image.

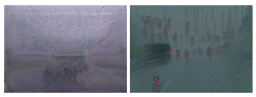

Fig. 3. Artifacts caused by incorrectly learning the background of the image train set in the Translated-Raindrop domain.

the summation after squaring the values of each channel within this activation map. Additionally, B signifies the bilinear upsampling of the reduced attention map size, and Φ represents the spatial softmax operation.

The comprehensive depiction of this process is visually represented in Fig. 2. Subfigures (a) and (b) present examples of Generated-Raindrop, (c) depicts Translated-Raindrop, and (d) exhibits a fake Rainy-Image resulting from the fusion of the Original-Image and Translated-Raindrop.

A. Raindrop Generation

The process of raindrop generation involves the creation of Generated-Raindrop utilizing the Python library PIL. The generation procedure consists of two primary steps: the creation of arc-shaped raindrops (Fig. 2 (a)) and the creation of line-shaped raindrops (Fig. 2 (b)). The algorithm for generating Generated-Raindrop is presented in Algorithm 1.

Initially, the size of the Generated-Raindrop is set to match the Original-Image. The algorithm requires the specification of the number of arc-shaped raindrops, line-shaped raindrops, and points constituting the line-shaped raindrops.

An arc-shaped raindrop is drawn after the arc start angle, central angle, and size are determined randomly. A line-shaped raindrop is drawn after the length of the line, starting point, and distance between points constituting the line are determined randomly. For a more realistic effect, the angle of the small line segments that make up the line varies from 0 to 1.8 degrees, and the distance between adjacent points varies from 0 to 3 pixels on the x-axis and 0 to 1 pixel on the y-axis. Finally, after all the arc-shaped raindrops and line-shaped raindrops are created within a single image, it results in the completion of one Generated-Raindrop image.

B. Raindrop Translation

The raindrop translation process involves the utilization of DCLGAN to convert Generated-Raindrop into Translated-Raindrop (Fig. 2 (c)). This necessitates the training of DCLGAN using datasets from both the Generated-Raindrop domain and the Translated-Raindrop domain to enable bidirectional conversion. A description of data sourcing is in the Appendix.

Algorithm 1 Generating Fake Raindrops

Output: Generated Raindrops Image

$w, h \leftarrow$ width, height of images
$n_{arc}, n_{line} \leftarrow$ number of arc, line-shaped raindrops
$n_{line\_point} \leftarrow$ number of points per line-shaped raindrop
for $i = 0$ to $n_{arc} - 1$ **do**
    $x_1, y_1 \leftarrow$ random integers(0, w), (0, h)
    $x_2 \leftarrow x_1 +$ random($w//200, w//80$)
    $y_2 \leftarrow x_2 + 2 * (x_2 - x_1)$
    Draw an arc from random angle(0, 90) to random($start$, 360) in (x_1, y_1, x_2, y_2)
end for
for $j = 0$ to $n_{line} - 1$ **do**
    $x_3, y_3 \leftarrow$ random integers(0, w), (0, h)
    $line\_angle \leftarrow$ random float(0, 360)
    for $k = 0$ to $n_{line\_point} - 1$ **do**
        $length \leftarrow$ random float(0, 3)
        $x_4 \leftarrow x_3 + length * \cos(line\_angle +$ random(0, 0.01*180))
        $y_4 \leftarrow y_3 + length * \sin(line\_angle +$ random(0, 0.01*180))
        Draw a line from (x_3, y_3) to (x_4, y_4)
        $x_3, y_3 \leftarrow x_4 +$ random(0, 3), $y_4 +$ random(0, 1)
    end for
end for

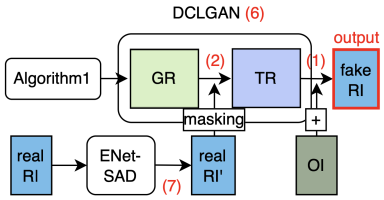

Fig. 4. The entire framework that turns GR created from Algorithm1 into fake RI through DCLGAN.

However, as depicted in Fig. 3, directly training DCLGAN with these images poses a challenge. While the Generated-Raindrop domain images solely contain raindrops, the Translated-Raindrop domain images encompass both raindrops and a background. This discrepancy can lead to unintended artifacts in the resulting generated images. To mitigate this, the prior raindrop-emphasizing attention map obtained from ENet-SAD was employed. ENet-SAD, originally trained for lane segmentation, has exhibited proficiency in distinguishing raindrops already. In addition, the objective of generating fake Rainy-Image is to introduce raindrop data that ENet-SAD might mistakenly identify as lanes into the ENet-SAD dataset. Therefore, using the attention map of ENet-SAD has the following advantages: (1) The influence of the background of Rainy-Image can be weakened and the influence of the raindrop position can be strengthened, and (2) Raindrop information that ENet-SAD trained only with Original-Image might be confused about can be reflected as an inductive bias in the training process of DCLGAN that creates fake raindrops.

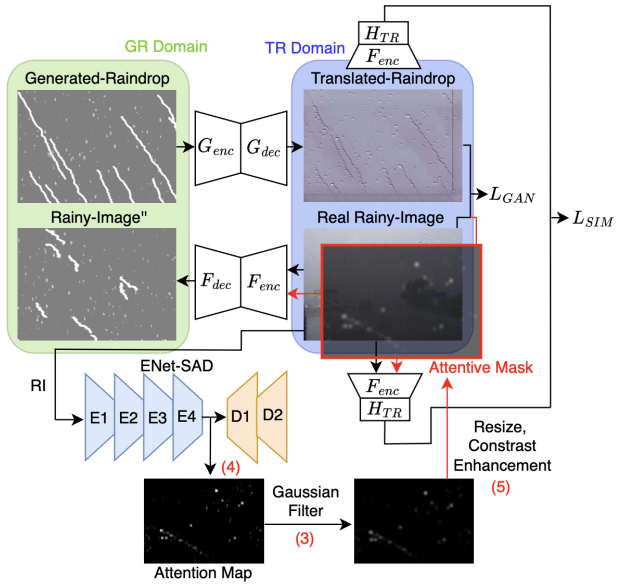

Fig. 5. The flow of GAN loss and Similarity loss in equation (6) and the process of creating an attentive mask. G_{enc}, H_{GR}, L_{GAN}, and L_{SIM} on the GR domain side are omitted in the figure.

TABLE I
NOTATIONS

Symbol	Description
RI'	Rainy-Image with the raindrop position masked
RI''	Rainy-Image translated to GR domain
G_{enc}, G_{dec}	$GR \rightarrow TR$ Generator's encoder and decoder
F_{enc}, F_{dec}	$TR \rightarrow GR$ Generator's encoder and decoder
D_{GR}, D_{TR}	Discriminator within GR or TR domain
H_{GR}, H_{TR}	MLP head for embedding features of images
H_{GRr}, H_{GRf}	MLP head to compare RI'' and GR features
H_{TRr}, H_{TRf}	MLP head to compare TR and RI features
L_{GAN}	Adversarial loss
$L_{PatchNCE}$	NCE loss between translated patch and original
L_{SIM}	Similarity loss between embedded vectors of real and fake image within the same domain
$L_{identity}$	Identity loss to maintain color when translating
E1-E4, D1-D2	Encoder and decoder layers of ENet-SAD

Fig. 4 and Fig. 5 indicate the part to which each equation is assigned during the Rainy-Image creation process. Fig. 5 shows the overall framework for training DCLGAN after obtaining an attention mask from Rainy-Image and adding it back to Rainy-Image. As indicated in (4), the attention map is obtained by squaring and summing the values of each channel in the activation output of the last encoder layer of the ENet-SAD model which has the most detailed attention information. The extracted attention map undergoes a smooth transformation by convolving it with a Gaussian filter. To enhance the contrast and facilitate clearer differentiation between the raindrop area and the background area, a min-max normalization contrast enhancement technique (5) is applied. s_k is the value of the k-th pixel after conversion to enhancement, r_k is the value of the k-th pixel of the original image, r_{max} is the highest value among all pixels in the image, and r_{min} is the lowest value.

$$s_k = 255(r_k - r_{min})/(r_{max} - r_{min}) \tag{5}$$

Subsequently, this attentive mask is concatenated along the first dimension (representing RGB) of the Real Rainy-Images in the Translated-Raindrop domain. The opacity of this attentive mask, as depicted in (2), is modulated by the β value.

DCLGAN is then trained using the augmented training set. The addition of the attentive mask to the Translated-Raindrop domain images guides the F encoder, which performs the encoding step of the TR→GR process, to focus more on the raindrop area rather than the background of the image. This can be checked in detail in DCLGAN's SimDCL Loss function. The loss function modified by reflecting the attentive mask is shown in (6). A description or value of each notation can be found in Table I and Appendix. Through this loss function, the generator can learn how to convert GR to TR, and at this time, GAN Loss and SIM Loss are applied together so that TR is as close to the real Rainy-Image as possible. RI', the target of TR's comparison, is covered with an attentive mask according to the equation (7).

$$\begin{aligned}
&L_{SimDCL}(G, F, D_{GR}, D_{TR}, H_{GR}, H_{TR}) \\
&= \lambda_{GAN}(L_{GAN}(G, D_{TR}, GR, TR, RI') \\
&+ L_{GAN}(F, D_{GR}, GR, RI', RI'')) \\
&+ \lambda_{NCE} L_{PatchNCE_{GR}}(G, H_{GR}, H_{TR}, GR) \\
&+ \lambda_{NCE} L_{PatchNCE_{TR}}(F, H_{GR}, H_{TR}, RI') \\
&+ \lambda_{SIM} L_{SIM}(G, F, H_{GR}, H_{TR}, H_{GRr}, H_{GRf}, \\
&H_{TRr}, H_{TRf}) + \lambda_{idt} L_{identity}(G, F)
\end{aligned} \tag{6}$$

$$RI' = RI + \frac{\beta}{255} G(\Psi(RI)) \tag{7}$$

C. Lane Segmentation Data and Model Training

1) Extracting Lane Ground Truth: Conventionally, training a lane detection or segmentation model necessitates the individual annotation of each pixel. However, in this study, similar to the approach outlined in Pahk et al., 2023 [29], lane information will be automatically extracted from the segmentation images provided by CARLA. Algorithm 2 presents the step-by-step procedure for extracting the ground truth and conforming to the TuSimple format.

Starting from the top, the presented algorithm systematically examines each row of the image to identify pixels representing lanes based on their RGB values. A candidate list, h_sample, is defined to evaluate the presence of lanes. The RGB value corresponding to the lane in the segmentation image is stored in $lane\_rgb$. The variables $left\_end$ and $right\_end$ indicate whether the first or last column of the image has been encountered in the lanes, signifying that the first or last lane not within the visible range anymore. GT (Ground Truth), a two-dimensional array, records the column positions of pixels belonging to lanes 1, 2, 3, and 4 in the rows corresponding to each element of the h_sample.

Algorithm 2 Automatically Extracting Lane Ground Truth from CARLA Semantic Segmentation Image

Input Semantic Segmentation Images
Output Lane Ground Truth

$h\_sample \leftarrow$ list of heights
$lane\_rgb \leftarrow$ RGB value of lane
$h, w \leftarrow$ height, width of images
$GT \leftarrow$ 2D array for lane ground truth
for $i = 0, 1, 2, ..., h-1$ **do**
    $left\_end, right\_end \leftarrow$ check if **Input**[i][0], **Input**[i][w-1] = $lane\_rgb$
    if i $\in$ h_sample **then**
        for $j = 0, 1, 2, ..., w-1$ **do**
            Append j in the $GT[i]$ that
            satisfies **Input**$[i][j] = line\_rgb$
        end for
        if $len(GT[i]) = 1$ **then**
            if $left\_end = 1$ and $right\_end = 1$ **then**
                if $GT[i][0] < w/2$ **then**
                    Set lane 2
                else
                    Set lane 3
                end if
            else if $GT[i][0] < w/2$ **then**
                Set lane 1
            else
                Set lane 4
            end if
        else if $len(GT[i]) = 2$ **then**
            Set lane 2, 3
        else if $len(GT[i]) = 3$ **then**
            if $GT[i][1] < w/2$ **then**
                Set lane 1, 2, 3
            else
                Set lane 2, 3, 4
            end if
        else
            Set lane 1, 2, 3, 4
        end if
    end if
end for
Transpose(GT)

Proceeding from the top, the algorithm iteratively checks each row incrementally. Within the ith row, check whether the pixel value of each column matches $lane\_rgb$ and enter the column value indicating the location of the lane in GT[i]. It is essential to identify which lane corresponds to each of these four or fewer values. In the TuSimple dataset, the 1st, 2nd, 3rd, and 4th lanes refer to the left left lane, left lane, right lane, and right right lane of the vehicle's travel path, respectively. If there are all four values, each can be designated as lane (1, 2, 3, 4). If there are only three values, either lane (1,2,3) or lane (2,3,4) is selected. If the second lane is to the left of the center of the image, this lane is judged to be the left lane of the vehicle, and (1,2,3) is selected. If there are two values, it is judged as (2, 3), and if there is only one value, it is judged as (2) or (3) based on the center, as above. If either the $left\_end$ or $right\_end$ is not 1, the first lane or last lane has not yet gone outside the viewing angle, so it is judged as (1) or (4) based on the center.

The algorithm improves on its predecessor [29] by analyzing $left\_end$ and $right\_end$ values to identify specific lanes, even when $GT[i]$ contains only one element at the image's bottom.

2) ENet-SAD Training and Testing: The ENet-SAD lane segmentation model is now subjected to training from scratch, utilizing the Rainy-Image and the lane ground truth data generated in the previous steps. The process of obtaining the segmentation image is as follows: (1) During the CARLA simulation, the Original-Image is captured at a rate of 30 FPS while concurrently recording the driving trajectory. The map on which the vehicle traveled is a virtual implementation of the Korea Intelligent Automotive Parts Promotion Institute (KIAPI) high-speed driving circuit, and (2) Subsequently, the recorded trajectory is replayed, and the Semantic Segmentation Camera captures images at a rate of 30 FPS, resulting in the acquisition of the segmentation images.

The accuracy of the newly trained model is computed using the TuSimple accuracy formula outlined below, in equation (8). C_{clip} denotes the count of true positive points within each clip, while S_{clip} represents the count of ground truth points present in the clip. In the TuSimple accuracy measurement, if the point predicted from Ground Truth is within 20 pixels, it is treated as correct.

$$Accuracy = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}}. \tag{8}$$

TABLE II
FID, LPIPS SCORES BETWEEN EACH FAKE REAL-IMAGE AND REAL ONE

Rainy-Image Methods	FID↓	LPIPS↓
OI	370.60	0.738
OI + AR	350.81	0.560
OI + GR	376.30	0.585
OI + Fog	383.61	0.613
OI + ImageNet-C Fog (Hendrycks et al., 2019)	340.13	0.710
OI + ImageNet-C Snow (Hendrycks et al., 2019)	305.64	0.704
OI + Rain Rendering (Tremblay et al., 2021)	316.12	0.695
OI + Rain streak (Jeon et al., 2022)	312.99	0.654
OI + TR (ours)	**248.91**	**0.529**

IV. EXPERIMENTAL RESULTS

A. Fake Rainy-Image Results

Examples of the fake Rainy-Image are shown in Fig. 6 (b) and (d). For comparison, examples of (a) Real Rainy-Image and (c) fake Rainy-Image are also included. Observing (a) and (b), it can be seen that our method can simulate scenarios where raindrops are out of focus after sticking to the camera lens, the rainstorm is so strong that raindrops appear as lines, the background is at night, there is light rainfall with a few

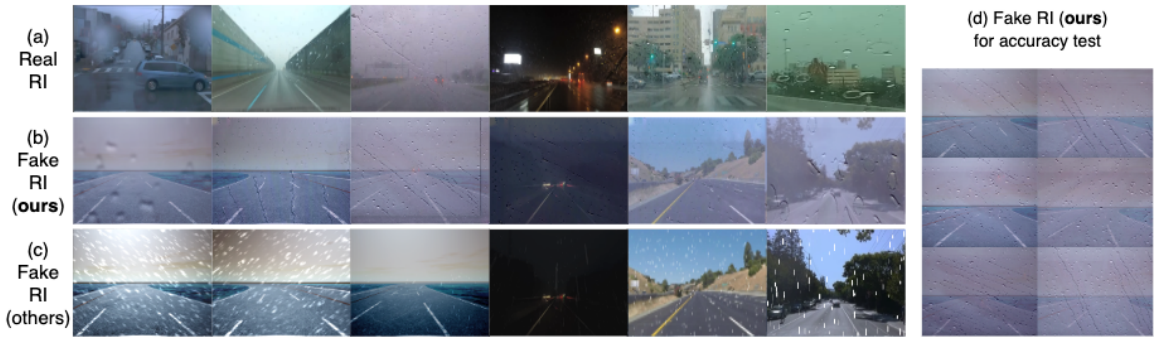

Fig. 6. Comparison between Real and Fake Rainy-Images. (a) Rainy-Images in actual road rainfall conditions (b) Rainy-Images created using our method (c) Rainy-Images produced by methods of Hedrycks et al. (2019), Tremblay et al. (2021), and Jeon et al. (2022) (d) Fake Rainy-Images generated by adjusting parameters such as n_{arc}, α for accuracy testing.

TABLE III

ORIGINAL ACCURACY AND SIMULATION RAINY ACCURACY OF WEIGHTS TRAINED USING ORIGINAL-IMAGE, REAL AND FAKE RAINY-IMAGE, AND OTHER AUGMENTATION METHODS

weight	Original Accuracy (%)↑	Simulation Rainy Accuracy (%)↑				Real Rainy Accuracy(%)↑					
		n_{arc}=200, n_{line}=10		α=150, n_{line}=10		(5)	(6)	(7)	(8)	(9)	(10)
		(1) α=150	(2) α=175	(3) n_{arc}=300	(4) n_{arc}=400	3.556 mm	(5)+(1)	(5)+(2)	(5)+(3)	7.112mm	(8)+(1)
w_{OI} ($\theta_{ENet,frozen}$)	87.66	72.39	60.36	53.07	67.60	56.18	49.59	46.66	44.57	29.76	25.14
w_{RI} **(ours)**	89.85	**85.92**	72.42	**84.87**	**82.01**	55.04	44.37	43.14	43.04	**37.70**	23.05
w_{OI+RI} **(ours)**	**90.85**	79.64	**82.88**	80.82	76.04	55.26	49.71	**49.46**	**49.94**	24.82	**28.30**
w_{IN} (Hendrycks et al., 2019)	89.18	64.36	60.17	60.32	73.18	51.84	**54.03**	44.49	43.42	21.29	23.90
w_{MB} (Hendrycks et al., 2019)	70.57	68.44	66.53	63.27	39.96	**57.63**	50.38	40.93	42.87	30.13	26.37
w_{S} (Hendrycks et al., 2019)	82.94	77.87	59.09	64.36	55.67	52.45	48.02	35.62	44.37	24.64	21.66
w_{F} (Hendrycks et al., 2019)	86.70	76.53	64.80	64.55	53.81	56.04	46.86	47.36	30.58	25.87	26.00
w_{RR} (Tremblay et al., 2021)	88.40	80.42	59.20	52.89	50.16	57.40	51.39	46.37	37.87	24.17	24.24
w_{RS} (Jeon et al., 2022)	73.28	64.39	58.15	68.06	69.53	57.39	46.13	46.70	37.67	21.36	23.11
w_{real}	87.56	83.14	71.89	70.86	60.31	**58.26**	44.71	42.54	35.33	27.48	**24.48**
$w_{real+RI}$ **(ours)**	**90.49**	**83.39**	**75.81**	**78.14**	**77.61**	53.75	**47.64**	**46.01**	**47.18**	**30.42**	23.59

raindrops, and the shape of the raindrops is irregular ellipses or curves. This is because our method allows diversification through the adjustment of various parameters such as n_{arc}, n_{line}, α, $start\_angle$, and raindrop shape. In contrast, (c) shows that using other augmentation methods like those of Hedrycks et al. [16], Tremblay et al. [17], Jeon et al. [10], these diverse situations cannot be adequately reflected. While it's possible to adjust the number and size of raindrops, they do not appear as realistically qualitative as our method, and the shape of the raindrops is fixed, unable to replicate the various forms of raindrop sensor blockage seen on camera lenses or car windows in reality. Tremblay et al. devised a realistic rendering that accounts for distance differences but has the downside of requiring a depth map for each image.

Fig. 6 (d) is an example of our fake Rainy-Images used for testing to measure the accuracy of the lane segmentation model in section IV. C. It well demonstrates the differences resulting from quantitative adjustments of parameters. The first and second columns represent when α is 150 and 175, respectively. The first, second, and third rows are when n_{arc} is 200, 300, and 400, respectively.

B. FID and LPIPS measurements

In order to assess the quantitaive quality of the fake Rainy-Image generated through the aforementioned method, the FID (Fréchet Inception Distance) and the LPIPS (Learned

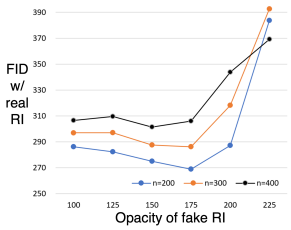

Fig. 7. Graph between α value and FID score with real images of raindrops when precipitation is 3.556 mm according to change in the n_{arc} value.

Perceptual Image Patch Similarity) score, which serve as an evaluation metric for image generation models, was employed. Table II presents a comprehensive summary of the images produced at each stage during the creation of the fake Rainy-Image and other augmentation, along with the corresponding distance scores with the real Rainy-Images. The lower the score, the closer it is to the real Rainy-Image, which means that more realistic raindrops and Rainy-Images were created. The real Rainy-Image dataset comprises of 900 images that were not utilized during the training process of DCLGAN.

In order from the table, the metric score of Original-Image, Arc-shaped Raindrop over OI, Generated Raindrop over OI, Fog rgb value (144, 142, 166) of average real Rainy-Image over OI, Hendrycks et al. (2019), Tremblay et al. (2021),

Jeon et al. (2022), and our final fake Rainy-Image are shown. Our fake Rainy-Image (OI+TR) showed the lowest values for both FID and LPIPS metrics. This indicates that the Rainy-Image generated by our method is closer to reality than other non-learning based method of drawing multiple white lines.

C. Evaluation of the Newly Trained ENet-SAD Model

In this session, we tested how much accuracy is preserved in rainy situations when a new lane segmentation model is trained using our or other methods. The results are presented in Table III. A total of 11 categories of trained weights are considered. w_{OI}, w_{RI}, and w_{OI+RI} represent the weights trained solely on Original-Image data, weights trained on fake Rainy-Images generated using our algorithm (α=200, n_{arc}=200, n_{line}=10), and weights trained on a combined dataset of the two, respectively. Subsequently, w_{IN}, w_{MB}, w_S, w_F, w_{RR}, and w_{RS} are weights trained by other augmentation methods. Specifically, they correspond to Impulse Noise, Motion Blur, Snow, and Frost by Hendrycks et al. (2019), Rain Rendering by Tremblay et al. (2021), and rain streak by Jeon et al. (2022) By comparing the accuracies of the weights, we aimed to assess how robust our weights (w_{RI} and w_{OI+RI}) are under rainy conditions compared to other augmentation methods. In addition, to assess the effectiveness of our algorithm when training on real images rather than simulator images, we also introduced w_{real} and $w_{real+RI}$. These represent the weights trained on real Rainy-Images and a combination of real Rainy-Images with synthetic Rainy-Images, respectively.

We evaluated three types of lane detection accuracy: Original Accuracy, Simulation Rainy Accuracy, and Real Rainy Accuracy. These represent the accuracy measured in a normal environment without rain augmentation, the accuracy measured using Rainy-Image created with the CARLA Simulator's images and synthesis method, and the accuracy measured in an actual rainy road environment, respectively. Example images of these three test environments can be seen in Fig. 8. We subdivided the parameter conditions and conducted a total of six experiments (1)-(10).

The reason for setting the conditions of Simulation Rainy Accuracy as (1)-(4) is as follows. According to [30], the standard for heavy rain is more than 4 mm of rain per hour. We obtained real-world Rainy-Image video with estimated precipitation rates of 3.556 mm and 7.112 mm, respectively. We aimed to find the simulation Rainy-Image parameter conditions most similar to the 3.556 mm Rainy-Images, which are close to the standard for heavy rain, and measure accuracy in a test environment that could be considered heavy rain. The most similar parameter combination to 3.556 mm was when α=[150-175] and n_{arc}=200 as shown in Fig. 7. Using this as a starting point, we increased n_{arc} by 100 to create a heavier rain situation and measured the accuracy. Accuracy for real Rainy-Image (5) 3.556 mm and (8) 7.112 mm is also in the Table III. (6)-(7) and (9)-(10) are test data created by mixing fake Rainy-Image with each image used in (5) and (8).

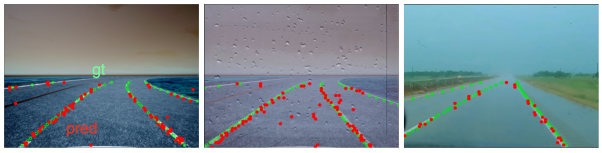

Fig. 8. Example of comparison between prediction results of lane segmentation model and ground truth in (from left to right) Original-Image, fake Rainy-Image, and real Rainy-Image situations.

Looking at Simulation Rainy Accuracy (1)-(4), it is generally observed that accuracy decreases overall as the intensity of fog or rain increases, regardless of augmentation. However, weights trained with our algorithm (w_{OI}, w_{OI+RI}, $w_{real+RI}$) showed a relatively smaller decline in accuracy, and demonstrated the highest accuracy in various cases compared to other methods. Likewise, in Real Rainy-Accuracy (5)-(10) for the real Rainy-Image and real + fake Rainy-Image mixed test data, our weights tended to perform the best overall. Our method appears to fulfill the purpose of augmentation for heavy rain well, as it generally showed higher accuracy in most heavy rain situations compared to non-augmented conditions, and also it performed better than other non-learning based methods.

V. Conclusion

In this research paper, we introduce the Rainy-Image data augmentation method to mitigate the adverse effects of various forms of raindrops (e.g., arc-shaped, line-shaped, transparent, opaque) on the performance of lane segmentation models, which have received limited attention in previous studies. Gathering real-world rainy condition data is challenging due to the unpredictable and inconsistent nature of weather, and direct annotation of lane segmentation data is expensive. However, our algorithm simplifies creating rainy datasets by using unlabeled raindrop images or videos from the Internet and training a patch-based GAN with contrastive learning. During the training process of the DCLGAN, artifacts arise from the disparities in background when using incomplete raindrop images generated using Python libraries like OpenCV or PIL and real Rainy-Images. To address this issue, we incorporate a step of generating an attentive mask by utilizing the attention map of ENet-SAD and overlaying it on a real Rainy-Image. Additionally, we present an improved automatic algorithm for extracting lane ground truth using the CARLA Simulator to facilitate the training of ENet-SAD. Consequently, the ENet-SAD model trained on the augmented Rainy-Image dataset exhibits superior performance compared to the model trained without this augmentation, particularly in scenarios with increased raindrop presence and severe fog.

The proposed algorithm's key advantage lies in its simplicity and adaptability to various images, enabling easy modification of elements like raindrop quantity and fog intensity. However, it faces several limitations: Firstly, as a learning-based method, it requires more computational resources due to its multistage Translated-Raindrops process. Secondly, in real autonomous driving settings, additional training on

specific raindrop data and adjustments in parameters like size, cloudiness, and opacity might be needed. Lastly, the reliance on simulation could lead to discrepancies between simulated and actual scenarios. Future improvements aim to enhance the model's performance and incorporate a broader range of adverse weather conditions for practical use in autonomous driving systems.

REFERENCES

[1] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel, "Benchmarking robustness in object detection: Autonomous driving when winter is coming," arXiv preprint arXiv:1907.07484, 2019.

[2] Y. Zhang, A. Carballo, H. Yang, and K. Takeda, "Perception and Sensing for Autonomous Vehicles Under Adverse Weather Conditions: A Survey," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 196, pp. 146-177, 2023.

[3] M. Hnewa and H. Radha, "Object Detection Under Rainy Conditions for Autonomous Vehicles: A Review of State-of-the-Art and Emerging Techniques," IEEE Signal Processing Magazine, vol. 38, no. 1, pp. 53-67, Jan. 2021.

[4] Y. Wang, Y. Liu, Z. Wang, and J. Liang, "The Impact of Adverse Weather Conditions on Autonomous Vehicles: How Rain, Snow, Fog, and Hail Affect the Performance of a Self-Driving Car," IEEE Vehicular Technology Magazine, vol. 14, no. 2, pp. 103-111, Jun. 2019, doi: 10.1109/MVT.2019.2892497.

[5] Abu Al-Haija, M. Gharaibeh, and A. Odeh, "Detection in Adverse Weather Conditions for Autonomous Vehicles via Multi-Scale Feature Fusion Network," AI, vol. 3, no. 2, pp. 303-317, Apr. 2022.

[6] K. Garg and S. K. Nayar, "Vision and rain," International Journal of Computer Vision, vol. 75, no. 1, pp. 3–27, 2007.

[7] X. Zhang, Y. Wang, J. Li, Y. Wang, and J. Liu, "Image Deraining Methods: A Survey," Science China Information Sciences, vol. 65, no. 1, 111101, 2022.

[8] W. Li, Q. Zhang, J. Zhang, Z. Huang, X. Tian, and D. Tao, "Toward Real-world Single Image Deraining: A New Benchmark and Beyond," arXiv preprint arXiv:2206.05514, 2022.

[9] H. Zhang, V. Sindagi and V. M. Patel, "Image De-Raining Using a Conditional Generative Adversarial Network," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 11, pp. 3943-3956, Nov. 2020,

[10] H. Jeon, Y. Kim, M. Choi, D. Park, S. Son, J. Lee, G. Choi, and Y. Lim, "CARLA Simulator-Based Evaluation Framework Development of Lane Detection Accuracy Performance Under Sensor Blockage Caused by Heavy Rain for Autonomous Vehicle," IEEE Robotics and Automation Letters, vol. 7, no. 4, pp. 9977-9984, 2022, doi: 10.1109/LRA.2022.3192632.

[11] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, T. Darrell, "BDD100K: A Diverse Driving Video Database," arXiv, arXiv:1805.04687, 2020. Available: https://arxiv.org/abs/1805.04687

[12] "Waymo Open Dataset: An autonomous driving dataset," Waymo, 2019. [Online]. Available: https://www.waymo.com/open.

[13] J. Han, M. Shoeiby, L. Petersson, and M. A. Armin, "Dual Contrastive Learning for Unsupervised Image-to-Image Translation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRw), pp. 746-755, 2021.

[14] Q. Hoang, T. Le and S. Huang, "Data Augmentation for Improving SSD Performance in Rainy Weather Conditions," 2020 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), Taoyuan, Taiwan, 2020, pp. 1-2, doi: 10.1109/ICCE-Taiwan49838.2020.9258127.

[15] Y. Shen, L. Zheng, M. Shu, W. Li, T. Goldstein, and M. Lin, "Gradient-free adversarial training against image corruption for learning-based steering," Advances in Neural Information Processing Systems, 34, pp. 26250-26263, 2021.

[16] D. Hendrycks and T. G. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," in Proc. International Conference on Learning Representations (ICLR), 2019.

[17] M. Tremblay, S. S. Halder, R. De. Charatte, and J. F. Lalonde, "Rain rendering for evaluating and improving robustness to bad weather," in International Journal of Computer Vision (IJCV), pp. 341-360, 2021.

[18] S. Li, W. Ren, F. Wang, I. B. Araujo, E. K. Tokuda, R. Hirata Junior, R. M. Cesar-Jr., Z. Wang, and X. Cao, "Single Image Deraining: A Comprehensive Benchmark Analysis," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 3838-3847.

[19] Y. Li, R. T. Tan, X. Guo, J. Lu, and M. S. Brown, "Rain streak removal using layer priors," in IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2736-2744.

[20] R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, "Attentive generative adversarial network for raindrop removal from a single image," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 2482-2491.

[21] T. Wang, X. Yang, K. Xu, S. Chen, Q. Zhang, and R. W.H. Lau, "Spatial Attentive Single-Image Deraining with a High Quality Real Rain Dataset," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 12270-12279.

[22] Y. Quan, S. Deng, Y. Chen, and H. Ji, "Deep Learning for Seeing Through Window With Raindrops," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Oct. 2019, pp. 2463-2471.

[23] J. Xiao, X. Fu, A. Liu, F. Wu and Z. -J. Zha, "Image De-Raining Transformer," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 11, pp. 12978-12995, 1 Nov. 2023, doi: 10.1109/TPAMI.2022.3183612.

[24] E. Strelcenia and S. Prakoonwit, "A Survey on GAN Techniques for Data Augmentation to Address the Imbalanced Data Issues in Credit Card Fraud Detection," in Mach. Learn. Knowl. Extr., vol. 5, no. 1, pp. 304-329, Mar. 2023. [Online]. Available: https://doi.org/10.3390/make5010019.

[25] F. Farahanipad, M. Rezaei, M. S. Nasr, F. Kamangar, and V. Athitsos, "A Survey on GAN-Based Data Augmentation for Hand Pose Estimation Problem," in Technologies, vol. 10, no. 2, Mar. 2022. [Online]. Available: https://doi.org/10.3390/technologies10020043.

[26] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," arXiv preprint arXiv:1411.1784, 2014.

[27] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," in 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 2223-2232.

[28] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning Lightweight Lane Detection CNNs by Self Attention Distillation," in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 1013-1021.

[29] J. Pahk, J. Shim, M. Baek, Y. Lim and G. Choi, "Effects of Sim2Real Image Translation via DCLGAN on Lane Keeping Assist System in CARLA Simulator," in IEEE Access, vol. 11, pp. 33915-33927, 2023, doi: 10.1109/ACCESS.2023.3262991.

[30] "Aviation — Hazards — Precipitation," World Meteorological Organization. [Online]. Available: https://community.wmo.int/en/activity-areas/aviation/hazards/precipitation. [Accessed: Dec 27, 2023].

APPENDIX

α and β represent the alpha channel values ranging from 0 to 255. The DCLGAN was trained using 5498 GR images and 4984 real RIs, extracted from videos at 0.05 FPS, available at `https://www.youtube.com/watch?app=desktop&v=UyAB5SSMYUg` 14:47-29:26, `https://youtube.com/watch?v=Dwswey-GqQc` 0:00:00-1:47:12, and `https://www.youtube.com/watch?v=FNTHBcHmrIg` 1:24:37-2:17:10. The training dataset of ENet-SAD is composed of 5977 OI and 4545 fake RI. The testset comprises 698 OI and 698 fake RI. When using the GeForce RTX 3090 Ti, the inference computation speed of DCLGAN was 11.57 FPS and ENet-SAD was 32.53 FPS with an image size of (622, 477).

$\lambda_{GAN} = 1$, $\lambda_{NCE} = 3$, $\lambda_{SIM} = 10$, $\lambda_{idt} = 1$.