electronics

MDPI

Article

GAN-Based Data Augmentation with Vehicle Color Changes to Train a Vehicle Detection CNN

Aroona Ayub [1] and HyungWon Kim [2,*]

[1] Graduate School of Computer Science, Chungbuk National University, Cheongju 28644, Republic of Korea; aroona@chungbuk.ac.kr
[2] College of Electrical and Computer Engineering, Chungbuk National University, Cheongju 28644, Republic of Korea
* Correspondence: hwkim@chungbuk.ac.kr

Abstract: Object detection is a challenging task that requires a lot of labeled data to train convolutional neural networks (CNNs) that can achieve human-level accuracy. However, such data are not easy to obtain, as they involve significant manual work and costs to annotate the objects in images. Researchers have used traditional data augmentation techniques to increase the amount of training data available to them. A recent trend in object detection is to use generative models to automatically create annotated data that can enrich a training set and improve the performance of the target model. This paper presents a method of training the proposed ColorGAN network, which is used to generate augmented data for the target domain of interest with the least compromise in quality. We demonstrate a method to train a GAN with images of vehicles in different colors. Then, we demonstrate that our ColorGAN can change the color of vehicles of any given vehicle dataset to a set of specified colors, which can serve as an augmented training dataset. Our experimental results show that the augmented dataset generated by the proposed method helps enhance the detection performance of a CNN for applications where the original training data are limited. Our experiments also show that the model can achieve a higher mAP of 76% when the model is trained with augmented images along with the original training dataset.

Keywords: convolutional neural network (CNN); generative adversarial network (GAN); data augmentation; generative models; object detection models

Citation: Ayub, A.; Kim, H. GAN-Based Data Augmentation with Vehicle Color Changes to Train a Vehicle Detection CNN. *Electronics* **2024**, *13*, 1231. https://doi.org/10.3390/electronics13071231

Academic Editors: Barbara Calabrese and Chiara Zucco

Received: 15 January 2024
Revised: 29 February 2024
Accepted: 4 March 2024
Published: 26 March 2024

1. Introduction

The performance of a convolutional neural network is heavily affected by the presence of diverse and large quantities of training images. CNNs require a significant number of training images to perform well, but when these images are few in number, CNNs may over-fit and lose accuracy. For example, a large-scale image dataset for self-driving vehicles called BDD 100k [1] contains 100,000 images with different objects, such as vehicles, pedestrians, and traffic signals. However, collecting and annotating such a large amount of images is not always feasible, as it is very time-consuming and costly. Data augmentation techniques can help to solve this problem by creating more diverse training images from existing images without acquiring new samples.

Conventional data augmentation methods apply to various transformations that can be implemented on an original dataset, such as the linear translation, rotation, flip, crop, and cut–mix transformations. However, such techniques produce limited augmented data with low diversity and variance. In this paper, we propose a data augmentation method that can explore a larger variance space through training a generative adversarial network (GAN) in a different domain called the source domain. We then use the trained GAN to generate augmented images in the target domain. We demonstrate that, when trained with this generated data along with the original data, a convolutional neural network (CNN) can exhibit improved detection performance.

Image-to-image translation is a process that modifies a specific aspect of an image to a desired domain using GAN networks. There are many models that can perform this task for different applications. For instance, FACEGAN [2] can alter facial attributes, Weather GAN [3] can change weather conditions, and CycleGAN [4] can transform images from one domain to another, such as a picture to a painting, summer to winter, and a zebra to a horse. These tasks have shown remarkable progress in recent years. However, most GANs have mainly focused on faces and medical images, and there has not been much research on vehicle color changes or vehicle augmentation.

To produce an effective translation of vehicle colors, we propose a method employing a GAN model called ColorGAN, which re-targets StarGAN-v2 [5] for vehicle color change to generate diverse images across color domains for vehicles. Our proposed model can be used to extend any vehicle dataset by generating colored vehicle images.

2. Related Work

2.1. Generative Adversarial Networks

GANs have been improved by many techniques since their introduction [6]. New techniques have enhanced the quality and variety of the images generated by GANs. As a result, images generated by GANs have enhanced quality and variety. For example, VAGAN [7] can create images with poor weather conditions in their background, and CycleGAN [4] can produce artificial night-time images. Object detectors can benefit from using augmented images [8,9] for training. However, most of the previous work focused on modifying the background or the environment of the image; thus, there is no existing work on generating vehicle images with different color attributes. In this paper, we are concerned with transforming only the color attributes of vehicles in the input images while keeping the background. Figure 1 shows the general structure of the StyleGAN [10] model, while Figures 2 and 3 show the architectures of CycleGAN [4] and StarGAN [5], respectively.

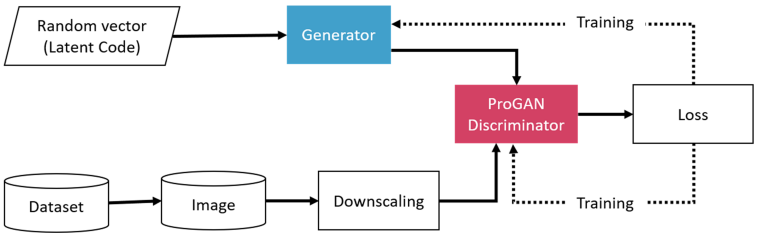

Figure 1. The StyleGAN [10] generator is an improved version of ProGAN's image generator. It starts by training with a very-low-resolution image (e.g., 4×4 pixels) and adds a higher resolution layer with every iteration. A mapping network, style modules (AdaIN), stochastic variations, style mixing, truncation, and fine-tuning were added to the traditional ProGAN image generator to create StyleGAN.

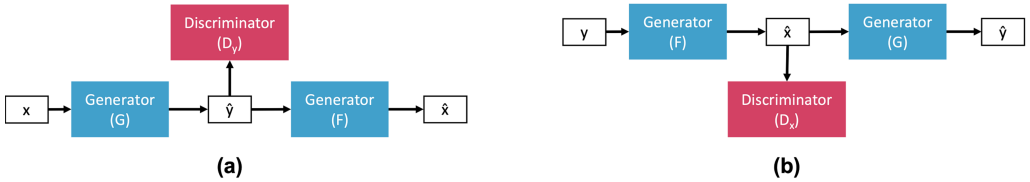

Figure 2. The CycleGAN [4] model contains two mapping functions, $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and the associated adversarial discriminators DY and DX. DY encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for DX and F. To further regularize the mappings, two cycle consistency losses were introduced to capture the intuition that if we translate from one domain to the other and back again, we should arrive at where we started. These two cycle consistency losses are as follows: (**a**) forward cycle consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$; (**b**) backward cycle consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx$ y.

2.1.1. StyleGAN

This GAN model is used to apply the style of the target image to the source image. The generated image looks like a combination of the source and target image. The goal is to blend the source and target image. StyleGAN is designed to change the styles of face images, and so is trained with a dataset named FFHQ [11] consisting of highly variant generated images of faces.

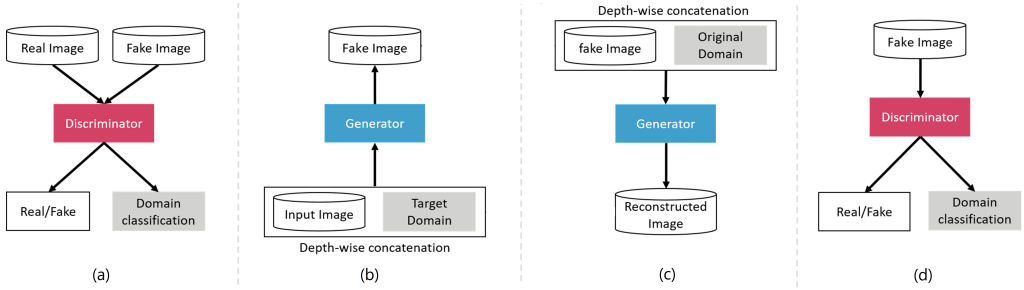

Figure 3. Overview of StarGAN [12], consisting of two modules, a discriminator and a generator. (**a**) Discriminator learns to distinguish between real and fake images and classify the real images to its corresponding domain. (**b**) Generator takes in as input both the image and target domain label and generates a fake image. The target domain label is spatially replicated and concatenated with the input image. (**c**) Generator tries to reconstruct the original image from the fake image given the original domain label. (**d**) Generator tries to generate images indistinguishable from real images and classifiable as target domain by Discriminator.

StyleGAN is not a suitable model for our intended purpose, which is to modify only the color of the vehicle while preserving the other attributes such as model, shape and angle. When applied to a vehicle dataset for color change, StyleGAN also modifies the other attributes of the vehicle in the generated images. It interpolates between a source image and a target image of the vehicle. However, it lacks control over the selection of attributes to keep, the attributes to retain and those to alter.

2.1.2. CycleGAN

CycleGAN is a model designed for image-to-image translation between two domains using unpaired datasets. It operates as a one-to-one mapping network, requiring $k(k-1)$ generators to learn the mappings among k domains. Additionally, the network cannot simultaneously train generators across different domains.

To utilize this model for our objective, multiple CycleGAN models must be trained for each color domain. For instance, transforming a vehicle from black color to red, blue, and green color requires training six generator models, each with one color domain. However, training several generators is a resource-intensive and time-consuming task, rendering this approach unsuitable for our research. Consequently, while CycleGAN can excel in tasks like style transfer, object transformation, season change and photo enhancement, it is not well suited for our objective.

2.1.3. StarGAN

StarGAN is a model that performs image-to-image translation among multiple domains, which is an enhancement over CycleGAN. It has a generator that takes a set of training images and domain labels as input. It then learns to translate images from one domain into multiple target domains simultaneously.

However, StarGAN still learns a fixed mapping for each domain. Since a label is represented by a hard-coded vector, the generator produces the same output for each domain, given a source image.

2.1.4. StarGAN-v2

StarGAN-v2 is an improved model over StarGAN. It uses domain-specific style codes instead of domain labels to represent different styles within a specific domain. Therefore, StarGAN-v2 can create diverse images among multiple domains, unlike StarGAN.

When used as a base model, StarGAN-v2 produced higher-quality vehicle color images compared to StyleGAN, CycleGAN and StarGAN. Therefore, we chose StarGAN-v2 as the base model to implement the proposed GAN model called ColorGAN by extending StarGAN-v2 to produce vehicle images across multiple color translation domains.

2.2. Image-to-Image Translation

Recent works have achieved impressive results in image-to-image translation. For instance, StyleGAN [10] learns the features of faces and transforms the image with the learned features. Our approach is to make a GAN model learn multiple color attributes of vehicles and generate vehicle images with a set of desired colors.

2.3. Data Augmentation

To reduce overfitting, many CNN models adopt label-preserving transformations such as flipping, rotation, cropping, data mixing and local and affine distortion. Recently, Auto ML has been utilized to explore data augmentation for a given dataset and task. However, using GANs for data augmentation remains yet to be proven and thus requires further research. In this paper, we introduce a novel approach to utilizing GANs for data augmentation.

2.4. Target Object Detection CNN Model

You Only Look Once (YOLO) v5 is an object detection model that predicts bounding boxes and classes of each detected object in an image. While it stands as a state-of-the-art detector due to its high performance and speed, enhancing its detection accuracy (mAP) solely only using available datasets with hand-edited labels proves challenging. Our approach efficiently generates a large number of augmented images from the provided dataset by altering the vehicle colors. We demonstrate that our augmented dataset can effectively improve the detection accuracy of the targeted CNN model.

3. Proposed GAN Model

We utilize a vehicle dataset and employ our novel method, ColorGAN (as depicted in Figure 4), to learn the colors of the vehicles. Subsequently, the trained ColorGAN augments a given dataset to generate an expanded dataset, upon which target CNN models are trained and evaluated.

ColorGAN is trained using a dataset containing various vehicle attributes, as our target domain of interest revolves around vehicle color attributes. We have successfully translated the source image to the target image, maintaining the same shape, model, and angle attributes, while only modifying the color attribute of the vehicle.

3.1. Pre-Processing

3.1.1. Dataset Selection

Among various public datasets that have been analyzed, two Vehicle Re-identification datasets called VeRi-Wild [13] and VeRi-776 [14] compromise 416,314 and 49,357 vehicle images, respectively. These images are of a surveillance nature, captured from different cameras, viewpoints and illumination conditions. Due to their surveillance nature, the images are not very clear and also each image contains many overlapped vehicles with severe occlusions, making them unsuitable for training ColorGAN. Another dataset, the Stanford Cars Dataset, contains 16,185 images of cars from the web. However, its size is insufficient to train our GAN model.

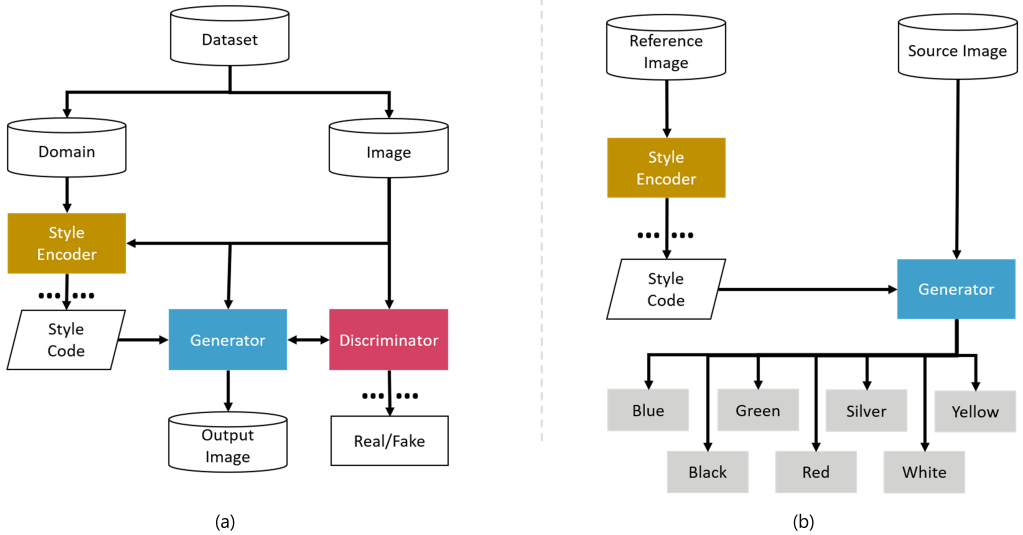

Figure 4. Overview of ColorGAN architecture. The dots at the style encoder and discriminator show multiple output branches, one of which is selected when training the corresponding domain. (**a**) The training process where the generator translates an input image into an output image reflecting the domain-specific style code. The style encoder extracts the style code of an image, allowing the generator to perform reference-guided image synthesis. The discriminator distinguishes between real and fake images from multiple domains. (**b**) During inference, the generator takes an input image and extracts the low-level features from the image. It then modulates the style information of the reference image coming from the style encoder and generates an image depicting the styles of the reference image.

The vehicle dataset selected for training the ColorGAN model is called the Comprehensive Cars Dataset (CompCars) [15], which contains both surveillance and web–nature images. We specifically selected web–nature images, as shown in Figure 5, due to their high resolution and adequate quantity to effectively train a GAN model. The CompCars dataset consists of 136,727 images, each containing one vehicle per image. The distribution of the data is outlined in Table 1.

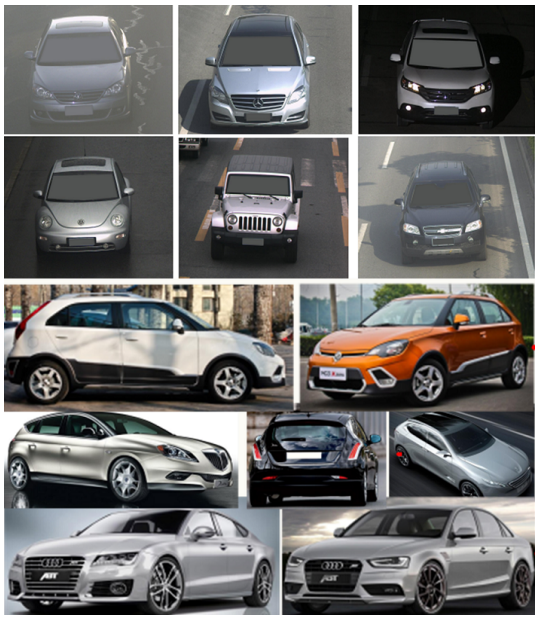

Figure 5. Image samples from CompCars dataset. The first two rows belong to surveillance–nature images, while the remaining rows are from web–nature images.

Table 1. CompCars Dataset distribution.

Total no. of Images	Training Images	Testing Images	Validation Images
136,726	100,368	25,638	10,720

3.1.2. Color Detection Algorithm

The dataset we used, CompCars, contains information about the viewpoint, car model, and bounding box of the vehicles. We utilized the bounding box information to locate the vehicle and extract its color information from the image. The dataset was divided into several groups based on the color of the vehicles. The number of groups is defined as N_{Aug}, which later determines the number of augmented images synthesized for each given source image. In this work, we chose $N_{Aug} = 7$, a value used consistently throughout the paper.

An algorithm called pixel binning is proposed in this work. It extracts vehicle color information from the selected dataset, as illustrated in Figure 6. Binning involves grouping data based on the range of the pixel values. All the individual pixel values of R, G, and B channels in an image are categorized in some bins. For the experiment presented in this work, we selected four bins of equal size—0–63, 64–127, 127–191 and 192–255—which can provide 64 different colors. However, similar color families were combined into major colors. For example, pale and bright red are categorized into the major color, red. Based on the most common vehicle colors in the CompCars dataset, we selected the following seven major colors: yellow, white, silver, red, green, black, and blue as shown in Table 2. Figure 6 illustrates the proposed algorithm, which determines where each color pixel value falls among the bins and selects the color label of each vehicle in the image among the seven colors listed in Table 2. Step (c) of Figure 6 crops only the vehicle's body area, excluding the window regions and tire areas, to select the representative body color of the vehicle. Some of the extracted colors on CompCars are displayed in Figure 7. The distribution of colors in training and validation subsets of the CompCars dataset is provided in Table 2.

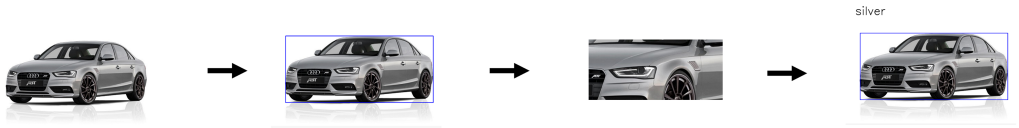

Figure 6. Algorithm to extract the color of the vehicle: (**a**) Read labels of the original image, (**b**) draw bounding box, (**c**) crop ¼ of the bounding box from center, (**d**) detect the color using the bins method.

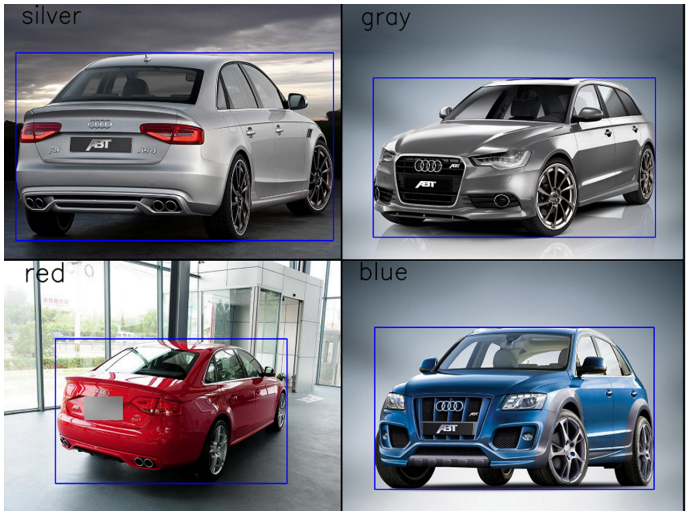

Figure 7. Detected colors on the CompCars dataset.

Table 2. CompCars training and validation images categorized by colors by color detection algorithm, when $N_{Aug} = 7$.

Domain	Black	Blue	Green	Red	Silver	White	Yellow
Training Subset	31,532	8093	6345	17,602	15,055	19,726	2015
Validation Subset	3400	800	700	1900	1600	2100	220

3.2. Image Augmentation Steps in ColorGAN

The HACKATHON dataset is chosen in this work to assess ColorGAN's ability to augment the dataset for training a target CNN. The HACKATHON dataset compromises scenes of roads with vehicles, aimed at training a vehicle detection CNN. The total number of images in the dataset is 53,000, with 40,000 belonging to the training subset and 13,000 to the validation subset. It includes bounding box labels for various classes such as car, bus, truck, traffic signal, etc. Among these classes, only the car class is utilized to evaluate the proposed augmentation technique based on ColorGAN. The following steps elaborate on the details of ColorGAN's process for vehicle color augmentation.

3.2.1. Vehicle Cropping Step

An image from the training dataset (HACKATHON in the current work) compromises vehicles, traffic signals, pedestrians, etc. To change the color of the vehicles in each image, this step involves cropping the vehicles out of the images. It utilizes the bounding box information to determine the location of the vehicles' so-called cropped patches. For these cropped patches, only vehicles v_i meeting the following conditions are only selected:

$$\text{Condition 1} : BB_{width}(v_i) > T_{width}, \tag{1}$$

$$\text{Condition 2} : IoU(v_i, v_j) < T_{IoU} for all neighbors v_j, \tag{2}$$

$$\text{Condition 3} : BB_{left} > T_R and BB_{right} < T_R, \tag{3}$$

Here, $BB_{width}(v_i)$ indicates the width of the bounding box of vehicle v_i, while T_{width} is a specified threshold for the width. In the current experiment, we chose a T_{width} of 80 pixels. $IoU(v_i, v_j)$ indicates the intersection over union calculated between the target vehicle v_i with respect to its neighboring vehicles v_j. T_{IoU} is the specified threshold for the IoU, for which we chose 10% in the experiment presented in this paper. We found that a small T_{IoU} is more desirable. If we increase T_{IoU}, the algorithm tends to select more vehicle objects that are partially occluded by other vehicles, which can degrade the accuracy. In Condition 3, BB_{left} and BB_{right} indicate the left and right edges of the bounding box, where T_R and T_L are specified thresholds of the right and left edges of the image, respectively. Condition 3 ensures the selection of vehicle objects showing a complete view with no clipped views. These conditions are shown in Figure 8. Our extensive experiments show that ColorGAN produces higher-quality color transitions when it is provided with vehicle images without occlusions.

For the cropped vehicle patches selected by the above conditions, the vehicle cropping step determines the color of the vehicle using the color detection algorithm shown in Figure 6. These cropped vehicle patches are then grouped based on the color determined by the color detection algorithm.

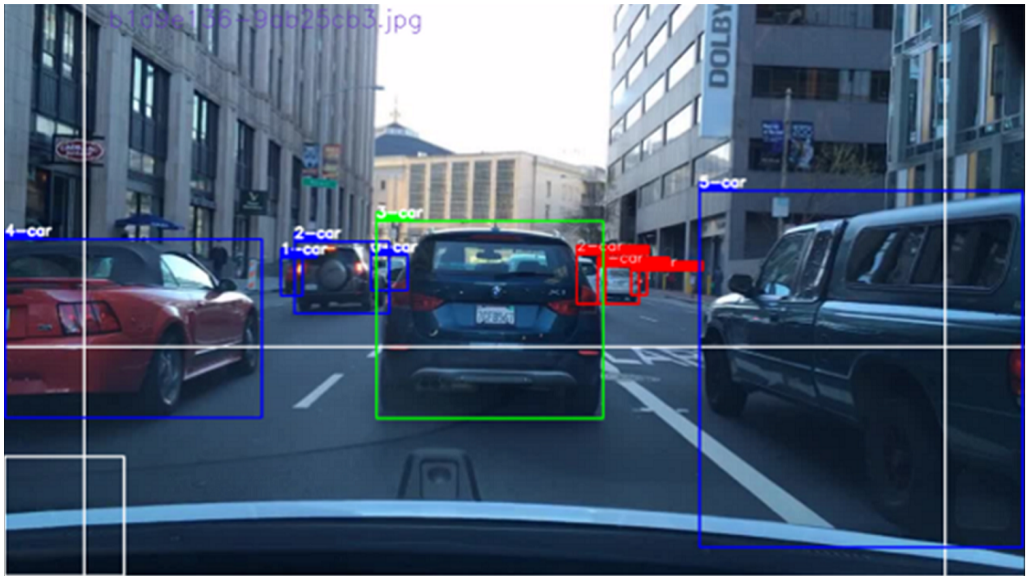

Figure 8. Bounding box colors based on detection condition: Red—occluded; Blue—too small; Green—BB satisfying all conditions; Blue—BB touching sides.

3.2.2. Vehicle Color Synthesis Step

The proposed color augmentation converts the original vehicle patches by generating diverse vehicle colors with specified color domains. This step initially groups the vehicle patches by their domain. Then, it applies ColorGAN to change the color of each vehicle patch to N_{Aug} different color domains.

The generator network of our ColorGAN model consists of three stages: encoder, bottleneck and decoder, as illustrated in Figure 9. Each stage is designed with six layers with a minimum resolution of 32 $\times$ 32 to a maximum resolution of 512 $\times$ 512. The resolution of the hidden layers is capped at 512 pixels to reduce the network size. Figure 9a depicts the structure of each stage of the generator network. The style code vector dimension is set to 128, while the latent vector dimension is set to 32. The style code encoder network of our ColorGAN model consists of 10 layers. The input dimension equals the latent vector of 3 $\times$ 32 while the network produces an output of 512xN_{Aug} dimensions. Figure 9b shows the structure of the style encoder network. We set the total number of domains to $N_{Aug} = 7$ in this work, although it can be extended to a larger number N_{Aug}. We resized the input image resolution to 512 $\times$ 512 to match the generated output image. The resulting vehicle patches from this color augmentation step are referred to as augmented vehicle patches. Figure 10 shows examples of results produced by the image synthesis step applied on both CompCars and BDD100k datasets. Since ColorGAN is trained using the CompCars dataset, the images chosen for synthesis from CompCars are the ones that are neither a part of the training nor validation subset of the datasets. For the BDD dataset, vehicle bounding boxes are first cropped from road view images, and then the cropped images are used for evaluating the performance of the image synthesis.

The generated images preserve the pose and shape of the source vehicle images while replacing their original color with the color information from the reference images. This process augments a vehicle image to N_{Aug} different color domains, which are later utilized to augment the dataset.

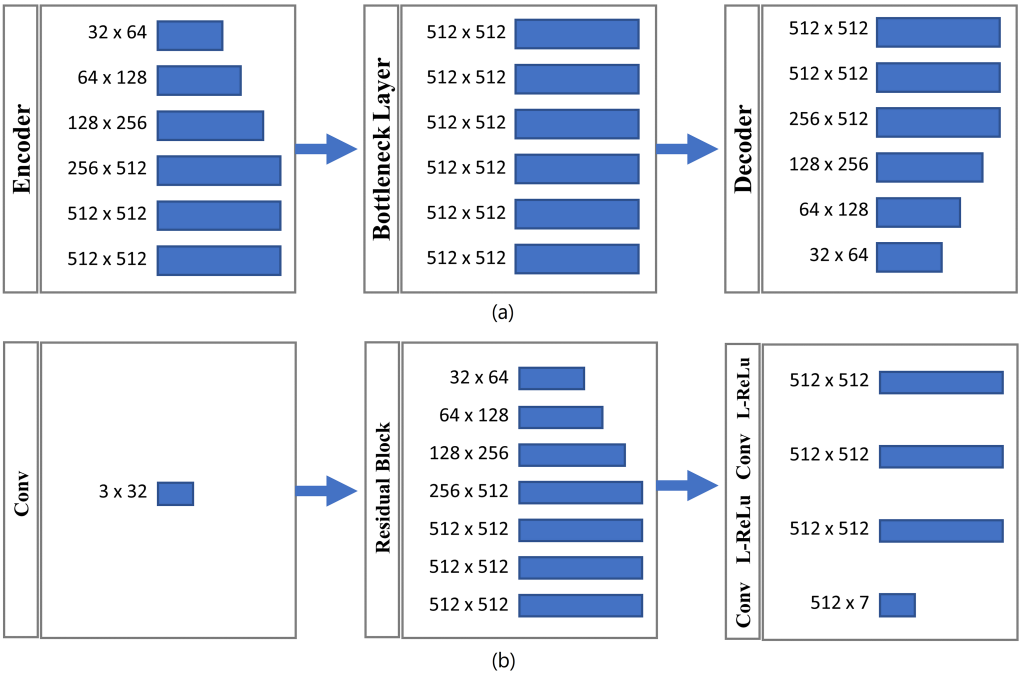

Figure 9. The internal structure of each layer of ColorGAN generator (**a**) and ColorGAN style encoder (**b**).

Figure 10. Images synthesized by ColorGAN trained using CompCars: (**a**) results synthesized by applying ColorGAN to BDD100k dataset; (**b**) results synthesized by applying ColorGAN to CompCars dataset.

3.2.3. Vehicle Background Recovery Step

Augmentation with ColorGAN alters the background of the cropped vehicle patches. When the resulting vehicle patches are reinserted into the image, the distorted background of the vehicles creates a discontinuity in the image, as depicted in Figure 11a. To address this issue, we implement a specialized step called "background recovery" based on instance segmentation of vehicle instances. This step involves extracting the vehicle's background patch from the original image using an instance segmentation CNN model. Subsequently, the recovered background patch is merged into the augmented vehicle patch, as illustrated in Figure 11b.

(a) (b)

Figure 11. (**a**) Merging the augmented image without vehicle segmentation step; (**b**) merging the augmented image with vehicle segmentation step.

For instance, segmentation, it utilizes YOLACT [16], which is pretrained on the MS COCO dataset [17]. Figure 12 illustrates the procedure of the vehicle background recovery step. The segmented vehicle from a cropped patch is used as a mask to carve out the background-only patch from the original image. Additionally, we extract a vehicle-only patch with the background masked out from the synthesized vehicle image. Subsequently, we merge the recovered background patch with the synthesized vehicle-only patch, as depicted in Figure 12d–h.

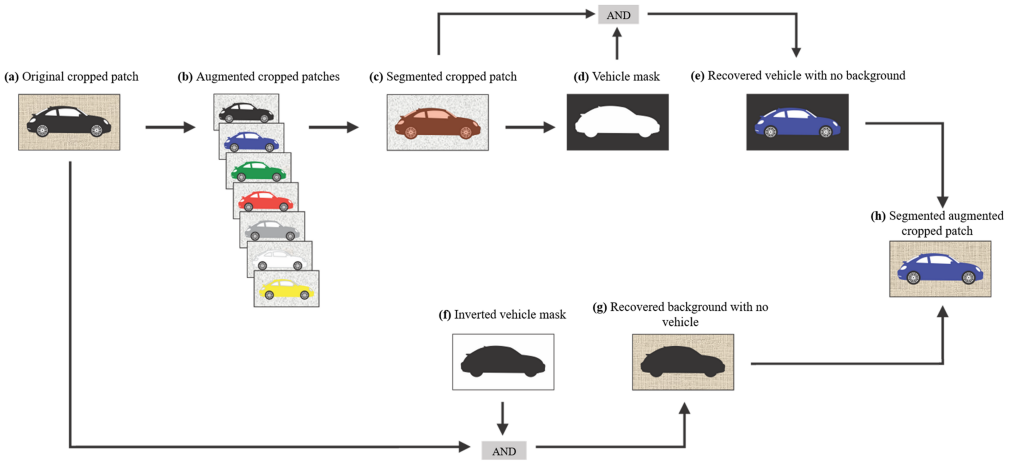

Figure 12. The process of background recovery for an augmented vehicle patch: (**a**) Original cropped patch, (**b**) augmented cropped patches, (**c**) segmented cropped patch, (**d**) vehicle mask, (**e**) segmented vehicle with its background masked out, (**f**) inverted vehicle mask, (**g**) segmented background with the vehicle masked out (**h**) augmented cropped patch with the original background recovered.

3.2.4. Vehicle Insertion Step

After the vehicle patches augmented with N_{Aug} colors receive their recovered background by the above steps, the vehicle insertion step inserts each of the N_{Aug} synthesized vehicle patches into the original image to produce N_{Aug} augmented images. As an example of the augmented images produced by this step, Table 3 shows the number of original images and the number of images with one cropped vehicle and up to four cropped vehicles in the training and validation subsets of the HACKATHON dataset. For the experiment presented in this work, we set N_{Aug} to 7.

3.3. Augmentation Results

Figure 13 shows example results of augmented images from the HACKATHON dataset, once the augmented cropped patches of N_{Aug} color domains were merged with original images using the aforementioned augmentation steps. Table 4 displays the number of images obtained after augmenting the HACKATHON dataset.

Table 3. Number of cropped patches obtained from HACKATHON dataset.

HACKATHON Dataset	Total Images	Cropped Cars	Images with 1 Cropped Car	Images with 2 Cropped Cars	Images with 3 Cropped Cars	Images with 4 Cropped Cars
Training Subset	40,000	19,176	15,486	3300	366	24
Validation Subset	6541	3705	1148	127	4	2

Table 4. Number of images obtained after augmenting a HACKATHON dataset.

	Total Images	Cropped Images	Augmented Images	Total Images in New Dataset (Original + Augmented)
Train Subset	40,000	19,176	134,232	174,232
Validation Subset	13,896	6541	45,787	59,683

Figure 13. The result of merging an augmented cropped patch to the original HACKATHON image. The top left image is the original image, the other seven are the augmented image results.

4. Evaluation

4.1. Training CNN Models

To evaluate the effectiveness of the proposed ColorGAN as a data augmentation technique in enhancing the accuracy of a CNN model, we selected two well-known object detection CNN models: YOLOv5m and YOLOv5s. The HACKATHON dataset served as the base training dataset. Using the proposed vehicle color synthesis with $N_{Aug} = 7$,

the base training dataset was augmented to eight times its original size, compromising original and colored augmented images. All seven synthesized colors were utilized for training, while the validation dataset consisted of original images only. We re-trained the two CNN models and compared their mean Average Precision (mAP) performance in different scenarios using the original HACKATHON dataset and the augmented dataset for 500 epochs. The training process took three days on an RTX 3090 GPU.

4.2. Evaluation Metrics

To the best of our knowledge, the performance improvement of object detection CNN models based on vehicle color augmentation represents the world's first attempt, which has not been studied in previous works. Therefore, a direct comparison with previous papers is not possible. Thus, we can only demonstrate the map improvement of target CNN models achieved by the proposed ColorGAN technique compared to the initial mAP obtained by training with the original dataset.

We evaluate the mAP performance changes of both target CNN models, YOLOv5s and YOLOv5m, trained using the original HACKATHON dataset and the augmented dataset generated by ColorGAN, respectively. The performance is measured by using accumulated mean Average Precision (mAP0.5:0.95), which is an mAP averaged over IOU thresholds ranging from 0.5 to 0.95. Table 5 compares the mAP0.5:0.95 of the first CNN model, YOLOv5m, in two training scenarios: the first scenario trains the model using only the original HACKATHON dataset, while the second scenario trains the model using both the original dataset and augmented dataset using the vehicle color synthesis. The first scenario yields a mAP of 78.4%, while the second scenario shows an improvement of up to 0.7% when blue augmented colors are added. When all augmented colors are included in the training dataset, it shows an improvement of 0.8%.

Table 5. Precision results of YOLO-v5(M) model trained over HACKATHON dataset and augmented dataset, individually with each color domain.

	Number of Training Images	**Number of Validation Images**	**mAP [0.5:0.95]**
Original (Baseline)	40,000	13,896	78.4
Original + Black	59,176	20,437	78.8
Original + White	59,176	20,437	78.8
Original + Blue	59,176	20,437	79.1
Original + Red	9176	20,437	78.8
Original + Silver	59,176	20,437	78.8
Original + Yellow	59,176	20,437	78.7
Original + Green	59,176	20,437	78.7
Original + All Augmented Colors	**174,232**	**59,683**	**79.2**

Table 6 compares mAP0.5:0.95 of the second CNN model, YOLOv5s, with the same two scenarios mentioned above. The model showed an mAP of 75.2% and showed an improvement of up to 0.7% with the addition of various augmented colors. When all augmented colors are added to the training dataset, it shows an improvement of 0.9%.

These results demonstrate that the proposed ColorGAN and vehicle color augmentation technique can significantly enhance the detection accuracy of CNN models whose accuracy cannot be improved by only the existing full dataset. Furthermore, the entire augmentation process, as well as the retraining process, is fully automated. Therefore, we expect that the proposed method can be readily utilized as an additional data augmentation procedure for vehicle detection CNN models intended for autonomous driving applications. Additionally, this approach can be extended to color augmentation for CNN models.

Table 6. Precision results of YOLO-v5(S) model trained over HACKATHON dataset and augmented dataset, individually with each color domain.

	Number of Training Images	Number of Validation Images	mAP [0.5:0.95]
Original (Baseline)	40,000	13,896	75.2
Original + Black	59,176	20,437	75.7
Original + White	59,176	20,437	75.7
Original + Blue	59,176	20,437	75.7
Original + Red	9176	20,437	75.7
Original + Silver	59,176	20,437	75.7
Original + Yellow	59,176	20,437	75.7
Original + Green	59,176	20,437	75.8
Original + All Augmented Colors	**174,232**	**59,683**	**76.1**

5. Conclusions

Data augmentation is a widely applicable approach to improving the accuracy of convolutional neural networks (CNNs). In this paper, we proposed a model called ColorGAN to generate an augmented dataset with a variety of color domains. We also introduced an effective technique to automatically create labels for vehicle color domains and train the ColorGAN model using those domains. We conducted experiments to train the ColorGAN model using the CompCars dataset and then to produce an augmented dataset by changing the vehicle colors of a larger-scale dataset, HACKATHON, to seven different colors. The proposed augmentation method can produce a larger dataset with high variance. While the number of images in the original HACKATHON dataset is 40,000, the augmented dataset with seven synthesized vehicle colors increases to 174,232. We also demonstrated that the augmented dataset improves the accumulated mAP performance of two CNN models, YOLOv5m and YOLOv5s, by up to 0.8% and 0.9% compared with the original HACKATHON dataset. Since the entire procedures of the proposed ColorGAN training and data augmentation are fully automated, our color augmentation technique can be easily adapted to enhance the training accuracy of many CNN models.

Although these results are promising, it is important to acknowledge that there is room for improvement. Future research could explore ways to improve the ColorGAN model, investigate additional color augmentations, or integrate more diverse datasets for broader applications. Additionally, improvements in automated labeling processes and exploration of alternative architectures for ColorGAN may contribute to more robust results.

Author Contributions: Conceptualization, A.A. and H.K.; methodology, A.A.; software, A.A.; validation, A.A. and H.K.; resources, H.K.; writing—original draft preparation, A.A.; writing—review and editing, H.K.; supervision, H.K.; funding acquisition, H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant for RLRC funded by the Korea government (MSIT) (No. 2022R1A5A 8026986, RLRC) and was also supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-01304, Development of Self-learnable Mobile Recursive Neural Network Processor Technology). It was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2023-2020-0-01462, Grand-ICT) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation). This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2020-0-01077, Development of Intelligent SoC having Multimodal IOT Interface for Data Sensing, Edge computing analysis and Data sharing. This work was supported by

the Starting growth Technological R&D Program (S3318502) funded by the Ministry of SMEs and Startups (MSS, Korea).

Data Availability Statement: The CompCars dataset can be found at mmlab.ie.cuhk.edu.hk/datasets/comp_cars (accessed on 3 March 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yu, F.; Xian, W.; Chen, Y.; Liu, F.; Liao, M.; Madhavan, V.; Darrell, T. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv* **2018**, arXiv:1805.04687.
2. Tripathy, S.; Kannala, J.; Rahtu, E. Facegan: Facial attribute controllable reenactment gan. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Online, 5–9 January 2021; pp. 1329–1338.
3. Li, X.; Kou, K.; Zhao, B. Weather GAN: Multi-domain weather translation using generative adversarial networks. *arXiv* **2021**, arXiv:2103.05422.
4. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2223–2232.
5. Choi, Y.; Uh, Y.; Yoo, J.; Ha, J.W. Stargan v2: Diverse image synthesis for multiple domains. *arXiv* **2020**, arXiv:1912.01865.
6. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *arXiv* **2014**, arXiv:1406.2661.
7. Wang, J.G.; Wan, K.W.; Yau, W.Y.; Pang, C.H.; Lai, F.L. Vagan: Vehicle-aware generative adversarial networks for vehicle detection in rain. In Proceedings of the 2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV), Shenzhen, China, 13–15 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 363–368.
8. Ayub, A.; Kim, H. GAN based Data Augmentation with vehicle color change for training vehicle detection CNN. In Proceedings of the 8th International Conference on Next Generation Computing, Tianjin, China, 18–21 March 2022; pp. 261–262.
9. Ayub, A.; Kim, H. Training and Synthesis Based Vehicle Color Conversion Technique Using GAN. *J. Multimed. Soc.* **2023**, *26*, 713–721. [CrossRef]
10. Horev, R. Style-Based GANs—Generating and Tuning Realistic Artificial Faces. Lyrn.AI. Archived from the Original on November 5, 2020. Retrieved February 16, 2019. Available online: https://towardsdatascience.com/explained-a-style-based-generator-architecture-for-gans-generating-and-tuning-realistic-6cb2be0f431 (accessed on 3 March 2024).
11. Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 4401–4410.
12. Choi, Y.; Choi, M.; Kim, M.; Ha, J.W.; Kim, S.; Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 8789–8797.
13. Lou, Y.; Bai, Y.; Liu, J.; Wang, S.; Duan, L. Veri-wild: A large dataset and a new method for vehicle re-identification in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 3235–3243.
14. Liu, X.; Liu, W.; Mei, T.; Ma, H. A deep learning-based approach to progressive vehicle re-identification for urban surveillance. In *Computer Vision—ECCV 2016, Proceedings of the 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016*; Springer: Cham, Switzerland, 2016; pp. 869–884.
15. Yang, L.; Luo, P.; Change Loy, C.; Tang, X. A large-scale car dataset for fine-grained categorization and verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3973–3981.
16. Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. Yolact: Real-time instance segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9157–9166.
17. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *Computer Vision—ECCV 2014, Proceedings 13th European Conference, Zurich, Switzerland, 6–12 September 2014*; Proceedings, Part V 13; Springer: Cham, Switzerland, 2014; pp. 740–755.