Improving the Performance of Automotive Vision-based Applications Under Rainy Conditions

Yazan Hamzeh, Alireza Mohammadi, Samir A. Rawashdeh

Department of Electrical and Computer Engineering, University of Michigan-Dearborn: Dearborn, MI 48128, USA

Abstract

Input images are the main source of information for vision-based algorithms. The presence of raindrops in input images degrades their quality and, consequently, reduces the quality of the target vision-based algorithm that consumes them. Many image restoration algorithms were proposed in the literature to remove rain presence in images to improve the input image quality. These algorithms, however, cannot remove all the raindrop presence and sometimes introduce undesirable side-effects, such as the blurring rain-occluded sections of the image and incorrectly de-raining areas in the image that are clear. We hypothesize that a comparable performance improvement can be achieved by decreasing the sensitivity of vision-based algorithms to noisy input images, rather than denoising these images, through the process of de-raining. To test this hypothesis, we evaluated the performance of state-of-the-art object detection and semantic segmentation models, with de-rained image datasets used as input, and compared it to that performance of the same models, retrained with rained image sets. Results showed that the performance of the retrained models was better than that of the baseline detector with de-rained images used as input.

Keywords

Raindrop Detection, Rain Removal, De-raining, Transfer Learning, Deep Neural Network, Automotive, Image Distortion, Object Detection, Semantic Segmentation

1 Introduction

Automotive systems including vision-based applications are highly regulated and are required to meet high performance and safety standards. This means that these systems must operate under all conditions, favorable or adverse. The quality of the system inputs has a direct impact on its performance, in the sense that noisy inputs result in degradation in system performance.

Two approaches are usually implemented to reduce the effect of noisy inputs on system performance, denoising the inputs, or reducing system sensitivity to noise. Filtering analog signals and debouncing digital ones are two examples of common input signal denoising techniques. Predictive modeling and sensor fusion are system design techniques that lead to reduced system sensitivity to noisy inputs.

Rain is a type of adverse weather condition that degrades the quality of images and the performance of vision-based algorithms that consume them. In a previous research work [1], we showed that the performance of state-of-the-art object detectors (including YOLOv3, RCNN, and SSD) greatly degrades when applied to image sets containing adherent raindrops. Test results showed the drop in performance of the tested object detectors was as high as 77%, as measured by the total number of objects detected and the recall metric [1].

Most of the research work (see, for example, [2, 3, 4]) is focused on image restoration of rained images, by applying a de-raining process on them. As we have shown in our survey paper on adherent raindrop removal techniques [5] , none of the reviewed de-raining algorithms could perfectly restore the rained images to resemble the clear ones. The improvements in deep-learning and convolution neural networks (CNN) opened the door for a new set of de-raining techniques that, generally, achieved better performance levels compared with classical machine learning algorithms. Classical de-raining techniques use some set of features, such as raindrop shape, size, intensity, chromatic, and optical properties, to create the raindrop detection model [6, 7].

CNN models, require large sets of data for training. For de-raining algorithms, an accurate mask of raindrops is needed to train the CNN model. This requires a large set of matched clear and rained images to generate such a mask. Constructing such a dataset of paired images is not an easy task, due to the unpredictability of rain and the background objects, and due to the variations of the raindrop sizes, shapes, and orientations.

We propose a different approach to improving vision-based system performance under rainy conditions. Rather than denoising (de-raining) the input images, we propose to reduce the system's sensitivity to noisy inputs. This can be achieved by retraining models that are already trained with clear image sets, with matching sets of rained images. This approach eliminates the need for developing and training the de-raining network. Furthermore, retraining networks designed for common automotive vision applications (e.g., traffic sign recognition, object detection, lane detection) is efficient and fast, since it employs transfer learning, whereas training a de-raining network may need to be done from scratch.[1] Table 1 shows some differences between the input denoising approach and the network retraining one.

[1] In practice, system developers employ both input denoising and system desensitizing to outside noise, in order to achieve optimal system robustness. We studied the two approaches separately, since our goal is to evaluate their individual contribution to the robustness of vision-based systems, and compare these contributions quantitatively.

Table 1: differences between input denoising and network retraining approaches for improving vision system performance

Performance boost approach Comparison points	De-raining of input images	Retraining model with rained images
Training type	Training from scratch	Transfer learning
Training dataset size	Large	Relatively small
Input type	(Clear, Rained) pair dataset plus raindrop mask and/or structure or texture maps	(Clear, Rained) pair dataset
Objects of interest	Natural raindrops with weak borders and variable shapes, sizes, and orientations	Man-made objects with strong boundaries and uniform shapes (e.g., Vehicles, traffic signs, road marks)

To test our hypothesis, we trained state-of-the-art object detector and semantic segmentation models with a clear image set, then retrained it with generated raindrops dataset. A comparison of the models' performance with clear, rained, and de-rained images showed that the retraining approach showed better performance improvement than the de-raining approach. Our contributions can be summarized as follows.

- We are proposing and demonstrating the feasibility of transfer-learning and relearning methods as an alternative means of improving DNN vision models against degradations caused by the presence of adherent raindrops in the input images.

- We are demonstrating the limitations of a state-of-the-art de-raining process on rained images. In particular, we are demonstrating that the target vision models that are represented by object detection and image segmentation DNN models are prone to degradation under the state-of-the-art de-raining process.

- As an alternative to the limitations of the state-of-the-art method for de-raining, we are synthesizing proper raindrops as the training sets for object detection and image segmentation DNN models.

The remainder of this paper is arranged as follows. Section 2 is a summary of some research work related to ours. In section 3, we describe the data collection process, and in section 4 we describe the model training process and the test cases we have developed for our research work. Section 5 is a summary of the test results with a detailed analysis. The research conclusion and a discussion of future work are presented in Section 6.

2 Related work

In this section, we provide an overview of some of the recent DNN-based models, which represent the state-of-the-art in the de-raining algorithms. To the best of our knowledge, model retraining and transfer-learning were not presented in any published work, as a viable alternative to the de-raining process, to improve overall vision-based system performance. Retraining and transfer learning were proposed as a viable solution to DNN models for other applications, as we will describe in this section.

Qian et al. [8] developed an adherent rain de-raining model, based on the Generative Adversarial Network (GAN) architecture. To prove the usefulness of their model for vision-based applications, they used the Google Vision API (https://cloud.google.com/vision/) to test if the de-rained output of their model provided any improvement on the object recognition performance. The results showed an average of 10% performance improvement over the same tests done with rained images. The use of Google Vision API provides an independent validation method for the applicability and usefulness of the de-raining approach in vision-based applications. One drawback was that for smaller objects, such as fences and cottages, the recognition algorithm performed worse with the de-rained image set, as compared to the rained one.

Peng et al. [9] developed a de-raining model based on the encoder-decoder architecture. The Google Vision API, like Qian et al. [8], to evaluate the usefulness of the de-rained images that were synthesized by their model for real vision-based applications. The results showed that their images performed better with the Google Vision API as compared to Qian's reported test results. It was not clear from their paper, however, if they had encountered the same degradation in performance on small objects, as what had been reported by Qian et al. [8].

Alletto et al. [10] developed a self-supervised de-raining model, based on the GAN architecture with Spatio-temporal augmentations. To test their model's visual and temporal consistency, they used the I3D network [11] as the inception network, and calculated the "Fréchet Inception Distance" (FID) [12], for clear versus de-rained video sequences. Test results showed that the FID score for their model was smaller (better) than that of Qian et al. [8], and Wang et al. [13] models. The FID is a good indicator of the similarity of the statistics of synthetic (de-rained) images to real (clear) ones. The main drawback of this process was that, like Qian et al. [8], the rained data set used for training and testing was not of real adherent raindrops but rather of images with synthetic raindrops, created by spraying water of a tilted glass surface. As we will show in our experiments, the quality of de-rained images from our true rain dataset using the model by Alletto et al. [10] was lower than the quality scores they reported using their synthetic raindrop dataset.

Talukdar et al. [14] trained TensorFlow implementation of some state-of-the-art object detectors, including SSD, Faster-RCNN, and R-FCN using synthetic images of packed food products in a refrigerator. They developed different datasets and retrained the detectors using these datasets. Using Precision, Recall and mean Average Precision (mAP) as performance evaluation metrics, they concluded that the selection of retraining datasets was of great importance in the successful transfer-learning from the original to the desired models. A dataset that is balanced in terms of scene diversity, variance, and noise is desirable for transfer learning. Our image datasets were extracted from video recordings of different drive

cycles on highways and local roads. These datasets were naturally balanced in terms of noise and the variety of objects commonly encountered on the road.

Nguyen et al. [15] used transfer learning to solve the problem of the lack of large facial expression datasets. Their approach was based on transfer learning and proved successful in transferring the emotional facial expression from one dataset to another, without discarding prior learned information. In our work, the retrained models with rained images also retained the prior learned information from the baseline models that were trained with rain-free images and even showed some improved performance when tested with the same rain-free images.

Tabik et al. [16] used the MNIST handwritten digits classification problem as a case study to evaluate the effect of data preprocessing on the accuracy and training speeds of three CNN models, LeNet, Network3, and DropConnect. They applied a combination of different preprocessing techniques, including centering, elastic deformation, translation, rotation, cropping, and resizing. They reported an average of 0.74% accuracy improvements of networks trained with preprocessed datasets versus the original non-preprocessed ones.

Nemade et al. [17] studied the effect of geometric transformation on the annotation performance of various CNNs such as Alexnet, GoogleNet, ResNet50, and DenseNet201 on the Corel dataset. Different preprocessing techniques were investigated including cropping, horizontal flipping, vertical flipping, rotation by 45°, rotation by 90°, and rotation by 180°. Results showed that the contribution of data preprocessing to the enhancement of network performance was related to the network structure and the type of classes in the dataset. The ResNet50 model, for example, was more powerful for cropping, vertical flipping, and rotation 180° operations versus the rest of preprocessing operations. The classification of the “Cat” class benefited most from the rotation by 90°, using the ResNet50 network. In general, DenseNet201 outperformed other models, when trained with preprocessed datasets.

MATLAB provides a slew of data preprocessing functions that can be used in machine learning and deep-convolutional learning applications [18]. In our work, we used some preprocessing operations on the training datasets that included cropping, rotation, flipping, and intensity remapping, as described in the data preprocessing section of this paper.

3 Data Collection and Data Preprocessing

We used different datasets for training and testing the object detection network and for training and testing the image semantic segmentation network. Figure 1 depicts the main process steps for data collection and generation.

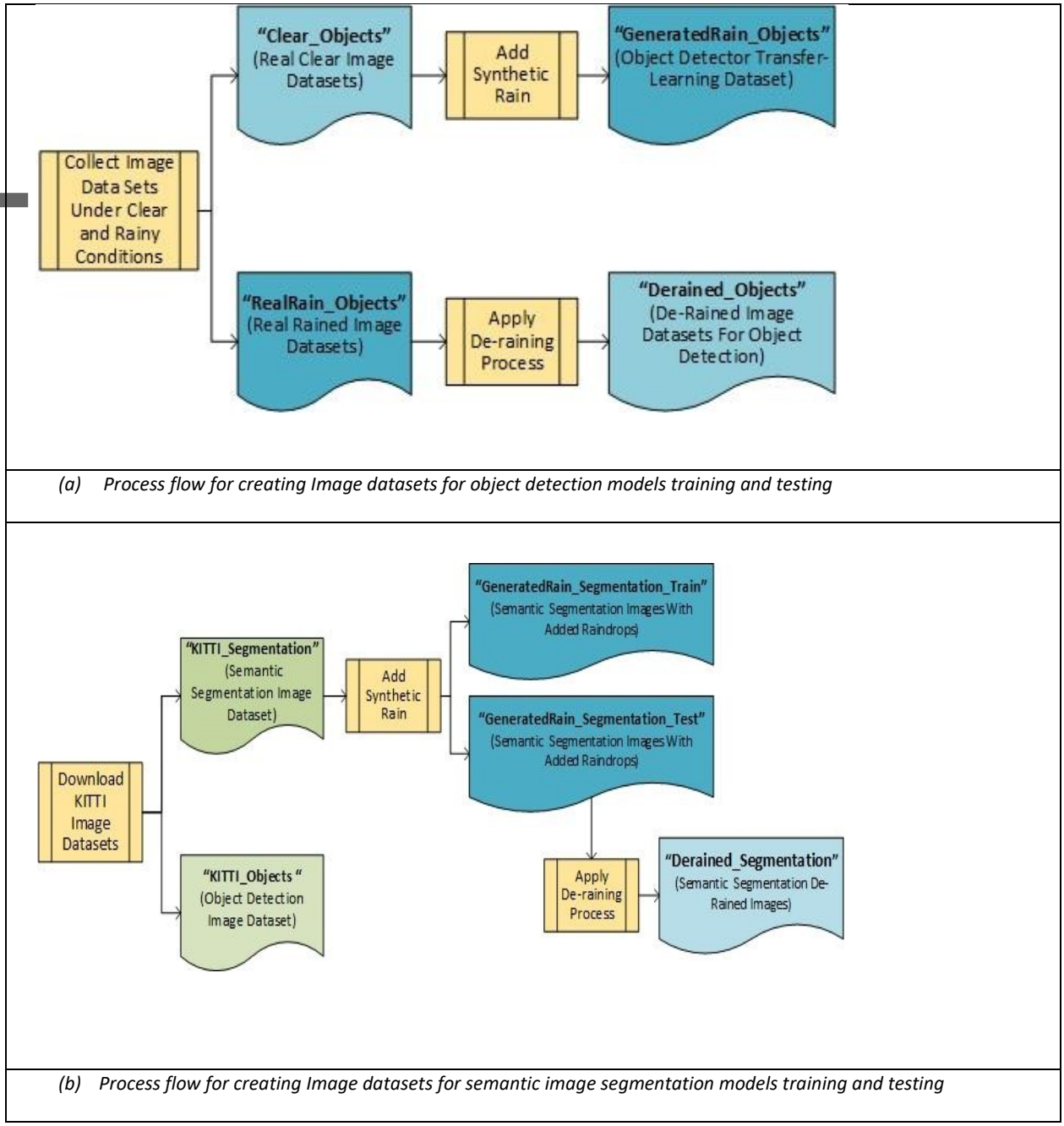


Figure 1: Process flows for generation of training and testing Datasets for the object detection and semantic segmentation models

3.1 Data Preprocessing

We employed the following image preprocessing techniques to prepare raw images for utilization in the training and testing of the models.

3.1.1 Image Cropping

Due to the utilized method of onboard camera attachment, the captured images included a part of the trunk of the ego vehicle. This image section occupied the lower 20% section of the image and carried no useful information. We used image cropping to eliminate the trunk section from the captured images.

We also used image cropping to select a Region of Interest (ROI) from the rained imagers that were used as inputs to the de-raining algorithm. The ROI was selected as a 500x500 pixel area from the center of the original images. This size provided optimal de-raining results. We selected through image cropping the same ROI from the rain-free images that matched the rained ones, to be used for performance comparison analysis.

3.1.2 Image Intensity Adjustments

The KITTI dataset included images captured under clear weather conditions. We used intensity adjustment to add an overcast effect to synthesized rained image datasets that used the KITTI datasets as their baselines. To create the overcast effect, the intensity range of each baseline image was remapped into a tighter intensity range around its average intensity. This has the effects of lowering overall image intensity, and reducing the image color content, producing a less vivid image.

3.1.3 Dataset Augmentation Techniques

As part of the model training process, we enabled the automatic augmentation in the two models which allowed for the application of arbitrary image preprocessing techniques, including the cropping, rotating, flipping, and resizing of original images from the training datasets. This is a widely used process in DNN model training since it allows for increasing the size of the training dataset, without adding actual new training samples.

3.2 Object Detection Datasets

We used the 2d “Object Detection Evaluation” from the KITTI Vision Benchmark Suite [19] to train the baseline Yolo3 for detecting objects under rain-free conditions. We used this dataset with labeled objects for the training and testing tasks. This dataset consisted of 7482 color images of different drive-cycles and showed objects commonly encountered on the road. We modified the format of the label text files to be compatible with MathWorks’s deep-learning object label format. The five object classes we chose for the baseline were 'Pedestrian', 'Truck', 'Car', 'Cyclist', and 'Van'.

We collected our own dataset of paired clear and rained images, captured under different driving and weather conditions. We used the (ELP-960P2CAM-V90-VC) dual-lens stereo camera that was positioned approximately 10 cm away from the windshield. The windshield wiping event was used as a trigger to capture rained and clear image pairs where the frame before the wipe event was captured as the rained image and the frame after it as the clear image. We selected 1162 images with the maximum number of objects per image to construct the ‘Clear_Objects’ and the ‘RealRain_Objects’ datasets. The ‘GeneratedRain_Objects’ was constructed by adding generated raindrops to the ‘Clear_Objects’ dataset, using our previously-developed raindrop simulation model [20]. We chose 'Pedestrian', 'Truck', 'Car', and 'None' as the classes for the retrained Yolo3 model, and used MATLAB’s ‘Image Labeler’ app to label objects in the datasets. The new classes were intentionally different from the ones used in the baseline model. In transfer learning, the new model is retrained to detect different classes of objects than the ones the baseline model is trained to detect.

We applied a state-of-the-art de-raining algorithm developed by Quan et al. [21] [22] on the ‘RealRain_Objects’ dataset, to create the ‘Derained_Objects’ dataset. Figure 2 shows image samples from the different datasets we used in the object detection training and testing.

There are other publically available implementations of other DNN-based algorithms, including the implementation of Qian et al [8] [23], and Yasarla and Patel [24] [25]. We chose Quan et al. [22] implementation[2] since, first, it was an improvement over Qian's algorithm for image de-raining, given that Qian's algorithm [8] is becoming the new standard of adherent raindrop deraining. Quin et al. also reported de-raining results that surpassed other DNN-based algorithms, including Eigen et al. [4] and Isola et al. [26]. Yasarl and Patel's algorithm was developed for de-raining of falling rain streaks from images. As shown by Peng et al. [9], these rain streak removal algorithms do not yield satisfactory results compared to the ones designed for adherent raindrop removal, even when they retrained those algorithms on the same adherent raindrop datasets[3].

(a) KITTI Object Dataset

(b) Our Dataset - Clear

(c) Our Dataset – Rained

(d) Our Dataset – De-rained

Figure 2: Image samples from the different datasets we used in our research work. The KITTI dataset was captured under clear weather conditions, whereas our dataset was captured under rainy conditions.

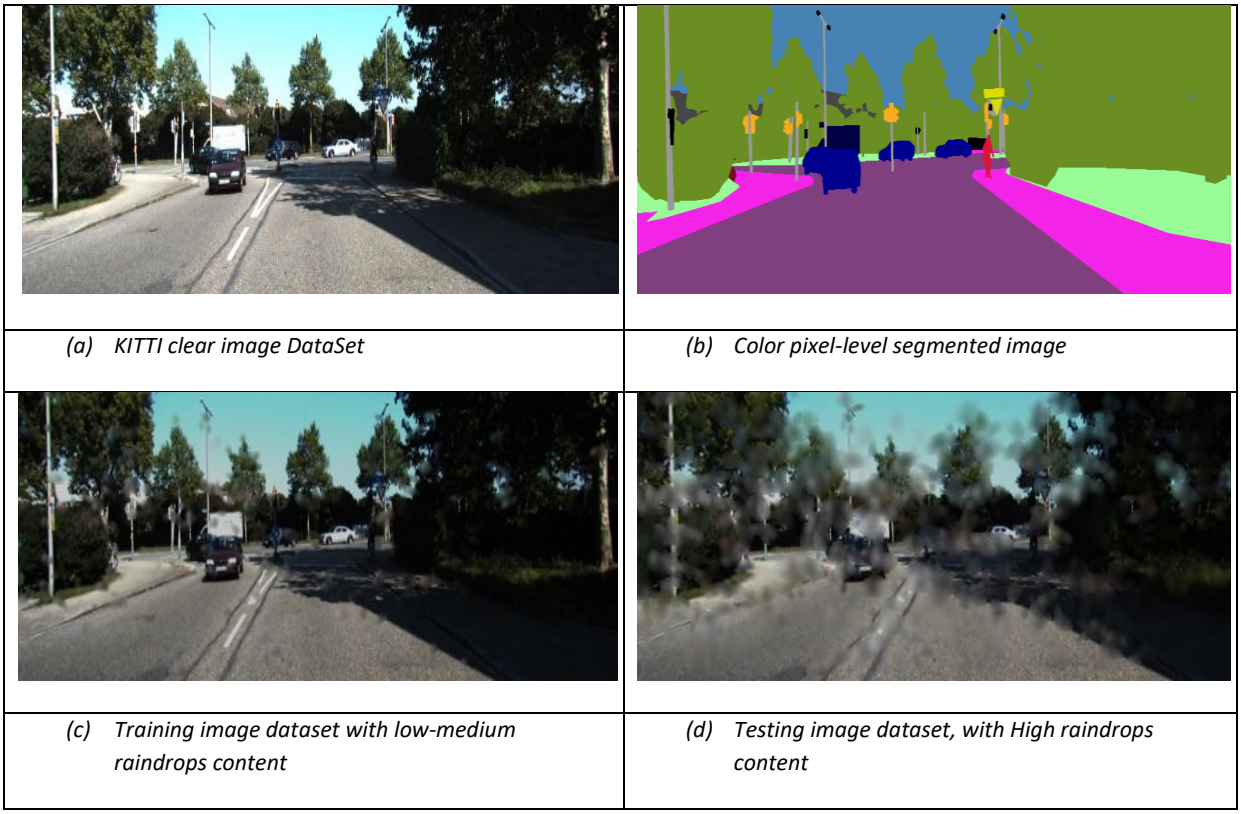
[2] To avoid any issue that may stem from inaccurate implementation the de-raining algorithm Proposed by Quan et al. [21], we used their implementation of that algorithm [22] with no modifications.

[3] The falling rain streaks and adherent raindrops are two different problems in terms of type of degradation they cause to input images. The characteristics (features) of rain streaks and adherent raindrops which the DNN system uses for learning are also different. it is not surprising based on the above that retraining a rain streak removal DNN on adherent raindrop datasets does not yield satisfactory results.

3.3 Image Segmentation Datasets

For the image segmentation, we used the "Semantic and Instance Segmentation Evaluation" dataset from the KITTI Vision Benchmark Suite [19], to train the baseline image segmentation network. The dataset consists of 200 images of street scenes, taken under clear weather conditions. Pixel-level color and gray-scale segmented images and instance-level segmented images are also included in the dataset. We grouped the 35 segmentation labels that the KITTI dataset provided, into six labels, 'Sky', 'Vehicle', 'Person', 'Background', 'Road', and 'Unlabeled' to construct the 'KITTI_Segmentation' dataset. We used the clear images and the color pixel-level segmented images to train the baseline segmentation model. For the retraining process, we used our raindrop simulator model to add generated rain at different intensity levels to the 'KITTI_Segmentation' dataset and created the 'GeneratedRain_Segmentation_Train' and 'GeneratedRain_Segmentation_Test' datasets. Structural Similarity Index (SSIM) was used as an indicator of the rain intensity for each image in the datasets. A smaller SSIM score indicates more raindrop content in an image and vice versa. An overcast effect was added to simulate real rain lighting conditions, as follows. The color image was first split into its red, green, and blue channels. The mean intensity for each color channel was then calculated, and the pixel intensities were remapped into a tighter intensity range around the mean intensity. This effectively reduced the color content for each channel, a natural consequence of reduced illumination under overcast conditions. The final recombined image looked darker and less colorful than the rain-free one.

Figure 3 shows examples of the image datasets we used to train and test the segmentation network.



(a) KITTI clear image DataSet

(b) Color pixel-level segmented image

(c) Training image dataset with low-medium raindrops content

(d) Testing image dataset, with High raindrops content

Figure 3: Datasets used for training and testing the image segmentation network. KITTI Semantic and Instance Segmentation Evaluation dataset, (a) and (b) is used to train the baseline segmentation network. We added an overcast effect and generated rain to the image sets in (c) and (d) to train and test the segmentation network under rainy conditions.

Table 2 shows a summary of the datasets we used in the object detection and image segmentation networks.

Table 2: A List of the datasets used in our research for training and testing the object detection and segmentation networks

Set ID	Usage
KITTI_Objects	Train the baseline detector using the KITTI dataset.
Clear_Objects	Retrain the baseline detector using our rain-free dataset
GeneratedRain_Objects	Retrain the baseline detector using our generated rain dataset
RealRain_Objects	Test the baseline and retrained detector under real-rain conditions.
Derained_Objects	Test the baseline detector using de-rained.
KITTI_Segmentation	Train the baseline segmentation model using the KITTI dataset.
GeneratedRain_Segmentation_Train	Retrain the baseline segmentation model using the generated-rain dataset
GeneratedRain_Segmentation_Test	Test the retrain segmentation model using the generated-rain dataset
Derained_Segmentation	Test the baseline segmentation model using de-rained images

4 Models Training Process and Testing

In this section, we will describe the training process and test cases we conducted for the object detection and semantic image segmentation models.

4.1 The Object Detection Model

4.1.1 Baseline Model Setup and Training

We used MathWorks's Yolov3 object detector example [27] as our starting model. The detector was based on SqueezeNet [28] Deep Neural Network (DNN with a relatively small architecture That allowed us to conduct all our training and testing on a desktop with outdated specifications (AMD FX-8350 with 16 GB of DDR3 RAM and an Nvidia 1050Ti GPU). Figure 4 shows the training stages and datasets used in each stage for the YOLOv3 object detector model.

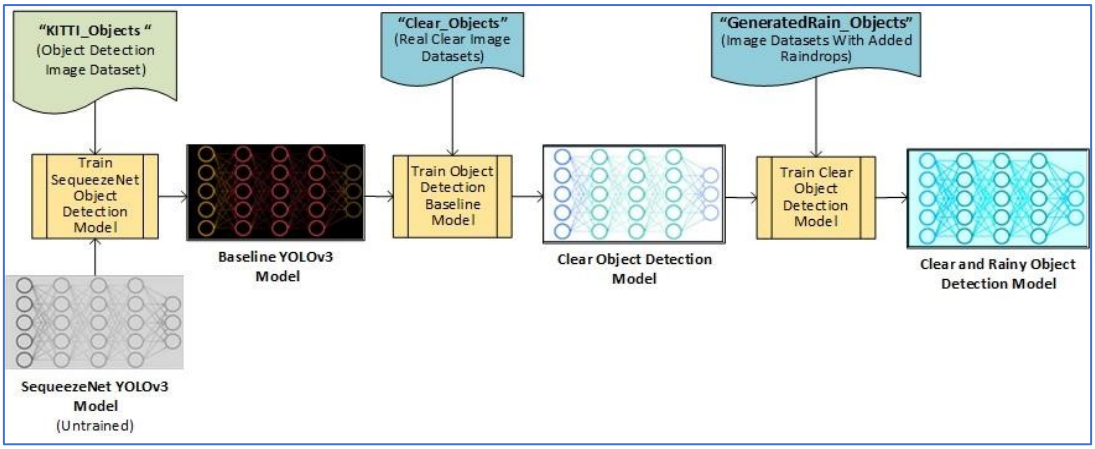


Figure 4: A Flow diagram showing the different YOLOv3 model training stages and the training dataset used in each stage.

The training process is described below.

4.1.2 Train the Automotive-specific Object Detector

We trained the starting model using the "KITTI_Objects" dataset, split as 70% training and 30% testing to establish an automotive-specific object detector. The number of epochs was set to 200, with a minimum batch size of 8 and a maximum learning rate of 0.001. We used image augmentation to increase input dataset size, without adding more images to the training dataset. We used six anchors to improve image object fitting.

Two common metrics used to evaluate the detection performance are the Average Precision (AP) and Log-Average Miss Rate (LAMR). The AP is calculated as the area under the curve the Precision-to-Recall curve. Similarly, the LAMR is calculated as the area under the curves representing the mapping between Miss Rate (MR) and False Positive Per Image (FFPI) metrics. We used MATLAB's "evaluateDetectionPrecision" function to calculate the AP score, and the "evaluateDetectionMissRate" function to calculate the LAMR score. Table 3 shows the statistical results of testing the resultant object detector using the remaining 30% of the "KITTI_Objects" dataset. Figure 5 shows an example image from the test dataset with detected objects annotated.

Table 3: The Average Precision and Log-Average Miss Rate scores, as calculated for the five object classes in the Automotive-Domain Object detector. Larger Average Precision scores and smaller Log-Average Miss Rate scores are desirable for better detection performance.

Object Class	AP	LAMR
Pedestrian	0.59	0.45
Truck	0.90	0.08
Car	0.81	0.37
Cyclist	0.64	0.34
Van	0.81	0.19

Figure 5: An example of the output of the Yolov3 detector that was trained in stage 1. The objects are identified with a bounding box, with a class tag and detection confidence level shown for each object.

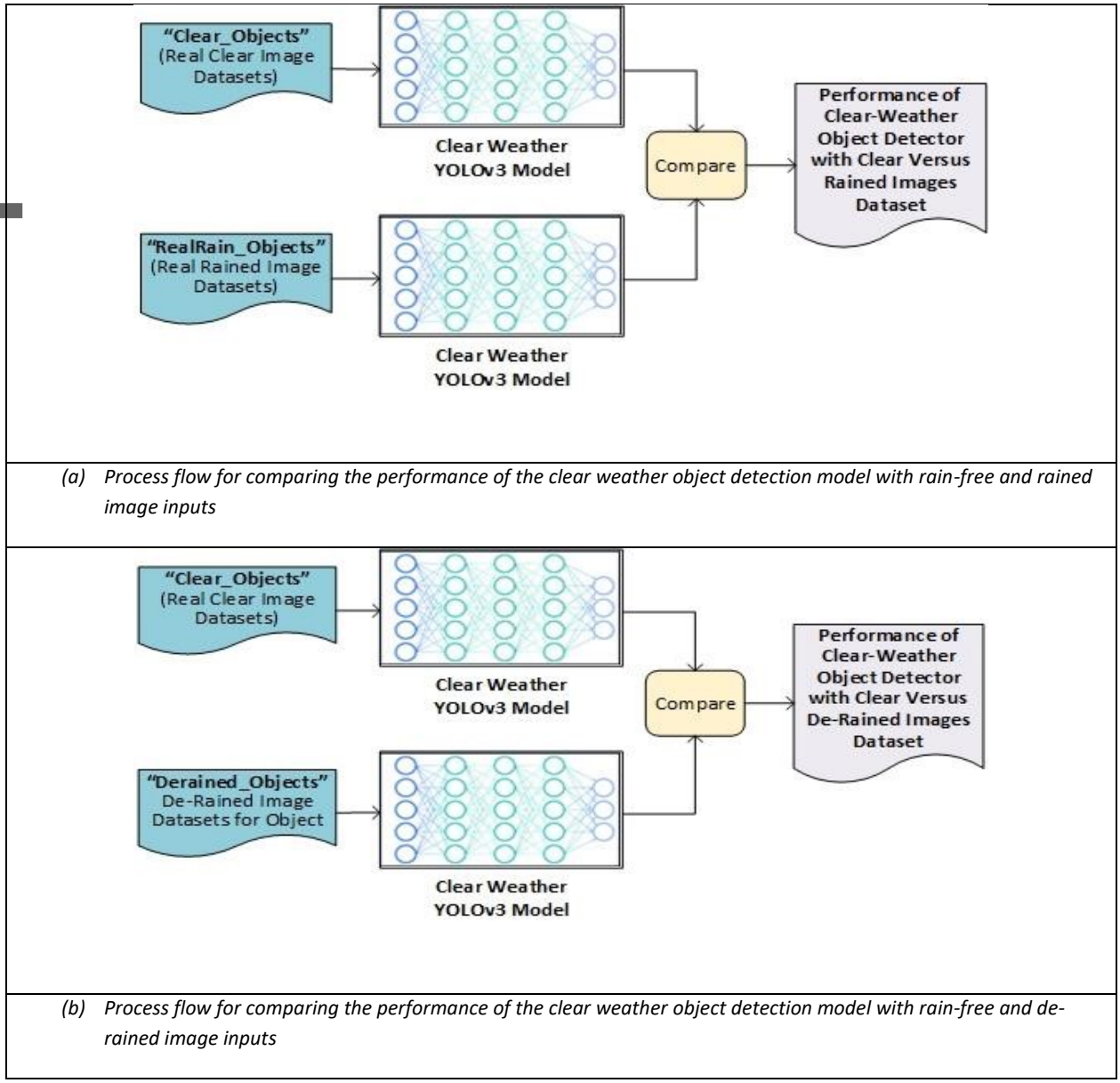
4.1.3 Train the Rain-free Object Detector

In this stage, we used the "Clear_Objects" dataset to retrain the Yolov3 network from the previous stage, to detect three different object classes, 'Pedestrian', 'Truck' and 'Car'. We retrained the object detector with very little change to the actual DNN structure. Since the dataset size in this stage is smaller than the one used in the previous stage, we increased the number of epochs to but we kept all other training parameters the same. We then tested the retrained detector using the "RealRain_Objects" dataset, to evaluate the amount of performance degradation due to the presence of raindrops. We also tested the retrained object detector on the "Derained_Objects" dataset, to evaluate if there were any performance improvements using de-rained images versus rained ones. Figure 6 shows the process flows for evaluating the Clear Weather object detector model from this stage using rain-free, real rained, and de-rained image datasets.

As expected, the detection performance of the detector retrained with the rain-free dataset degraded considerably, when tested using the rained image dataset. This is indicated in both decreased AP scores and increased LAMR scores for all three object classes. In addition, the performance of the retrained detector was worse with the de-rained dataset than with the original rained one. Table 4 shows a summary of the AP and LAMR performance metrics for the three object classes using rain-free, rained, and de-rained images. Figure 7 shows an example image of object detection at this stage.

Table 4: The Average Precision and Log-Average Miss Rate scores, as calculated for the three object classes in the rain-free Object detector. As shown in the table, there is a big degradation in detection performance when using rained images, and an even larger degradation when de-rained images are used.

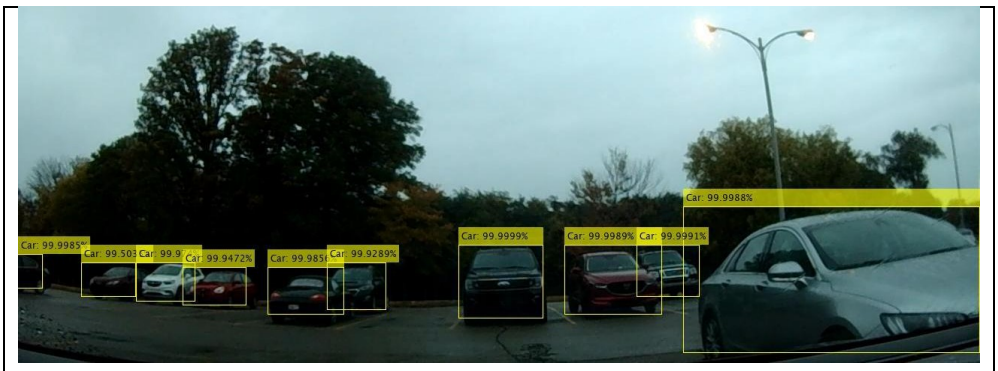
	Rain-free		Rained		De-rained	
	AP	LAMR	AP	LAMR	AP	LAMR
Car	0.92	0.09	0.36	0.63	0.18	.81
Truck	0.94	0.11	0.73	0.46	0.57	0.65
Pedestrian	0.64	0.36	0	1	0	1



(a) Process flow for comparing the performance of the clear weather object detection model with rain-free and rained image inputs

(b) Process flow for comparing the performance of the clear weather object detection model with rain-free and de-rained image inputs

Figure 6: Comparing the performance of clear weather object detectors with rain-free, rained, and de-rained image datasets.



(a) Object Detection on a clear image

(b) Object Detection on rained image

(c) Object Detection on de-rained image

Figure 7: An example of the output of the Yolov3 detector that was trained in stage 2 using clear, rained, and de-rained datasets. The objects are identified with a bounding box, with a class tag and detection confidence level shown for each object. Not much rain content was removed by the de-raining algorithm and no detection performance improvement in the de-rained image compared to the rained one.

4.1.4 Train the Rained Object Detector

For this stage, we used the "GeneratedRain_Objects" dataset to retrain the YOLOv3 object detector that we had trained in the previous stage. We then tested the retrained detector using the "RealRain_Objects" real rainy image set. Table 5 shows the AP and LAMR performance metrics for the three object classes under rained conditions. The detection performances for the 'Car' and 'Truck' classes were on-bar with those obtained with the detector from the previous stage that was trained with a Rain-free image dataset. Retraining the detector model with simulated rained images allowed it to overcome the raindrop-related image degradation, and perform at levels comparable to those under rain-free conditions. The detection performance for the Pedestrian class is still very low (AP≈0, LAMR≈1). This is because there are much fewer instances of Pedestrians in the dataset than cars and trucks.

The AP metric is calculated as the area under the curve that represents the Precision-to-Recall relation. Similarly, the LAMR is calculated as the area under the curves that represent the mapping between Miss Rate (MR) and False Positive Per Image (FFPI) metrics. This type of calculation is useful since it represents the entire curve (Precision/ Recall or MR/FPPI) by a single reference [29]. It does seem, however, to penalize classes with low-occurring instances in the form of very low AP and very high LAMR scores.

Table 5: The Average Precision and Log-Average Miss Rate scores, as calculated for the three object classes in the rained Object detector.

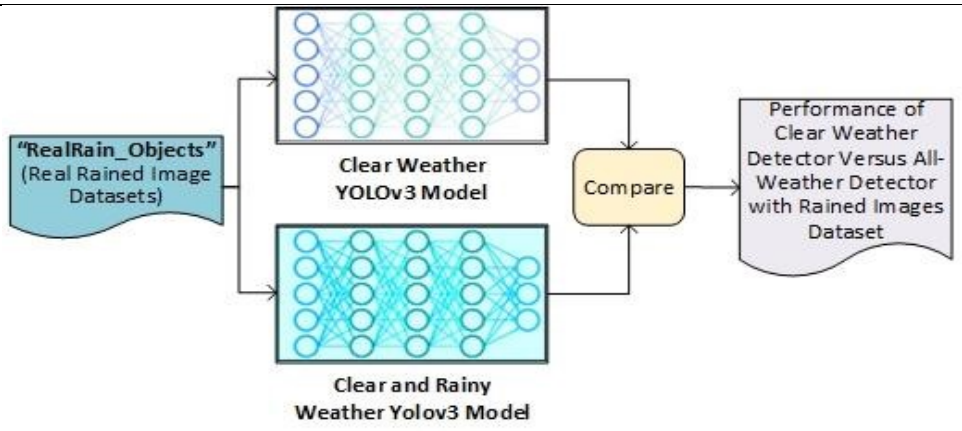
Object Class	AP	LAMR
Car	0.91	0.06
Truck	0.95	0.08
Pedestrian	0	1

To verify that the retrained detector with simulated rain retained the information learned by the original rain-free-trained detector model, we tested its performance with the rain-free "Clear_Objects "dataset. The results shown in Table 6 are identical to those shown in Table 4 for the rain-free dataset, which proves that the retrained model has retained the information it has learned from the original model.

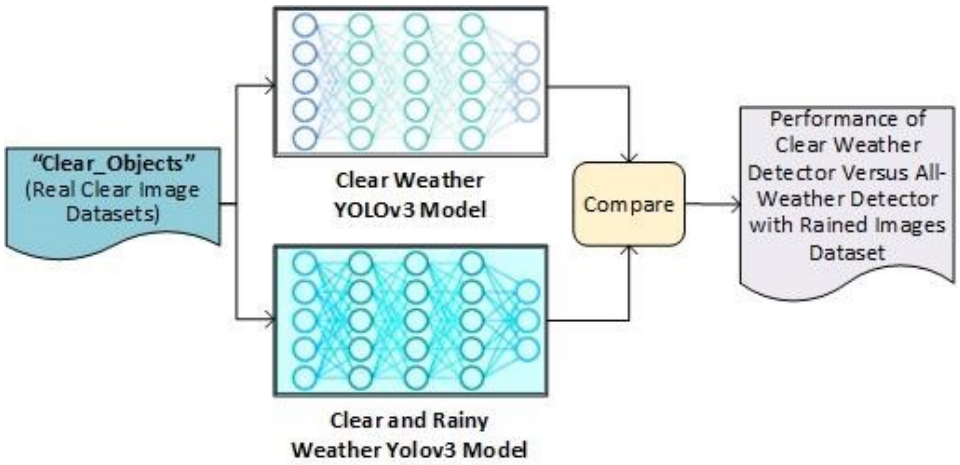
Table 6: The Average Precision and Log-Average Miss Rate scores, as calculated for the three object classes in the rained Object detector

Object Class	AP	LAMR
Car	0.92	0.09
Truck	0.94	0.11
Pedestrian	0.64	0.36

Figure 8 shows the process flow for comparing the performance of the Clear and Rainy object detection model (trained in this stage), using real rained (a) and rain-free (b) image datasets.



(a) Process flow for comparing the performance of the clear weather versus clear and rainy object detection models with real rained image inputs



(b) Process flow for comparing the performance of the clear weather versus clear and rainy object detection models with rain-free image inputs.

Figure 8: Testing the detection models with real rained and rain-free images, to verify detection performance boost (a), and the retention of previously learned detection knowledge (b).

4.2 The Image Segmentation Model

4.2.1 Baseline Model Setup and Training

We used MathWorks's semantic segmentation example [30] as our starting model. The example describes the process to train Deeplab v3+ [31]. Figure 9 shows the process flow for training the semantic segmentation model.

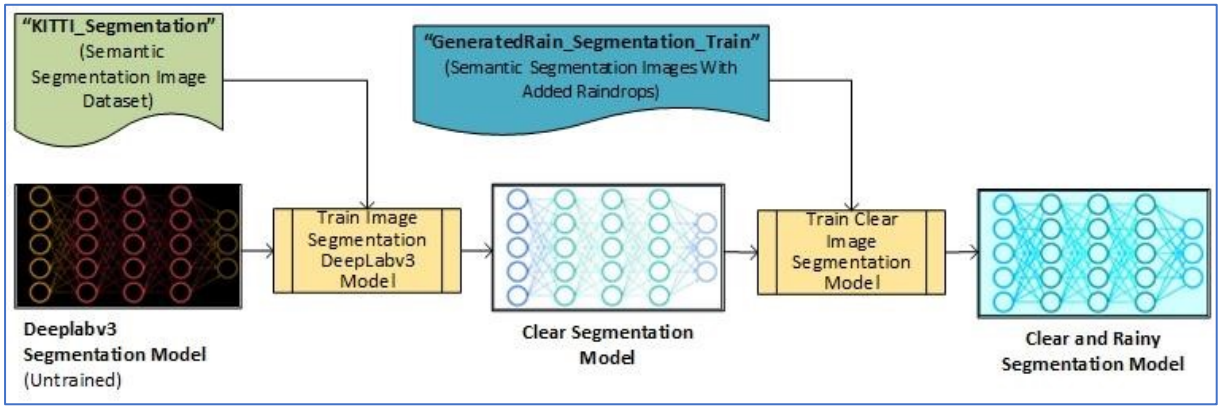


Figure 9: The process for training the rained semantic segmentation model. Starting with a pretrained DeepLapv3+ network, we train the model on a dataset that is more specific to automotive domain applications. We then retrain the segmentation model with simulated-rain images, to improve system robustness to rain-induced image degradation.

To create a baseline segmentation model for the automotive domain applications, we train the Deeplab v3+ with the "KITTI_Segmentation" dataset. We split the dataset as 75% training, 10% validation, and 15% testing, and set the maximum epochs to 300 and the minimum batch size to 8. The initial learning rate was set to 0.001 and was reduced after each concluded epoch. Data augmentation is used to increase the "effective" training dataset size without adding more images.

As a common solution to mismatched representations of segmentation classes in the training dataset, where some classes were represented more often than others, the training weights were adjusted to be inversely proportional to the frequency of occurrence of any given class. The output of this stage is the Baseline_Segmentation_DNN model which we tested using the test part of "KITTI_Segmentation".

We used the Intersection over Union (IoU), Accuracy, and MeanBFScore quality metrics to evaluate the quality of segmentation provided by the model. Accuracy is the ratio of correctly classified pixels in each class to the total actual pixel in that class. Using the True Positive (TP), and False Negative (FN) numbers, Accuracy can be given as,

$$Accuracy = \frac{TP}{TP + FN} \quad (1)$$

"IoU for a given class can be given using TP, FN, and False Positive (FP) numbers as,

$$IoU = \frac{TP}{TP + FP + FN} \quad (2)$$

MeanBFScore is a measure of the mean Boundary F1 (BF) which indicates how well aligned the predicted boundary of a given class is aligned with the actual boundary of that class.

The MATLAB function "evaluateSemanticSegmentation" can be used to calculate these three metrics in image segmentation applications. Table 7 shows a summary of model performance using the above-described metrics. The table shows that the segmentation model performed well for all classes, except the "person" class because this class was much smaller in terms of pixels count than the others, which made it more sensitive to any mismatches between the predicted and actual segmentation. The confusion matrix in Figure 10 shows a high rate of correct segmentation per class (diagonal cells) versus a low rate of incorrect classifications (off-diagonal cells).

Table 7: The Accuracy, IoU, and MeanBFScore segmentation quality metrics are shown for the classes that are identifiable by the baseline model across all images in the rain-free test dataset

Metric Label	Accuracy	IoU	MeanBFScore
unlabeled	0.724	0.266	0.417
sky	0.983	0.964	0.929
vehicle	0.967	0.828	0.837
person	0.328	0.153	0.400
background	0.940	0.924	0.923
road	0.958	0.923	0.868

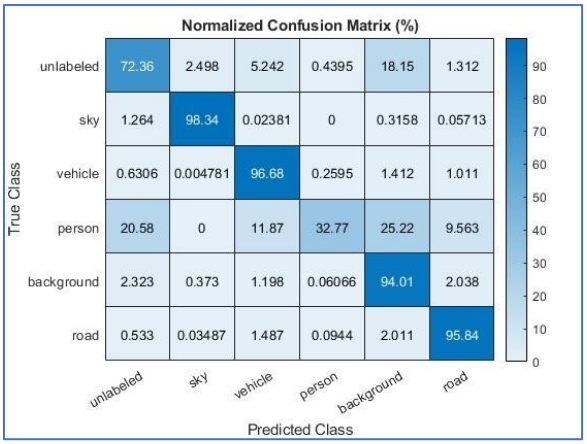


Figure 10: The confusion matrix shows the percentage of correct and incorrect segmentation of all classes supported by the segmentation model. The diagonal cells represent the percentage of correct class segmentation, and the off-diagonal cells represent the percentage of incorrect segmentation of the pixels of a given class as belonging to another class.

4.2.2 Testing the Baseline_Segmentation_DNN Model with Rained and De-Rained Datasets

To evaluate the effect of rain on the semantic segmentation process, we tested the Baseline_Segmentation_DNN model using the "GeneratedRain_Segmentation_Test" dataset. Table 8 shows noticeable degradations in segmentation quality for the rained dataset compared to the rain-free one. The confusion matrix in Figure 11 shows that for the rained dataset, the correct segmentation percentage is still larger than the incorrect one, except for the "person" class. We also observe that the highest segmentation mismatch occurred in the "background" class. Unlike the other classes, the "background" class was made of many small and disconnected segments that were adjacent to other class segments in the image. This caused the segmentation model to misclassify objects to be of the "background" class more often than the other segmentation classes.

Table 8. Segmentation quality of the baseline model when tested with the rained image set.

	Accuracy	IoU	MeanBFScore
sky	0.840	0.809	0.740
vehicle	0.750	0.463	0.482
person	0.075	0.046	0.060
background	0.920	0.813	0.825
road	0.812	0.772	0.677

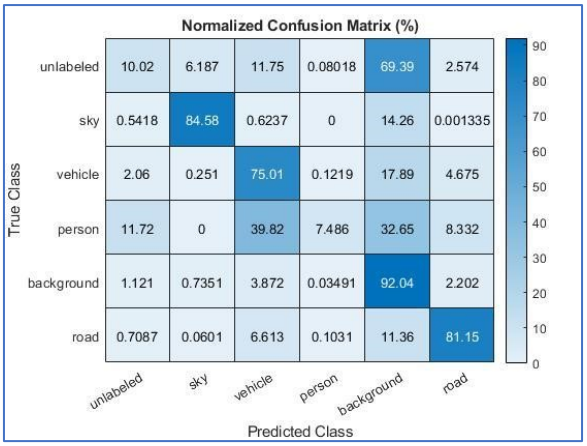


Figure 11. The confusion matrix shows a drop in the correct segmentation percentage and an increase in incorrect segmentation percentage across all classes.

We then tested the model using the "Derained_Segmentation". Table 9 shows a noticeable degradation in segmentation quality when using a de-rained dataset over the original rained one, which was summarized in Table 8 from the previous stage. The largest drop in segmentation quality is observed in the "sky" and "vehicle" classes, as indicated by the quality metrics. The confusion matrix in Figure 12 shows that only "road" and "background" classes have a higher correct segmentation percentage than incorrect ones. Another interesting observation is that most incorrect observations are classified as "background" class. The same phenomenon was observed under rain-free and rained segmentation testing which indicates a possible segmentation bias towards the "background" class, even though we used the wights reverse-frequency technique in our design and training.

Table 9: The segmentation quality metrics show lower performance of the rain-free (baseline) segmentation model with the de-rained dataset than that under rained dataset. Performance drop was highest for "sky" and "vehicle" classes and the least drop was observed for the "road" class

Metric Label	Accuracy	IoU	MeanBFScore
sky	0.374	0.244	0.419
vehicle	0.215	0.105	0.242
person	0.000	0.000	0.002
background	0.701	0.525	0.687
road	0.714	0.570	0.598

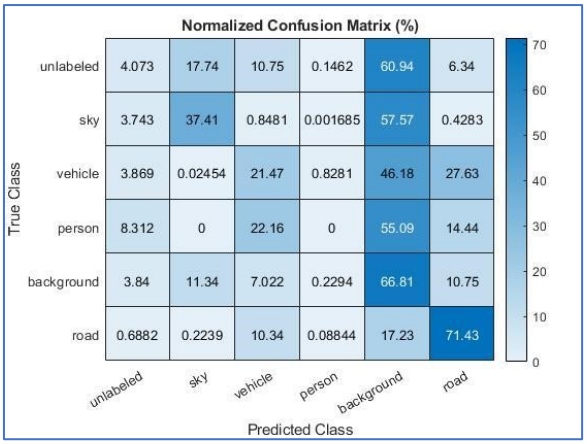
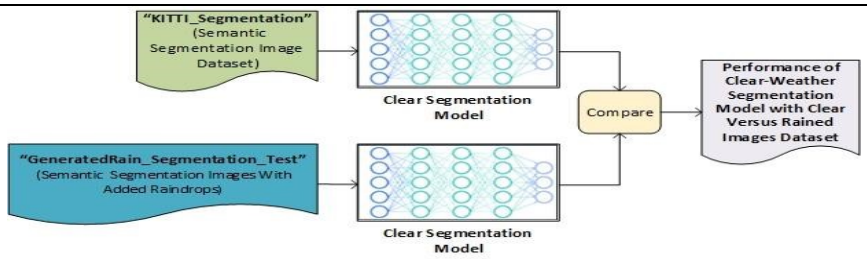
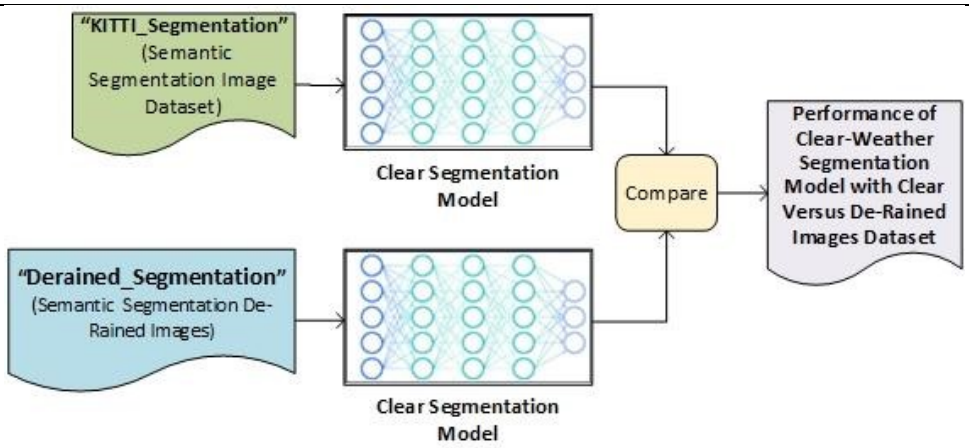


Figure 12: the confusion matrix for class segmentation results shows that only "background" and "road" classes still show more correct than incorrect segmentation under de-rained dataset and rain-free segmentation model mix. It also shows that the "background" class contributed to the most percentage of incorrect classifications.

Figure 13 shows the process flow for evaluating the performance of the Clear Segmentation Model with a clear image dataset against rained image dataset (a) and de-rained image dataset (b).



(a) Process flow for comparing the performance of the clear weather image segmentation model with rain-free and rained image inputs



(b) Process flow for comparing the performance of the clear weather image segmentation model with rain-free and de-rained image inputs

Figure 13: Comparing the performance of clear weather image segmentation model with rain-free, rained, and de-rained image datasets

4.3 Retraining the Baseline_Segmentation_DNN

We retrained the Baseline_Segmentation_DNN model from the previous steps using the "GeneratedRain_Segmentation_Train" dataset. The dataset consists of 400 images with low and medium intensity of generated raindrops added. We split the dataset 75% training, 10% validation, and 15% testing datasets and ran the training process for 200 epochs. All other hyperparameters we left intact from the previous training process. As shown in Table 10, there is a big improvement in the segmentation with the rained model compared with the rain-free model, both tested with real rain image dataset. The confusion matrix in Figure 14 shows more correct to incorrect segmentation for each class recognizable by the segmentation model. We retested the retrained segmentation model on the rain-free dataset, to verify that the retrained model retained the learned information from the previous model. Comparing the results of the retrained model in Table 11 to those of the baseline model in Table 7 shows that the retrained model retained the information learned by the baseline model, and even slightly improved on them.

Table 10: The segmentation performance metrics show that the retrained segmentation model performs on the rained dataset at levels comparable to the performance of the rain-free segmentation model that is tested with the clear dataset.

Metric Label	Accuracy	IoU	MeanBFScore
sky	0.963	0.871	0.812
vehicle	0.781	0.546	0.572
person	0.391	0.149	0.150
background	0.896	0.840	0.845
road	0.896	0.840	0.763

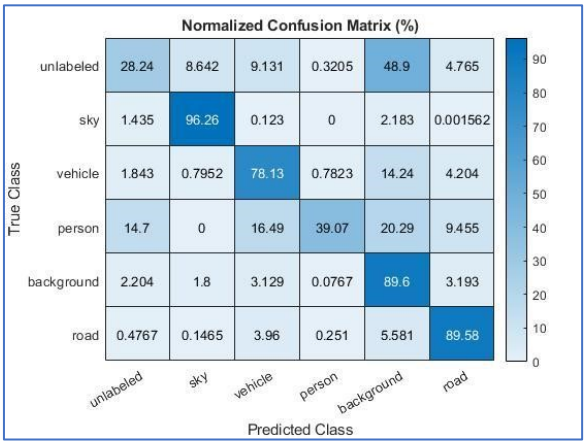
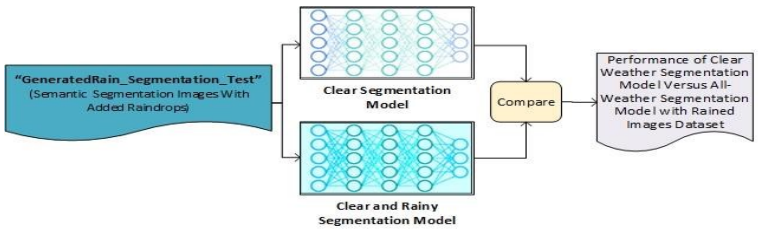


Figure 14: Testing the retrained rained segmentation model with a real rain dataset shows that A higher percentage of pixels are correctly segmented for each class than incorrectly segmented.

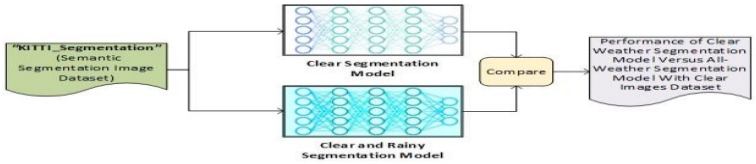
Table 11: Testing the retrained rained segmentation model shows no degradation in performance over the original rain-free segmentation model, both tested on the same rain-free dataset.

Metric Label	Accuracy	IoU	MeanBFScore
sky	0.986	0.949	0.901
vehicle	0.987	0.834	0.807
person	0.983	0.339	0.650
background	0.933	0.926	0.915
road	0.957	0.928	0.888

Figure 15 shows the process flow for comparing the performance of the Clear and Rainy Image Segmentation Models models, using rained (a) and rain-free (b) image datasets.



(a) Process flow for comparing the performance of the clear weather versus clear and rainy image segmentation models with rained image inputs



(b) Process flow for comparing the performance of the clear weather versus clear and rainy image segmentation models with rain-free image inputs.

Figure 15: Testing the image segmentation models with rained and rain-free images, to verify segmentation performance boost (a), and the retention of previously learned segmentation knowledge (b).

5 Results Discussion

We trained a YOLOv3 model to detect common objects encountered in a common drive cycle and tested git using rain-free, rained, and de-rained image sets. The detector performed well on rain-free images, but its performance degraded under rained image set input, as expected.

By analyzing the de-raining algorithm that had been developed by Quan et al. [21], we believe their model was too specific to the training and testing dataset they had used. This made it less useful for the real rain datasets we used in our research, due to the following two factors:

1. Quan's model used a training dataset that used synthetic raindrops for rained images. Real raindrops exhibit more variety in shape and size than the simple droplets formed by spraying water on a glass surface. This likely made raindrop detection harder with real raindrops than synthetic ones.
2. The synthetic dataset used in Quan's model was also taken under optimal lighting conditions which made it easier for raindrops to be detected. The overcast in the background of the real rain dataset, on the other hand, made it harder to identify raindrops by a human observer. This overcast in the real rain dataset likely affected the ability to learn raindrops by the de-raining DNN in [21]

The retrained YOLOv3 model with a simulated raindrop dataset showed great improvement of the rain-free object detector, both tested with the real-rain dataset.

The only class that did not show improved detection with the retrained rained detector was the "Pedestrian" class. We believe that two factors contributed to this limitation:

i. The size of the objects representing the "Pedestrian" class were mostly smaller than the other two class objects. This meant that these objects were more susceptible to the presence of rain, which usually occluded and distorted all or most of the pixels representing this class in the image.
ii. The number of occurrences of the "Pedestrian" object in the dataset we used for training was much smaller than the other two. We counted 15 "Pedestrian" object instances in the whole training dataset, compared to the thousands of occurrences for the other two classes. Our dataset was collected on motorways in Michigan and under rainy conditions, so the presence of pedestrians was the exception rather than the norm.

We also verified that the retrained detector performance did not degrade under rain-free conditions by retesting the rained detector with the original rain-free dataset. The retrained rained model performance was similar to that of the rain-free detector under the rain-free dataset which made us conclude that the retrained detector extended the detection capabilities by adding robustness to system input noise caused by the presence of raindrops.

The semantic segmentation test cases provided similar results to the object detection ones. The rain-free segmentation model performed well under rain-free conditions, but its performance degraded

when tested with rained image dataset. The degradation level was not as severe as that observed in the object detection application. This can be partially attributed to the fact that in the segmentation model, the classes were much larger than those in the object detection application, and thus less susceptible to the presence of raindrops in the input images.

The segmentation model trained on the rain-free dataset performed worse on the de-rained images dataset than on the original rained images dataset. The performance of the retrained image segmentation model showed considerable improvement in segmenting rained images after the baseline rain-free model was trained with the simulated raindrops dataset. Retesting the retrained image segmentation model with the rain-free dataset showed a performance improvement over the rain-free model. The performance gain can be partially attributed to retraining the rain-free model with simulated rained images that were based on the rain-free ones. We argue that the rained images acted as a transformed version of the original ones, even if the transformation caused some level of image quality degradation. In that sense, the rained images augmented the original rain-free dataset, and image augmentation is a standard technique used in the training of the DNN to improve performance.

The performance degraded even further for the de-rained image set test, a result we did not expect when we formed our hypothesis. Our results, however, align with the task-driven evaluation results reported by Li et al. [32]. Based on their own tests using different object detection algorithms, they concluded that "all existing de-raining algorithms will deteriorate the detection performance compared to directly using the rainy image" [32]. They hypothesized that the de-raining algorithms might need to be optimized to the goal of object detection. This, however, may require a specific de-raining solution to the target vision-based application and consequently reduces the useability and generalizability of the de-raining algorithms. Another study on haze removal done by Pei et al. [33] showed similar results. They concluded that "since those deraining algorithms were not trained/optimized towards the end goal of object detection, they are unnecessary to help this goal, and the deraining process itself might have lost discriminative, semantically meaningful true information" [33].

We believe that there may be no de-raining add-on fix to this problem, in a sense that a general-purpose de-raining algorithm can be plugged into the specific vision system (e.g., traffic light recognition) that would improve the system performance under rainy conditions. We believe it is possible to have a hybrid solution, employing both de-raining (denoising) of input images and reducing system sensitivity to raindrops through re-learning and transfer learning (system desensitization). This hybrid approach, however, requires further examination of what important features are removed in the de-raining process, that a given vision application is looking for, to perform its intended functionality.

Throughout our research work, we used images with synthetic raindrops added to retrain the baseline models that were previously trained on rain-free images. This process proved to be feasible and the retrained models showed a clear performance boost when tested with real rained images. The use of a synthetic dataset solves the problem of requiring large datasets with rained images for model retraining since the synthetic datasets (images with added generated raindrops) could be readily generated from the original rain-free image datasets. In addition, we observed that the retraining required a smaller training dataset and the model converged faster (fewer epochs), as

compared to the original baseline model training. This fact in the context of retraining DNN models to perform under rainy conditions implies that the retraining dataset does not need to be as large as the one used for the baseline model training (under rain-free conditions).

As a final comment on the limitations of our proposed performance enhancement approach through transfer-learning and relearning, we observed that in some test images with very heavy rain content, the retrained algorithm failed to detect any objects in the image. A human observer could not identify objects in these images with good confidence either. This aligns with the fact that, except for very special cases, the performance of a DNN classifier cannot surpass that of a human observer. We can generalize this observation to include other distortion contributors that would cause excessive degradation to the image quality. Heavy rain content and lack of sufficient illumination at night time effectively wash off all or most interesting features in an image that the DNN model relies on for classification.

6 Conclusions

We developed models for two common vision applications, namely object detection, and semantic image segmentation. We then showed that the robustness of these models to raindrop-induced image degradation could be enhanced utilizing transfer-learning, and retraining them on rained image datasets.

We also showed the benefit of using synthetic datasets for training DNN models. Both our models were trained with synthetic rained datasets that were generated through our rain simulator model and when tested with real rain datasets, showed clear performance improvements. In addition, the synthetic dataset seemed to play a secondary role as an image augmenter to the rain-free dataset which contributed to further performance improvements under clear rain conditions.

In our experiments, the de-raining process caused more performance degradation to both models, than by using the original rained images as input. We, therefore, concluded that de-raining may not be a viable approach to enhancing vision-based algorithms' performance under rainy conditions. Our current results lead us to several future different research avenues including parallelization of the discussed network retraining approaches and implementation of these algorithms on mobile GPUs and software-programmable FPGAs. Furthermore, for future work, we would like to investigate if a hybrid solution, which is based on relearning an image de-raining model, can provide us with better solutions, provided that the important features employed in the target system are not distorted by the de-raining process. Moreover, to optimize the amount of memory size for weight storage and reduce the power consumption in hardware/software-based implementations, we propose as a next step the quantization of most of the weights into two bits, by utilizing fixed-point optimization techniques, similar to the technique described by Shin et al. [34]. Finally, we would like to study some state-of-the-art de-raining models and identify which features were altered through the de-raining process, that caused the performance degradation in the target vision system.

References

[1] Y. Hamzeh, Z. El-Shair and S. A. Rawashdeh, "Effect of Adherent Rain on Vision-Based Object Detection Algorithms," *SAE International Journal of Advances and Current Practices in Mobility,* pp. 3051-3059, 2020.

[2] E. Fouad, E. Abdelhak and A. Salma, "Modelisation of raindrops based on declivity principle," in *13th International Conference on Computer Graphics, Imaging and Visualization (CGiV)*, Beni Mellal, Morocco, 2016.

[3] J. Ishizuka and K. Onoguchi, "Detection of Raindrop with Various Shapes on a Windshield," in *5th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2016),*, Roma, Italy , 2016.

[4] D. Eigen, D. Krishnan and R. Fergus, "Restoring An Image Taken Through a Window Covered with Dirt or Rain," in *2013 IEEE International Conference on Computer Vision*, Sydney, NSW, Australia, 2013.

[5] Y. Hamzeh and S. A. Rawashdeh, "A Review of Detection and Removal of Raindrops in Automotive Vision Systems," *Journal of Imaging,* vol. 7, no. 3, p. 52, 2021.

[6] M. Roser, J. Kurz and A. Geiger, "Realistic Modeling of Water Droplets for Monocular Adherent Raindrop Recognition using Bézier Curves," in *Asian Conference on Computer Vision*, Springer, Berlin, Heidelberg, 2010.

[7] H. Kurihata, T. Takahashi, k. Ide, Y. Mekada, H. Murase, Y. Tamatsu and T. Miyahara, "Rainy Weather Recognition from In-Vehicle Camera Images for Driver Assistance," in *IEEE Proceedings. Intelligent Vehicles Symposium*, Las Vegas, NV, USA, USA, 2005.

[8] R. Qian, R. T. Tan, W. Yang, J. Su and J. Liu, "Attentive Generative Adversarial Network for Raindrop Removal from A Single Image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, Utah, 2018.

[9] J. Peng, Y. Xu, T. Chen and Y. Huang, "Single-image raindrop removal using concurrent channel-spatial attention and long-short skip connections," *Pattern Recognition Letters,* vol. 131, pp. 121-127, 2020.

[10] S. Alletto, C. Carlin, L. Rigazio, Y. Ishii and Tsukizawa, "Adherent Raindrop Removal with Self-Supervised Attention Maps and Spatio-Temporal Generative Adversarial Networks," in *IEEE/CVF International Conference on Computer Vision Workshops*, Seoul, Korea, 2019.

[11] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[12] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," *Advances in Neural Information Processing Systems (NIPS 2017),* vol. 30, no. arXiv:1706.08500v6 [cs.LG] , 2018.

[13] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz and B. Catanzaro, "Video-to-Video Synthesis," in *Conference on Neural Information Processing Systems (NeurIPS)*, Montreal, Canada , 2018.

[14] J. Talukdar, S. Gupta, P. S. Rajpura and R. S. Hegde, "Transfer Learning for Object Detection using Stateof-the-Art Deep Neural Networks," in *5th International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, Delhi-NCR, 2018.

[15] D. Nguyen, K. Nguyen, S. Sridharan, I. Abbasnejad, D. Dean and C. Fookes, "Meta Transfer Learning for Facial Emotion Recognition," in *24th International Conference on Pattern Recognition (ICPR)*, Beijing, China, 2018.

[16] S. Tabik, D. Peralta, A. Herrera-Poyatos and F. Herrera, "A snapshot of image pre-processing for convolutional neural networks: case study of MNIST," *International Journal of Computational Intelligence Systems,* vol. 10, pp. 555-568, 2017.

[17] S. Nemade and S. Sonavane, "Comparative Analysis of Geometric Transformation Effects for Image Annotation Using Various CNN Models," in *Advances in Intelligent Systems and Computing*, Springer Singapore, 2020, pp. 362-369.

[18] MathWorks, "Augment Images for Deep Learning Workflows Using Image Processing Toolbox," 2021. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/image-augmentation-using-image-processing-toolbox.html. [Accessed 28 11 2021].

[19] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)}*, Rhode Island, 2012.

[20] Y. Hamzeh, Z. A. El-Shair, A. Chehade and S. A. Rawashdeh, "Dynamic Adherent Raindrop Simulator for Automotive Vision Systems," *IEEE Access,* vol. 9, pp. 114808-114820, 2021.

[21] Y. Quan, S. Deng, Y. Chen and H. Ji, "Deep Learning for Seeing Through Window With Raindrops," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), 2019.

[22] Y. Quan, S. Deng, Y. Chen and H. Ji, "ljm619/raindropAttention," 7 10 2019. [Online]. Available: https://github.com/ljm619/raindropAttention. [Accessed 1 7 2021].

[23] R. Qian, R. T.Tan, W. Yang, J. Su and J. Liu, "rui1996 / DeRaindrop," GitHub, 29 6 2018. [Online]. Available: https://github.com/rui1996/DeRaindrop. [Accessed 20 1 2021].

[24] R. Yasarla and V. M. Patel, "Uncertainty Guided Multi-Scale Residual Learning-using a Cycle

Spinning CNN for Single Image De-Raining," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, California, 2019.

[25] R. Yasarla and V. M. Patel, "rajeevyasarla/UMRL--using-Cycle-Spinning," 5 3 2020. [Online]. Available: https://github.com/rajeevyasarla/UMRL--using-Cycle-Spinning. [Accessed 1 8 2021].

[26] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Honolulu, Hawaii, 2017.

[27] MathWorks, "Object Detection Using YOLO v3 Deep Learning," MathWorks, 2021. [Online]. Available: https://www.mathworks.com/help/vision/ug/object-detection-using-yolo-v3-deep-learning.html. [Accessed 7 7 2021].

[28] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv:1602.07360v4 [cs.CV] ,* 2016.

[29] P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE transactions on pattern analysis and machine intelligence ,* pp. 743-761, 8 2011.

[30] MathWorks, "Semantic Segmentation Using Deep Learning," MathWorks, 2021. [Online]. Available: https://www.mathworks.com/help/vision/ug/semantic-segmentation-using-deep-learning.html. [Accessed 15 7 2021].

[31] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, 2018.

[32] S. Li, I. B. Araujo, W. Ren, Z. Wang, E. K. Tokuda2, R. H. Junior2, R. Cesar-Junior, J. Zhang, X. Guo and X. Cao, "Single Image Deraining: A Comprehensive Benchmark Analysis," *Computer Vision and Pattern Recognition ,* no. arXiv:1903.08558v1 [cs.CV]), 2019.

[33] Y. Pei, Y. Huang, Q. Zou, Y. Lu and S. Wang, "Does Haze Removal Help CNN-based Image Classification?," in *Proceedings of the European Conference on Computer Vision (ECCV). ,* 2018.

[34] S. Shin and W. Sung, "Dynamic Hand Gesture Recognition for Wearable Devices with Low Complexity Recurrent Neural Networks," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Montreal, Canada, 2016.