

# Normalizing images in various weather and lighting conditions using Pix2Pix GAN

by

Sanjida Tasnim

20201039

Ashif Mahmud Mostafa

23241036

Azmain Morshed

22141050

Namreen Shaiyaz

20201086

A thesis submitted to the Department of Computer Science  
and Engineering in partial fulfillment of the requirements for  
the degree of Bachelor in Computer Science or Computer  
Science and Engineering

Department of Computer Science and Engineering  
School of Data and Science  
Brac University  
January 2024

© 2024. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

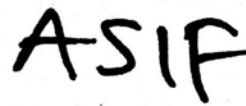
1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. I/We have acknowledged all main sources of help.

**Student's Full Name & Signature:**



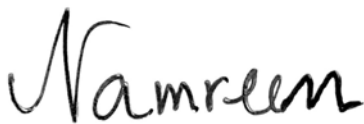
---

Sanjida Tasnim, 20201039



---

Ashif M. Mostafa, 23241036



---

Namreen Shaiyaz, 20201086



---

Azmain Morshed, 22141050

# Approval

The thesis/project titled “Normalizing images in various weather and lighting conditions using Pix2Pix GAN” was submitted by

1. Sanjida Tasnim (20201039)
2. Ashif M. Mostafa (23241036)
3. Namreen Shaiyaz (20201086)
4. Azmain Morshed (22141050)

in Fall 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 22, 2024.

Examining Committee:



Supervisor:  
(Member)

---

Md. Ashraful Alam, PhD  
Associate Professor  
Department of Computer Science and  
Engineering, BRAC University

Co-supervisor:  
(Member)

---

Md. Golam Rabiul Alam, PhD  
Professor  
Department of Computer Science and  
Engineering, BRAC University

Program Coordinator:  
(Member)

---

Md. Golam Rabiul Alam  
Professor  
Department of Computer Science and  
Engineering, BRAC University

Departmental Head:  
(Chair)

---

Sadia Hamid Kazi  
Associate Professor and Chairperson  
Department of Computer Science and  
Engineering, BRAC University

# Ethics Statement

We consciously assure that for this research that the following is fulfilled:

1. This material is the authors' own authentic work, which was not published previously.
2. The paper reflects all of the author's own research and analysis truthfully and completely.
3. The paper properly credits the meaningful contributions of co-authors and co-researchers.
4. All sources used are correctly disclosed with proper citations. A verbatim copy of the text was indicated as such by using quotation marks and giving proper references.
5. All authors have been personally and actively involved in significant work leading to this paper and will take shared responsibility for its content.
6. Finally, all authors acknowledge that the violation of the Ethical Statement rules may result in severe consequences.

# Abstract

Autonomous vehicles are widely regarded as the future of transportation due to its possible uses in a myriad of applications. In recent years, perception systems in driverless cars have had reasonable development through the various implementations of object detection systems with deep-learning algorithms. Noticeable progress has been made in this field of study as many isolated and multi-model systems have been developed and/or proposed to help overcome the shortcomings of the sensors and detection algorithms. These include research on sensing objects under varying environmental conditions (illumination, refractive indexes, weather conditions) as well as detection and removal of noise, clutter, and camouflage from the collected sensory inputs. However, in its current state, perception systems in autonomous vehicles are still incapable of accurately detecting objects in real-life scenarios using its visual/thermal camera, LiDAR, radar, and other sensors. Additionally, most systems lack the robustness to perform well under any given condition. Hence, this paper proposes to use advanced color vision techniques and Generative Adversarial Networks (GAN) to produce reconstructed images that can improve the accuracy of object detection systems for more precise predictions.

**Keywords:** Autonomous Vehicles; Object Detection; Image Normalization; Color Vision; Generative Adversarial Networks

## Acknowledgement

In preparation for this thesis, we reached out to many people, researchers, and academicians who greatly contributed to our understanding and views on this subject. In particular, we would like to express our sincere appreciation to our thesis supervisor, Professor Dr. Md. Ashraf ul Alam, and cosupervisor, Dr. Md. Golam Rabiul Alam, for their encouragement, guidance, and critics. Without their continued support and interest, this research would not have been the same as presented here. Additionally, our fellow students should also be recognized for their support. Thus, we express our sincere appreciation to all of our colleagues and others who have assisted on various occasions. Their views and tips were indeed helpful. Unfortunately, it is not possible to list all of them, but we are also very grateful to all of our family members for their continued support.

# Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iv
Abstract	v
Acknowledgment	vi
Table of Contents	vii
List of Figures	ix
List of Tables	ix
List of Equations	x
Nomenclature	xi
<b>1 Introduction</b>	<b>1</b>
1.1 Research Objectives . . . . .	2
<b>2 Problem Statement</b>	<b>3</b>
<b>3 Literature Review</b>	<b>6</b>
<b>4 Methodology</b>	<b>8</b>
4.1 Proposed methodology . . . . .	8
4.2 Data analysis . . . . .	9
4.2.1 Data acquisition . . . . .	10
4.2.2 Image preprocessing . . . . .	10
4.2.3 Augmented Data Analysis . . . . .	13
4.3 Working principle of Pix2Pix GAN . . . . .	16
4.4 Description of the Object Detection Models . . . . .	17
4.4.1 Faster R-CNN using ResNet50 . . . . .	17
4.4.2 Faster R-CNN using MobileNet v3 . . . . .	19
4.4.3 RetinaNet with ResNet50 . . . . .	20
<b>5 Implementation</b>	<b>22</b>
5.1 Pix2Pix GAN Model implementation . . . . .	22
5.1.1 Generator building . . . . .	22



5.1.2	Discriminator building . . . . .	23
5.1.3	Generator Loss Calculation . . . . .	24
5.1.4	Discriminator Loss Calculation . . . . .	25
5.1.5	Attention block . . . . .	25
5.2	Pix2Pix GAN model training and result analysis . . . . .	26
5.2.1	Training . . . . .	26
5.2.2	Pix2Pix Generated Results . . . . .	27
5.2.3	Primary result analysis . . . . .	28
5.3	Object Detection Model Implementation . . . . .	31
5.4	Object Detection Model Evaluation . . . . .	32
5.4.1	Performance Measures . . . . .	32
5.4.2	Comparative results of various models . . . . .	34
<b>6</b>	<b>Future Work and Conclusion</b>	<b>38</b>
<b>7</b>	<b>References</b>	<b>39</b>

# List of Figures

4.1.1 Methodology Flowchart . . . . .	8
4.2.1 Augmentation results after pre-processing . . . . .	12
4.2.2 Original vs augmented for Low Light (level 2) . . . . .	13
4.2.3 Original vs augmented for Bright Light (level 2) . . . . .	14
4.2.4 Original vs augmented for Fog (level 3) . . . . .	14
4.2.5 Original vs augmented for Heavy Rain . . . . .	15
4.3.1 Working principle of Pix2Pix GAN . . . . .	16
4.4.1 ResNet 50 architecture . . . . .	17
4.4.2 Stage 1 of Faster R-CNN . . . . .	18
4.4.3 Stage 2 of Faster R-CNN . . . . .	19
4.4.4 MobileNetV3 architecture . . . . .	20
4.4.5 RetinaNet structure using ResNet50 backbone . . . . .	21
5.2.1 Pix2Pix GAN output on weather condition dataset . . . . .	27
5.2.2 Pix2Pix GAN output on lighting condition dataset . . . . .	28
5.2.3 Histograms comparing augmented and normalized SSIM and PSNR scores for batch size 1 . . . . .	30
5.3.1 Flowchart representing object detection implementation . . . . .	31
5.4.1 Class-wise IoU scores for each Model (Lighting) . . . . .	35
5.4.2 Class-wise IoU scores for each Model (Weather) . . . . .	36

# List of Tables

5.1 Mean PSNR and SSIM for augmented and normalized images . . . . .	30
5.2 Average scores for lighting condition dataset . . . . .	34
5.3 Average scores for weather condition dataset . . . . .	34
5.4 Lighting condition dataset: Average class-wise IoU@[0.5:0.95] . . . . .	35
5.5 Weather condition dataset: Average class-wise IoU@[0.5:0.95] . . . . .	36

# List of Equations

4.0 Fast R-CNN loss . . . . .	19
5.0 Generator Loss . . . . .	25
5.0 Discriminator Loss . . . . .	25
5.0 PSNR . . . . .	29
5.0 SSIM . . . . .	29
5.0 SSIM . . . . .	29
5.0 IoU . . . . .	32
5.0 Precision . . . . .	32
5.0 Recall . . . . .	33
5.0 F1 . . . . .	33
5.0 mAP . . . . .	33

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

*AP* Average Precision

*cls* box-classification layer

*CNN* Convolutional Neural Network

*COCO* Common Objects in Context

*FC* fully-connected

*FCN* fully convolutional network

*FPN* Feature Pyramid Network

*GAN* Generative Adversarial Networks

*GDL* Gradient Difference Loss

*IoU* Intersection-over-Union

*mAP* Mean Average Precision

*MSE* mean squared error

*OpenCV* Open Source Computer Vision

*PSNR* Peak Signal to Noise Ratio

*reg* box-regression layer

*ReLU* Rectified Linear Unit

*RoI* Region of Interest

*RPN* Regional Proposal Network

*SSIM* Structural Similarity Index

*VAE* Variational Auto-Encoders

# Chapter 1

## Introduction

Identifying and localizing things in images or videos is the task of object detection- a crucial component in many computer vision applications. Traditionally, supervised learning with hand-crafted features has been used to perform this task, necessitating training on large labeled datasets [1]. However, in complex settings, this strategy can be less successful and labor-intensive. Hence, deep learning techniques are becoming more and more popular as a way to improve object detection.

An essential function in applications such as driverless vehicles is object detection. It supports navigation decisions by assisting these vehicles with detecting and tracking nearby objects. The wide variety of items that can be encountered and the numerous variables that can change an object's outlook in an image present object detection with a challenge. Autonomous vehicles generally use a variety of sensors, such as radar, LiDAR, thermal and optical cameras, and ultrasonic sensors, to detect objects. However, visual cameras are especially popular among them because of their affordability and capacity to record fine-grained visual information. However, depending just on color data from these cameras can be inadequate, particularly in difficult situations like dim illumination or shifting weather [5]. The accuracy of detection can be hampered by a variety of factors, such as differences in weather conditions like rain, snow, or fog [3, 4], as well as changes in lighting, viewing angles, and obstructions [2].

The ability to perceive and understand colors is known as advanced color vision, and it is essential for object detection. Object detection could be improved by combining deep learning methods like Generative Adversarial Networks (GANs) with sophisticated color vision algorithms. These techniques make full use of the range of color information available, enabling the extraction of more powerful and distinguishing characteristics. Therefore, through the utilization of the entire color spectrum, sophisticated color vision techniques may provide the best possible answers for intricate detecting situations. Thus, the main goal of this research is to investigate how enhanced color vision combined with GANs may be used to detect objects in autonomous cars. The effectiveness of this approach in comparison to conventional object detection algorithms is another goal of the study.

## 1.1 Research Objectives

This research aims to study the perception system in autonomous vehicles and to determine whether or not normalizing images with GAN can help improve its accuracy. Images from the surroundings of autonomous vehicles are picked up as color images by the camera. Machine vision is an important concept to understand as human vision and machine vision work differently when processing visual data. Thus, understanding the working principles of color vision in machines is significant in this research. Additionally, understanding normalization and the process of normalizing an input image, how image classification, extraction of the necessary information from images, and how various object detection models work are some of the important things this research focuses on. Also, understanding GANs and how to incorporate them into the object detection model is essential. Thus, the objectives can be summarized as follows-

1. Understand the applications of advanced color vision in autonomous vehicles or machine vision system(s).
2. Understand the effective application of generative adversarial networks (GANs), its types, implementations, and efficiency.
3. Demonstrate efficient object detection for autonomous vehicles after normalization.
4. Analyze and evaluate the performance of the used models.
5. Providing suggestions for further improvements and open questions (if any).

# Chapter 2

## Problem Statement

With the use of autonomous vehicles becoming increasingly prevalent in modern society, it raises the question of its reliability in recent times. How to judge the safety of self-driven automobiles, if they are better performing than human-driven vehicles, and whether or not they are capable of reducing road hazards are all social concerns [9]. These concerns may be mitigated by developing object detection systems that yield better results. To procure better-performing perception systems in autonomous vehicles it is imperative to develop an object detection system that is accurate, robust, and well-performing in real-time. Since object detection is an advancing field of research, several solutions have already been theorized, developed, and implemented for its miscellaneous applications. Hence, this paper is concerned with overcoming the challenges of implementing an object detection model that can improve the mentioned factor(s).

Developing a perception system for autonomous vehicles comes with challenges regarding data preparation, fusion methodology, and more [7].

A deep neural network model for autonomous vehicles requires large and diverse training data of high quality. Thus, using large multimodal data sets collected from different driving conditions, sensors, and object labels can improve the accuracy of detection. However, such tasks of collecting data are expensive and time-consuming. Additionally, the size of open multimodal data sets is significantly smaller than the size of image data sets. Furthermore, training data for deep neural networks are often labeled by humans which are susceptible to human errors [7]. Research [11], investigated the relationship between the quality of labels and the performance of convolutional neural networks (CNNs) which suggests that data size and coarse labels influence the training time of CNN, while [12] suggests that erroneous labels can impact the accuracy of CNNs.

As mentioned earlier, autonomous vehicles are equipped with multiple automotive sensors as the primary source of raw data, each of which captures its specialized data. Each sensor comprises modeling uncertainties that are difficult to quantify [7] but could heavily impact the reliability of autonomous vehicles under extreme scenarios. For example, harsh weather conditions, defective sensors, or unprecedented road situations are likely to cause higher modal uncertainty. Thus, the perception system must be robust enough to perceive the best outcome under any given condition. Recent studies [10], claim that Variational Auto-Encoders (VAE) and Generative

Adversarial Networks could be used to solve the multiple data modalities problem. Another paper [7] also suggests that these are rather popular methods used for image analysis and that GAN have recently been introduced to model Radar data for roadside detection in autonomous vehicles.

Therefore, the question this research is trying to address is:

**How effective Generative Adversarial Networks will be in improving the accuracy of object detection algorithms?**

There are various methods of solution for object detection systems for autonomous vehicles. Generally, either generative or discriminative models are used [6].

Generative models are usually used to strengthen the network in the system. Such models can generate newer instances of data that vary from one another and the data can be distributed without any supervision. On the other hand, discriminative models estimate the probability of an instance being part of a class and are done by analyzing large amounts of data from the class itself and its background [6]. These models learn to find the differences between given data and could be used to classify various objects that the vehicle sees, so it works more like a classifier. However, discriminative models do not tackle the issue of detecting the same object under different conditions of light and weather. Additionally, it would require a lot of labeled data so training such models would be time-consuming.

GANs, which are a type of generative model, are more suited to handling problems that arise with multi-modal perception systems. As stated previously, it is difficult to collect labeled real-time data, but GANs can easily generate newer instances of such labeled data using sensors. Since GANs can reconstruct the same image, even if an autonomous vehicle is driving across a scenario with bad environmental conditions such as rain or fog, the model can easily use the values it was trained on to generate a clear instance of the original by removing the obstructions. As a result, autonomous vehicles can continue driving without any complications. Besides, it is also useful in cases when sensors are not responding or have failed [7], reducing the uncertainty of sensor modalities. Thus, GAN can be used for image normalization.

Hence, this research will attempt to use GAN to normalize the data to a clear image before passing it through an image classifier to classify objects for detection.

The impact of object detection systems in autonomous vehicles could be enormous. Undoubtedly, its main objective is to reduce road accidents in various light and weather conditions or in real-time [7]. Besides, advanced object detection systems would ensure that autonomous vehicles are far safer and more reliable. Although societal acceptance and incorporation of such technology may not come as easily, however, with time autonomous vehicles with reliable object detection systems will be the stepping stones of robotics in human society. It will ensure optimal fuel consumption due to better decision-making abilities, can reduce carbon emissions lowering levels of greenhouse gasses, and noise pollution would also be reduced. Additionally, users and governments would benefit as well, as they would be able to ensure reduced road accidents or casualties, and speed limits could be monitored, which may also help enforce traffic rules and regulations. One other major benefit of



autonomous vehicles with advanced object detection systems would be the elimination of drunk driving, texting, speeding, or distracted driving and related casualties [8]. Hence, a model that tackles the problem of various light conditions and weather conditions will be able to ensure a system that is completely ready for the road.

# Chapter 3

## Literature Review

Up until this point, several methodologies and theories have been developed to explore object detection and its uses in its miscellaneous applications.

This section aims to critically analyze previous works that relate to color vision approaches for object detection. Additionally, it looks into various discriminative and generative methods of object detection that could also be used in autonomous vehicles.

Two recent studies [13] and [14] use autoencoder techniques to produce normalized forms of images under various conditions. The first work deals with weather conditions, and the second under different lighting conditions, but ultimately they use a similar methodology for image reconstruction. The autoencoding technique has three parts: the encoder, the bottleneck, and the decoder. The encoder compresses the image with unwanted features by reducing its dimensions through multiple CNN layers. The neuron count is reduced in each layer producing a bottleneck, where image compression is at its maximum. In the decoding layers, the compressed image will be decompressed, by increasing neurons in each layer. And the image is restored back to its original size, in a normalized form. This output image will be clearer as it removes noises from images with poor illumination or weather. The model works in both papers and has a high degree of accuracy. The issue with this approach is that bottlenecking and lossy output narrow down neurons, which can cause a loss in data. Additionally, the system may become too specialized and only work for certain types of images. Hence, the system would not be robust enough to work in multiple different weather or lighting conditions.

Research work [15] approaches color image analysis with an intrinsic reflection model. This system is based on a dichromatic reflection model. It starts by making a hypothesis on matte color clusters and exploits them for image segmentation. These matte hypotheses are extended to make hypotheses on skewed T's, which are once again segmented and exploited to separate all pixels into their reflection components. The output shows that the process can find the shape of the material without the external lighting effects. However, this is an outdated model, and will not work if the T-shape degrades into a line. This can happen when it lacks sufficient illumination. Metallic objects and rough surfaces do not give satisfactory results that can be analyzed properly.

Another research work [16] attempts to use non-parametric classification and multivariate decision trees for real-time color recognition. The non-parametric classification has two phases: a training phase to approximate a function representative of a distribution from a sample and a classification phase that determines the class of each pixel in the function. The multivariate decision tree is used to create a piecewise approximation of areas in feature space through the recursive division of feature space with hyperplanes to create instances. The instances are further processed and a decision tree is produced. The output gives a binary image where the suspected target pixels are white and the background is black. The model gives a lot of false positives as output, so it cannot be successfully used for object detection. Additionally, like all decision trees, it is prone to over-training. Furthermore, the system is very outdated compared to more modern algorithms and machine learning models present today.

A research paper [17] uses GANs to color-correct images taken underwater, which have been altered by changes in depth and illumination. CycleGAN is used to generate distorted images from a set of undistorted ones, which are false instances. These images are mixed in with a set of real-life distorted pictures, or true instances, and are fed into a discriminator, which attempts to distinguish whether the picture is a true instance or a false one. The aim is to train the generator well enough to be able to fool the discriminator into classifying the false instances as true. However, this can lead to a vanishing gradient or a mode collapse. A vanishing gradient is when the discriminator improves too much, it cannot be fooled anymore, and the system can not be trained. A mode collapse is when the GAN only produces a limited variety of outputs that will always fool the discriminator and do not learn to diversify. To overcome the vanishing gradient problem, a Wasserstein GAN is formulated, and combining this with the original system gives the completed underwater GAN or UGAN. As the CycleGAN images might be blurry, it can be optimized using a Gradient Difference Loss (GDL). The UGAN with the GDL considered is denoted as the UGAN-P. Both the UGAN and UGAN-P were trained, and both models have a high level of accuracy. They could recover lost color information while ignoring the images that do not need to be corrected. However, in its current form, it is too specialized for color correction only, and the CycleGAN is not able to produce a diverse set of images with other external factors like particle or lighting effects. Moreover, it does not provide any solutions if mode collapse occurs.

From the above discussion, it can be seen that every paper has approached its implementation of color vision and/or object detection in an array of methods. Each technique is unique and has its own advantages and disadvantages, and they all aim to solve a different problem while still having the same core concept. Likewise, this research will also attempt to detect objects with a new technique that is yet to be implemented. However, it will not limit itself to just image normalization or image classification separately.

# Chapter 4

## Methodology

### 4.1 Proposed methodology

Theoretically, this research may be divided into two major stages: image normalization and image classification. The image normalization model will attempt to reconstruct a normalized version of the images in various lighting and weather conditions while the classifier model will attempt to classify objects in those reconstructed images. The figure below illustrates the proposed methodology for this research:

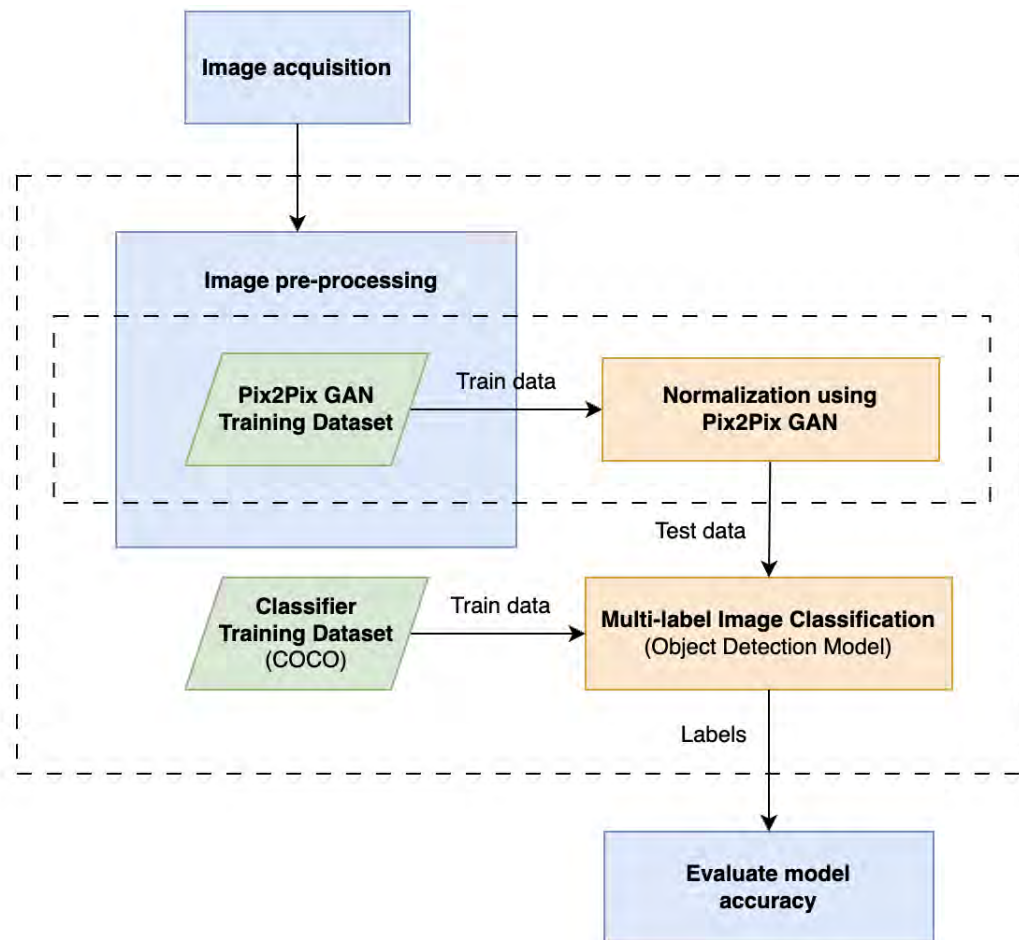


Figure 4.1.1: Methodology Flowchart

This research begins with acquiring high-resolution images of various driving scenes

from cameras as raw data. The raw images require pre-processing due to the numerous conditions of this research and the requirements of the research methodology.

Pix2Pix is a type of GAN that learns to map between an input image and an output image (target). To train the Pix2pix GAN, this model requires two types of images: clean images as the target, and images in various lighting or weather conditions in the same scenes. Since the original dataset can not fulfill such a requirement, nor would it be possible to capture images in the same driving scene with the same setting at various light/weather conditions, the original images need to be augmented. Using image augmentation techniques, the images can include variations of different lighting conditions such as bright, sunny, and dark; and different weather conditions such as rainy, and foggy. This may be done using image manipulation techniques such as brightness and contrast adjustments, color channel modifications, adding synthetic weather elements, and more.

During the pre-processing stage, this research will attempt to create two types of datasets. One dataset was needed for the normalization of images in varying lighting conditions. The other was needed to normalize images in varying weather conditions. Lastly, a large open-source dataset specialized for object detection in autonomous vehicles was used to train the classification model for multi-label classification.

The next step for this model is image normalization where the input images would be reconstructed by the GAN model to produce a normalized image for the noisy input. Although there are multiple types of GAN, for this research Pix2Pix was determined to be most suitable to normalize images in varying weather or light conditions. For this research, the input images of Pix2Pix would be the augmented driving scene images, and the output images would be the normalized versions of those images.

The last stage of this model will use multi-label classification. For this section of the methodology, this research implemented three different object detection models. They are: Faster R-CNN with a ResNet50 backbone, which has high accuracy but is slow; Faster R-CNN with a MobileNetV3 backbone, which is faster but with lower accuracy; and RetinaNet with a ResNet50 backbone which has a good balance between speed and accuracy. All three classifier models were pre-trained, and are used in object detection so that they can localize objects in an image, similar to the perception system in an autonomous vehicle. The aim of the object detector is to determine whether or not objects can be detected more accurately using normalized images compared to images that are not normalized under various road conditions, thus showing the effectiveness of the Pix2Pix GAN. Finally, this research aims to compare and analyze the classification result accuracy of images with and without normalization for all 3 models.

## 4.2 Data analysis

For this research, two datasets had to be prepared which included images of roads in a variety of weather and lighting conditions. Due to the nature of the model that was to be trained, the images had to be augmented to meet its requirements. This

section analyzes the acquisition, pre-processing, and the contents of the dataset.

### 4.2.1 Data acquisition

In order to acquire a dataset for the model, initially images were scoured from self-sourced dashcam footage and open-source datasets. However, due to the lack of variation in the season during which the dashcam footage was sourced, most of the images sourced did not meet the requirements. Additionally, they were of low resolution. Hence, open-source footage with higher-resolution images was preferred. However, it does not completely fit the required criteria for training the GAN model. This is because training a Pix2Pix GAN model requires a pair of images such that one is an input image must have a variation of weather or lighting conditions while the other is the target as it learns to map between them. Therefore both images must contain the same scene and objects. Hence, the clear and normal images from the open-source dataset were extracted, which gave around 10,000 clean images in total. To complete the dataset for the Pix2Pix GAN model, these clean images were used to produce the necessary augmented images with the required weather and lighting conditions.

### 4.2.2 Image preprocessing

After collecting all the clear data for the Pix2Pix model, all the images were first resized to 256 x 256 pixels. This was done due to hardware limitations. As described by [18], when creating a dataset for Pix2Pix GAN, it is necessary to add random jitters and mirroring to preprocess the training data set. However, for larger datasets jittering was not added. In this case, since the images will be augmented and as the dataset is large, no jitters were added. However, some of the images from the cleaned original dataset were mirrored before augmentation.

### Augmentation

To augment the clear images, Python's OpenCV (Open Source Computer Vision) and ImgAug libraries were used extensively. The OpenCV library is made up of programming functions intended to be used for real-time computer vision programs, while ImgAug is a powerful library for image augmentation in machine-learning experiments.

Initially, the images had to be converted from BGR (Blue, Green, Red) color format to RGB (Red, Green, Blue) color format using the `cvtColor()` function. This was done because OpenCV's `imread()` function loads images in BGR format by default which is not compatible with the ImgAug library, which uses RGB by default.

### Bright Light

To augment daytime images, the Multiply augmenter was created using the ImgAug library for brightness contrast. Three brightness multipliers with brightness coefficients of 1.5, 2.3, and 3 were defined to increase the brightness of the image. For each multiplier, the brightness augmenter was applied to create a new image with adjusted brightness. This helped to create images at different light intensities of

daylight.

### **Low Light**

Night-time image augmentation did not yield reliable results to be included in the datasets. Instead, this research aimed to create darker or low-light images. In order to achieve this, the brightness, saturation, and contrast were first adjusted using the PIL ImageEnhance module. By lowering the brightness, the entire image was made darker. In the next step, the saturation of the image was decreased between 0.9 to 0.6, making the colors less vivid and appear closer to shades of gray. This can create a more subdued or monochromatic look. Next, the contrast was increased to slightly sharpen the images. Then the image is converted from RGB color space to BGR color space so that the cv2 library can be used. Finally, the gamma correction technique allowed this research to adjust the luminance of the images. The images were adjusted using the cv2.LUT() function. If the gamma value is below 1, it produces a darker low-light image. If increased above 1, it increases image brightness. A gamma value of 0.5 to 0.7 was determined to produce the most acceptable low-light images.

### **Rain**

To artificially create rainy images, the Automold Road Augmentation Library [19] was utilized to augment clear daytime images to add synthetic rain. This library was designed to augment road images into various lighting and weather conditions that can pose challenges for training neural network models. First, the OpenCV library was used to process the images. Then the custom modules, Automold, and Helpers, present in the Automold library were used to add rainy effects. Automold was employed to add the rain effect to the images, while Helpers was used to load the images from the directory. Each image from the initial 10,000 clear images was taken, and a rain effect was applied with raindrops falling at random angles in each image. Additionally, this dataset includes rainy images augmented at three different levels of rain effects: drizzle, heavy, and torrential. Where the level of rain increases from little to very heavy rain. The type of rain is specified as a parameter to the am.add\_rain() function.

### **Fog**

Lastly, this research also created synthetic foggy images of various driving scenarios. The images were first converted to low-light images, to reduce the sunny or bright sky effects present in the original images. Now, creating artificial fog in various driving conditions was more challenging than other augmentation, as it requires depth estimation to produce realistic fog effects. Meaning, that the intensity of fog has to be higher at a distance away from the camera position compared to the view at a closer distance. Hence, the Monodepth2 [24] was employed for depth estimation. It is a Self-Supervised Monocular Depth estimation technique that has several modules for depth prediction. For this research, the mono\_640x192 was used which fits the image resolution range for this dataset. This module was used to generate a depth map which was then converted to a grayscale image. This grayscale was then normalized to a value between 0 and 1 using OpenCV's cv2.normalize func-

tion. The normalized depth map is then inverted ( $1 - \text{depth\_map}$ ), which results in larger values for pixels that are farther away from the camera and smaller values for pixels that are closer. Before adding fog, the depth map loaded as a grayscale image (single channel) is converted to a 3-color channel image using the `cv2.cvtColor` function. Then the depth map values are multiplied by a fog intensity factor which increases the contrast of the depth map, effectively determining the intensity of the fog. The fog effect is applied by taking a weighted sum of the original image and the fog color, where the weights are determined by the depth map. The operation  $(1 - \text{depth\_map\_3ch}) * \text{image} + \text{depth\_map\_3ch} * \text{fog\_color}$  results in pixels of the original image being blended with the fog color. Pixels corresponding to objects closer to the camera (small depth map values) retain more of their original color, while pixels corresponding to objects further away from the camera (large depth map values) take on more of the fog color.

After applying the adjustments above, each of the images was converted back to BGR form to become compatible with OpenCV's `imwrite()` function which was used to save the images. Some of the results of augmentation are shown as follows:

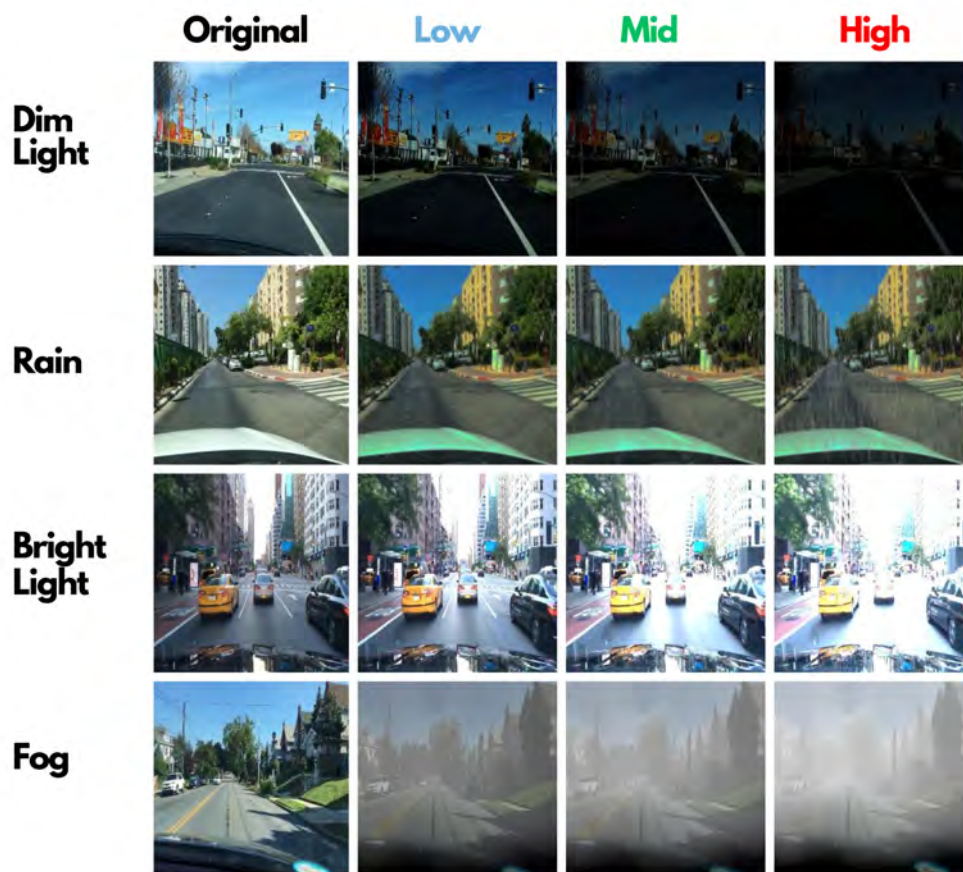


Figure 4.2.1: Augmentation results after pre-processing

Once the augmentation for each image was complete, the original image (target) and the augmented image (input) were combined and saved to form a single image using the Numpy library. The `np.concatenate()` function was used to achieve this. This resulted in a new  $512 \times 256$ -pixel image. Additionally, all the augmentation



images were saved separately, as they would be required to test the classifier model.

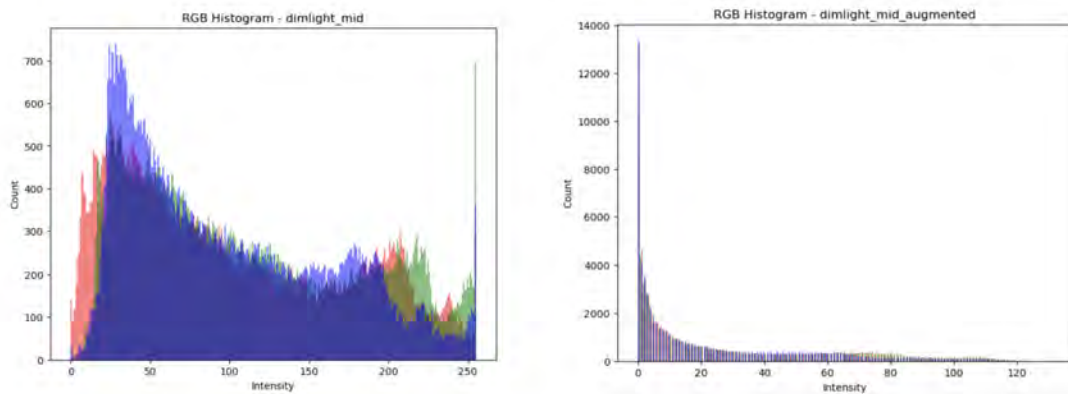
Finally, the acquired dataset was split data into train, test, and validation in the ratio 80:10:10. Therefore, the training dataset for each model would have about 48,000 images. While it would be tested with 6,000 images. When testing the data with the object detection models to measure the improvement of accuracy before and after normalization, several test datasets had to be created. Three tests were made, consisting of augmented images, predicted images and ground truth gathered from both lighting condition and weather condition datasets.

### 4.2.3 Augmented Data Analysis

Finally, 2 different datasets were procured from the aforementioned augmentation. Each dataset consisted of 60,000 images consisting of 10,000 images per variation. That is, the weather condition dataset had 30,000 images for 3 variations of rain (drizzle, heavy, torrential) and another 30,000 included the 3 variations of fog. The lighting condition data set, on the other hand, has 30,000 images for the 3 variations of darker or low light intensity images and another 30,000 for the 3 variations of brighter images. Therefore, both datasets had a uniform distribution of weather or lighting variations respectively.

The following section shows figures of the RGB histograms of some selected augmentations along with their analysis, to differentiate between the original and the augmented images.

#### 1. Low light



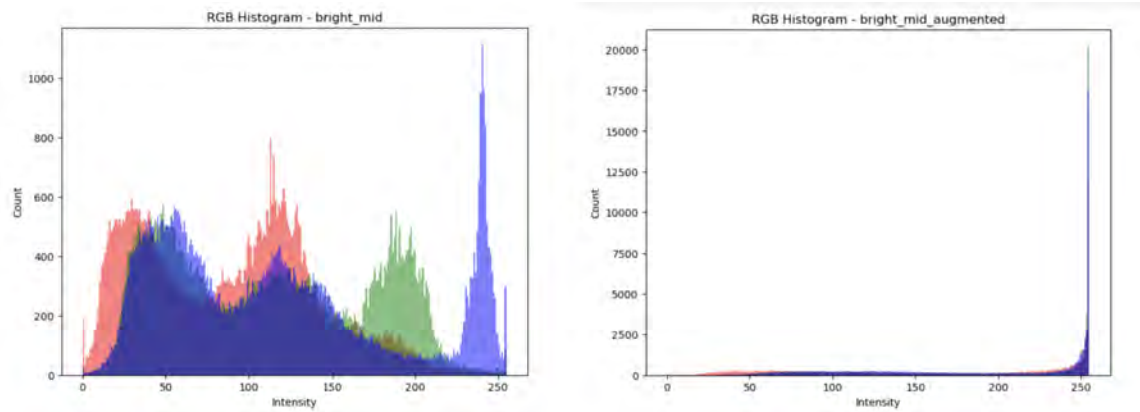
(a) RGB histogram of original image (b) RGB histogram after Dim light augmentation

Figure 4.2.2: Original vs augmented for Low Light (level 2)

After augmenting the images to low lighting conditions, the RGB color channels in Figure 4.2.2 (b) have turned into separable bars, and the color channels lean towards the left. The separable bar suggests that certain intensities have been removed from the RGB color channels selectively. Additionally, the intensities of the pixels have decreased drastically towards the minimum value 0 for all three channels. Hence, most of the pixel values are near (0, 0, 0) which tends to be black. As the intensities of the pixels are scaled down, the overall contrast in the image is decreased. This

makes the darker regions look even darker, and the range of intensity values becomes compressed leading to a significant loss of color information.

## 2. Bright light

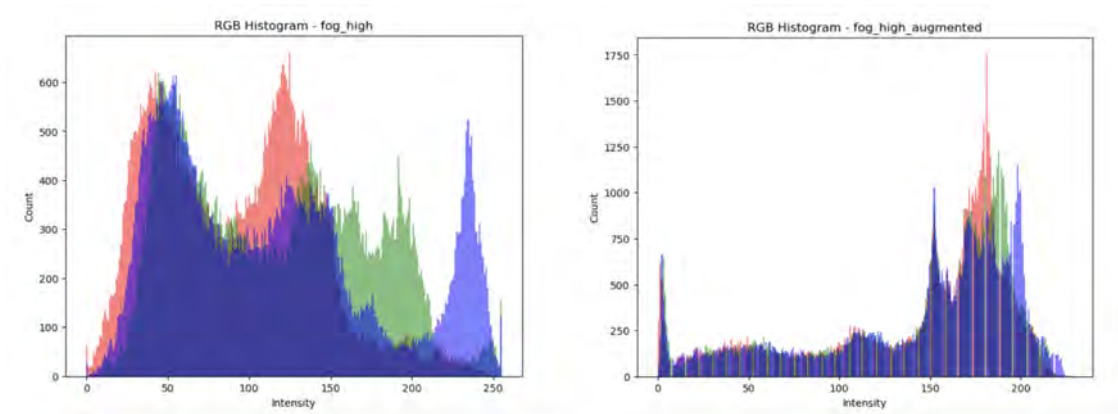


(a) RGB histogram of original image (b) RGB histogram after Bright light augmentation

Figure 4.2.3: Original vs augmented for Bright Light (level 2)

The addition of brightness has caused the intensity of Red, Green, and Blue color channels to increase significantly in Figure 4.2.3 (b) compared to Figure 4.2.3 (a), which has led to almost all of the color being washed out in the image. As the intensity of all the color channels is scaled up to a maximum value of 255, the RGB value of most of the pixels has become (255, 255, 255) which is suggested by the spiking count of nearly 20,000 pixels at an intensity of 255. This means that most of the pixel color tends to be white. The augmented histogram suggests that there has been a drastic loss of color information and details as it is now more difficult to differentiate color information in the augmented image. This is because, as pixels become saturated and approach the upper limit, the subtle differences in color variation are lost.

## 3. Fog



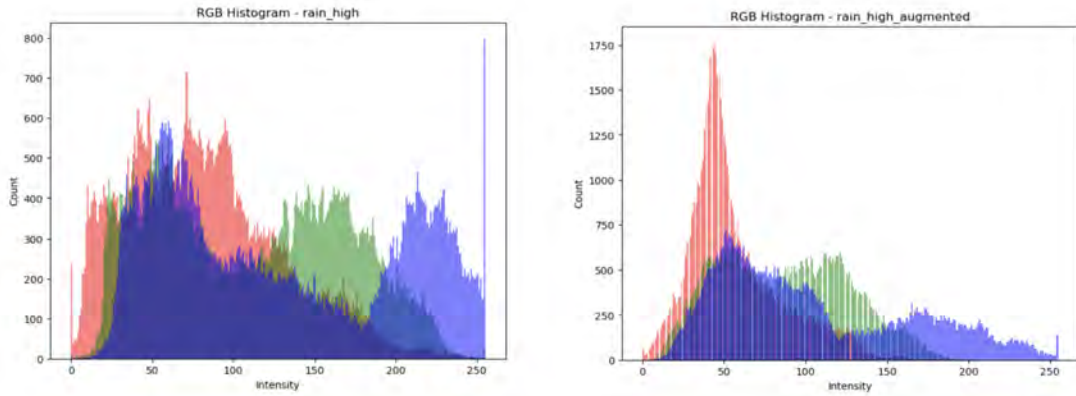
(a) RGB histogram of original image (b) RGB histogram after Fog augmentation

Figure 4.2.4: Original vs augmented for Fog (level 3)

After adding the fog artificially, all 3 color channels have shifted towards the right

Figure 4.2.4 (b). This means that the color intensities have increased. This could be due to the addition of artificial whitish fog making the pixel colors tend towards white. Additionally, the disappearance of lines within the Red, Green, and Blue channels individually suggests that the augmentation has affected the channels individually.

#### 4. Rain



(a) RGB histogram of original image

(b) RGB histogram after Rain augmentation

Figure 4.2.5: Original vs augmented for Heavy Rain

In these last set of histograms some disturbance can be seen in Figure 4.2.5 (b) as there has been a shift in color balance towards the left for all 3 color channels when compared to the original colors in Figure 4.2.5 (a). This suggests that the color intensity of all 3 channels has decreased. Thus the contrast has also decreased. Additionally, there are some disappearing lines at certain intensities in each channel individually. This could be the effect of individual channel transformations since the white areas in the blue channel do not result in the disappearance of values in the Green or Red channels at the same point and vice versa.

Although the loss of color information in the fog and rain-augmented images is not as significant as the lighting condition augmentations, it will, however, provide a challenge for the GAN to identify the exact color information.

### 4.3 Working principle of Pix2Pix GAN

The Pix2Pix model is a type of conditional GAN that learns to map between an input image and an output image using a pair of input images. In this case, the input pair would be a noisy image of various driving conditions and a clear target image in the same driving scene.

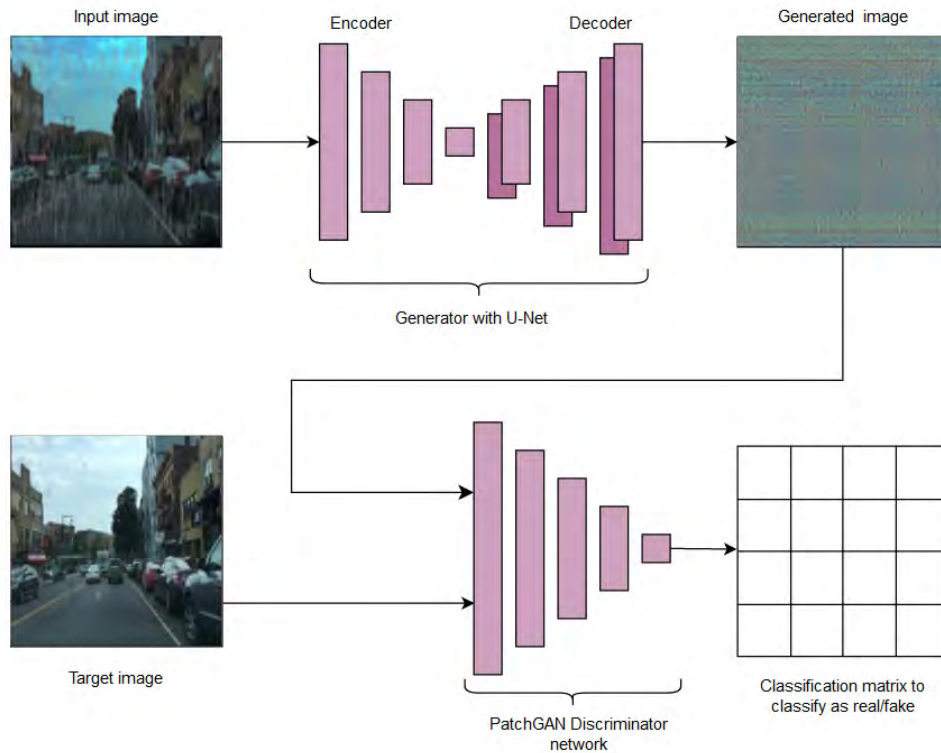


Figure 4.3.1: Working principle of Pix2Pix GAN

The working principle of Pix2Pix GAN can be divided into two main components, the Generator and Discriminator, which are described as follows:

1. Generator

The generator takes an input image and reconstructs an output image that is similar to a target image. The generator is a modified U-net that contains an encoder-decoder architecture, where the encoder is made up of a series of convolutional layers that take an image as an input, and down-samples it to extract features. The decoder, on the other hand, is a series of transposed convolutional layers that take these features and up-samples them to generate the output image. The encoder and decoder are based on CNNs. Together they attempt to generate new images for the input that would be similar to the target image.

2. Discriminator

The discriminator is a separate network that attempts to distinguish between real and generated images. It is a convolutional PatchGAN classifier, which attempts to classify the patches of images as real or fake [18]. Each block in the discriminator is a series of convolutional layers that down-sample the

input image and extract features. It takes in a pair of inputs which include the target image (real) and the generator-generated image (fake) and attempts to classify them as real or fake.

Once the Pix2Pix GAN is trained, it can be used to generate realistic output images for a given input image. The generated images can be used in a range of applications, such as image-to-image translation, image style transfer, and image inpainting. Overall, the Pix2Pix GAN working principle involves training a generator and discriminator network to work together to generate outputs that are similar to the target images.

## 4.4 Description of the Object Detection Models

### 4.4.1 Faster R-CNN using ResNet50

The Faster R-CNN model is a deep convolutional network used for object detection, and it works as one unified network. It can be used to accurately predict the locations of different objects in an image and classify them. Since it is a two-stage detector, it is made up of two modules: the RPN and the Fast R-CNN network, which share the same convolutional layers.

Module 1: RPN (Regional Proposal Network)

The RPN is used to form region proposals, which are bounding boxes in the image that mark the most likely position of images. To start the training, the objects to be detected in the image are labeled, and these areas are called ground truths. Next, the image is run through a fully convolutional network. For this model, it has been run through a ResNet50 backbone, which is a residual network composed of 50 layers [25].

The first layer has a 7x7 convolution with 64 filters. The following 48 layers are grouped into blocks of three layers, where each block is made up of 1x1, 3x3, and 1x1 convolutions. The 1x1 convolutions are used to decrease and then increase the dimensions so that the 3x3 works as a bottleneck. The blocks have shortcut connections between them, which reduces degradation, a common problem seen in deeper networks. The last layer is a global average pooling layer and a 1000-way fully-connected layer.

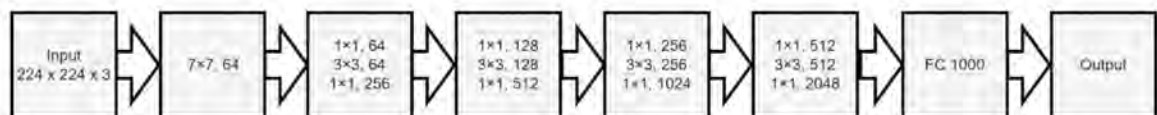


Figure 4.4.1: ResNet 50 architecture

After it has been passed through the network, it becomes downsampled along the spatial dimension. The output is a feature-rich representation of the image. Each point on the feature map is considered to be an anchor point. The anchor points are used to create anchor boxes, whose sizes and numbers depend on the number of region proposals to be made.

For each region proposal, a feature vector is extracted. This vector is then fed to 2 fully-connected (FC) layers known as a box-regression layer (reg) and box-classification layer (cls). The reg layer returns a 4-D vector defining the bounding box of the region, and the class layer, and the cls layer represents a binary classifier that generates the objectness score for each region proposal.

To generate the region proposals, the area of intersection between every anchor box and ground truth is checked. This area is known as the Intersection-over-Union or IoU. The IoU ranges from 0.0 to 1.0, and the more intersection there is, the higher the objectness score for each anchor. If the IoU level is above 0.7, then the anchor box is given a positive objectness label, so it is classified as an object, if IoU is less than 0.3 it is given a negative objectness label, and is classified as the background instead [22]. Anchor boxes with values between 0.3 and 0.7 are not used in training.

To adjust the anchors to fit the ground truths better, a 1x1 convolutional network is trained to predict the offsets from the ground truth boxes [23]. These predicted offsets are used to transform the anchor boxes to the desired regions, and these give the final region proposals that are used in the next step.

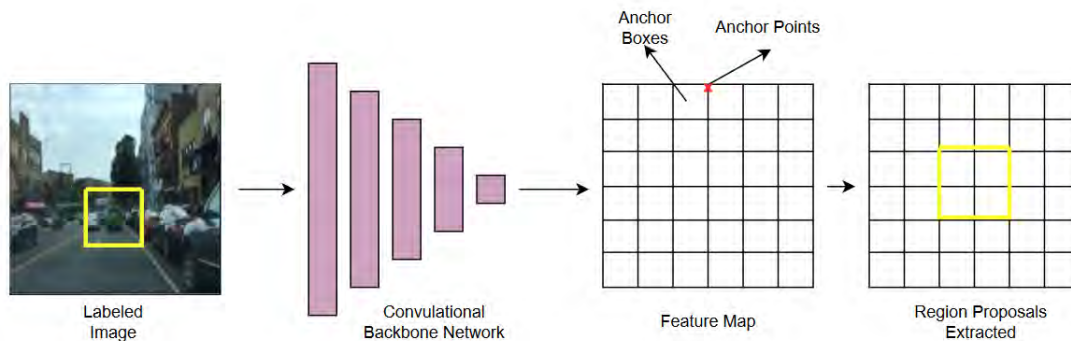


Figure 4.4.2: Stage 1 of Faster R-CNN

## Module 2: Fast R-CNN network

The Fast R-CNN network is trained to identify the objects using the region proposal that was produced in the first section. In this stage, this system learns to categorize the object of the region proposal by using a basic convolutional network. Region of Interest (RoI) Pooling is used to change the sizes of the regional proposals as they are not equal. Then, they are passed through a network, which learns to predict various categories using cross-entropy loss. A second network learns to guess the offsets between region proposals and ground truth boxes while trying to align them using L2 regression loss. Lastly, it computes the actual loss by taking a weighted combination of both losses. The formula used it:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad (1)$$

Where,  $i$  is the anchor's index, with  $p_i$  being the probability of it being an object.  $t_i$  and  $t_i^*$  are vectors for the 4 parameterized coordinates of the predicted bounded area, and the ground-truth boxes respectively. The classification loss  $L_{cls}$  is log loss over whether a class is an object or not. To ensure that regression loss will only be calculated if an anchor is positive, the term  $p_i^*$  is multiplied with the loss formula.  $\{p_i\}$  and  $\{t_i\}$  provide the outputs of the *cls* and *reg* layers respectively. These are normalized using  $N_{cls}$  and  $N_{reg}$ , which were equalized using the  $\lambda$ .

The image is passed through the initial RPN layer again to generate anchor boxes once again, sent back to the Fast R-CNN for RoI pooling and loss calculation. This process is repeated several times to improve the region proposals and increase the object detection accuracy.

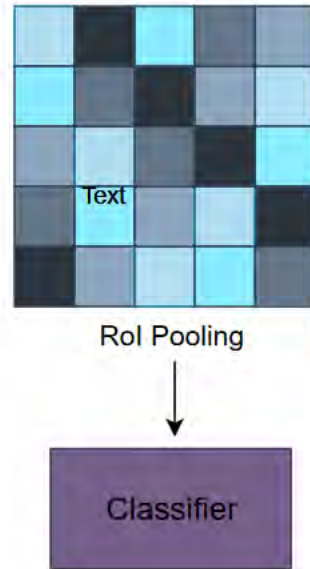


Figure 4.4.3: Stage 2 of Faster R-CNN

#### 4.4.2 Faster R-CNN using MobileNet v3

The second model works similarly to the first. The only difference is that it uses a MobileNetV3 backbone instead of ResNet50 in module 1. The MobileNetV3 is a convolutional neural network that has been built upon its predecessors in the MobileNet series [26].

It begins with a 2D convolutional layer before moving onto the blocks. The blocks in MobileNetV3 have a linear bottleneck and inverted residual structure, which was originally introduced in V2 and is resource-efficient. This structure is composed of a 1x1 expansion convolution followed by depth-wise convolutions and a 1x1 projection layer. The bottlenecks are either of size 3x3 or 5x5. Squeeze and excitation modules

have been integrated into the bottleneck structure for reducing parameters, and the layers have modified swish nonlinearities implemented into them to improve speed. These blocks lead off to a  $1 \times 1$  2D convolutional layer, a  $7 \times 7$  pooling layer, and finally two  $1 \times 1$  2D convolutional layers without batch normalization.

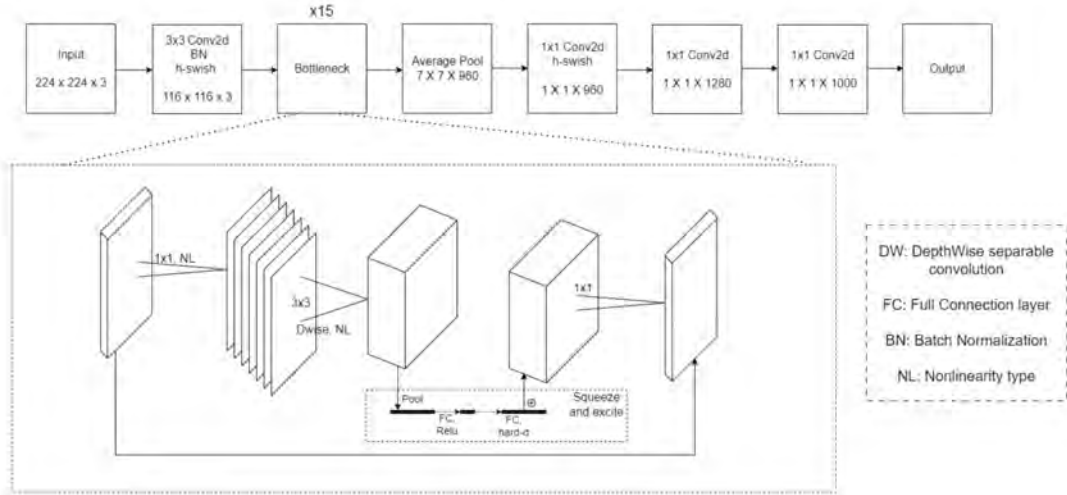


Figure 4.4.4: MobileNetV3 architecture

### 4.4.3 RetinaNet with ResNet50

RetinaNet is a dense, one-stage object detector. It is a single, unified network composed of a backbone network and two task-specific subnetworks [27].

The backbone network is used to compute a convolutional feature map over an entire input image. The Feature Pyramid Network (FPN) is used as the backbone, which augments a standard convolutional network with a top-down pathway and lateral connections. This allows the network to efficiently construct a rich, multi-scale feature pyramid from the given input image. Each level of the pyramid can detect objects at a different scale. This FPN is built upon the feedforward ResNet-50 structure, which is used to construct a pyramid with levels from  $P_3$  to  $P_7$ , where all pyramid levels have  $C = 256$  channels.

Translation-invariant anchor boxes are used on pyramid levels  $P_3$  to  $P_7$ , with areas of  $32^2$  to  $512^2$ . In total, there are  $A = 9$  anchors, and each anchor is assigned a length  $K$  one-hot vector of classification targets, where  $K$  is the number of object classes and a 4-vector of box regression targets. The anchors are assigned to ground-truth object boxes using an IoU threshold of 0.5, and to the background if the IoU is less than 0.4. In the corresponding entry, the  $K$  label vector is set to 1, and all other entries to 0. Unassigned anchors, that have IoU between 0.4 and 0.5 are ignored during training. The overall backbone is finally linked to the two subnetworks.

The first subnet is the classification subnet, which is used to perform convolutional object classification on the backbone's output. This subnet is a small fully convo-



lutional network (FCN) attached to each FPN level, and it predicts the probability of object presence at each spatial position for each of the  $A$  anchors and  $K$  object classes. Taking an input feature map with  $C$  channels from a given pyramid level, the subnet applies four  $3 \times 3$  conv layers, each with  $C$  filters and each followed by ReLU activations, followed by a  $3 \times 3$  conv layer with  $KA$  filters. Finally, sigmoid activations are attached to output the  $KA$  binary predictions per spatial location.

The second subnet performs is the box regression subnet, convolutional bounding box regression. This is another small FCN attached at every pyramid level, and works in parallel to the classification subnet to regress the offset from each anchor box to a nearby ground-truth object if one exists. The box regression subnet is identical to the classification subnet except that it terminates in  $4A$  linear outputs per spatial location. For each of the  $A$  anchors per spatial location, these 4 outputs predict the relative offset between the anchor and the ground-truth box.

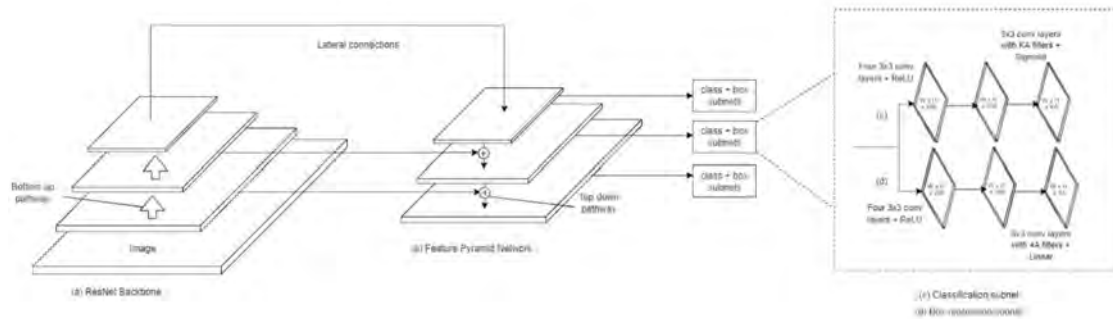


Figure 4.4.5: RetinaNet structure using ResNet50 backbone

# Chapter 5

## Implementation

### 5.1 Pix2Pix GAN Model implementation

Implementing the Pix2Pix model involves many libraries and functions. This section will briefly discuss how some of these important tools and parameters were used to implement it, along with the changes that were required to implement the GAN for the dataset this research created.

#### 5.1.1 Generator building

The U-Net architecture is commonly used as the generator in Pix2Pix GANs for image-to-image translation problems. It contains an encoder-decoder structure along with skip connections, enabling it to effectively capture both local and global features. This research used a generator that starts with an input layer, that takes input images with dimensions 256x256 pixels and a filter size of 3. Images of small dimensions were used due to hardware limitations and to reduce processing time. The input images were downsampled through 8 encoding layers to reduce its dimensions to 1x1 pixels and an output channel of 512, by the encoder.

The encoder is made up of 3 fundamental components described as follows:

1. Convolution: The encoder starts with a series of 2d convolutional layers. Each convolutional layer applies a set of learnable filters to the input image that can extract the features from the input. The number of filters determines the depth or the number of channels in the feature maps.
2. Batch Normalization: After each convolutional layer, batch normalization is applied. The Batch normalization is responsible for normalizing the activations of the previous layers, thus improving the stability and convergence of the model during the training process.
3. Leaky ReLU Activation: Leaky ReLU (Rectified Linear Unit) is a piecewise linear function, commonly used as the activation function after the batch normalization layer. It allows small negative values to propagate, preventing the issue of "dead" neurons and enabling the model to capture more diverse features.

At each downsampling step, the intermediate feature maps are stored in a skips list. These feature maps would be used later during upsampling for skip connections.

Next, the upsampling layers are used to increase the spatial dimensions to up to  $256 \times 256$  of the feature maps while decreasing the number of channels to 64. This process helps the model generate high-resolution output images from the learned features. The following section describes the decoder:

1. **Transposed Convolution:** The decoder starts with 2D transposed convolutional layers, also known as deconvolutional or upsampling layers. Transposed convolutional layers perform the opposite operation of convolution, which increases the spatial dimensions of the feature maps.
2. **Batch Normalization:** Similar to batch normalization in the encoder, it is also required in the decoder. It is applied after each transposed convolutional layer in the decoder to normalize the activations.
3. **Dropout:** is a regularization technique used to prevent overfitting. It is done by randomly setting a fraction of the input units to zero during training. It helps to reduce the co-adaptation of neurons and improves the model's ability to generalize.

The last layer is a 2D transposed convolutional layer with the defined output channel filters (which determine the number of channels in the output), a kernel size of  $4 \times 4$ , and tanh activation. This layer generates the final output image with the same spatial dimensions and filter size as the input image ( $256 \times 256 \times 3$ ).

Additionally, skip connections are a critical aspect of the U-Net architecture. They were added between the corresponding encoder and decoder layers. These connections allow the model to preserve high-resolution features from the encoder while combining them with the upsampled features from the decoder, enabling the model to capture both the local and the global information effectively. The U-Net architecture's skip connections and symmetrical structure aid in retaining fine-grained details while also allowing the model to generate coherent and clearer output images.

### 5.1.2 Discriminator building

The convolutional PatchGAN classifier is a key component of the discriminator in Pix2Pix GAN [21]. The discriminator is designed to classify the image patches of an input as real or fake [18].

It receives two input images of dimensions  $256 \times 256$  and a channel size of 3, which are the input image generated by the generator and the target image. The input images are concatenated along the channel dimension, resulting in a single image with doubled channels. This operation is done to provide the Discriminator with both the source and target images for assessing their relationship. The concatenated image has a shape of  $(\text{batch\_size}, 256, 256, \text{channels} \times 2)$ . Then the concatenated image goes through a series of downsampling layers till the filter size is increased to 256. This works similarly to the encoder as described below:

1. **Convolution:** The PatchGAN classifier applies a series of convolutional layers to extract features from local image patches. Each convolutional layer uses learnable filters to convolve over the input patches, capturing various visual characteristics and patterns of the images generated by the generator. The

number of filters determines the depth or number of channels in the feature maps.

2. **Batch Normalization:** is applied again after each convolutional layer. Similar to what is done in the encoder, it normalizes the activations of the previous layer, to improve the stability and convergence of the model during training. It helps in normalizing the output distribution and speeding up the training process of the discriminator.
3. **Leaky ReLU Activation:** Leaky ReLU is used again as the activation function after the batch normalization layer. It allows small negative values to propagate, preventing the issue of "dead" neurons and enabling the model to capture more diverse features. It introduces a small slope for negative values, typically 0.2, allowing some information flow even for negative inputs.

After the third downsampling layer, the feature map is zero-padded using `tf.keras.layers.ZeroPadding2D()`. This padding operation increases the spatial dimensions of the feature map. Now, a convolutional layer with 512 filters, a 4x4 kernel size, and a stride of 1 is applied to the padded feature map. The Batch normalization and LeakyReLU activation are applied again after this convolutional layer. Finally, the last convolutional layer with a single filter and a 4x4 kernel size is applied. This layer produces a single-channel output.

The shape of the output of the last layer are 30x30 image patches that classify 70x70 patches of the input image. By operating on patches rather than the whole image, the PatchGAN classifier encourages the discriminator to focus on local details and textures rather than relying solely on global image features. This patch-wise evaluation helps in achieving high-resolution image generation while maintaining perceptual coherence.

### 5.1.3 Generator Loss Calculation

The generator loss for the Pix2Pix GAN is a combination of the following two loss terms:

1. **Adversarial Loss:** is a loss term based on the idea that the generator should produce outputs that can not be differentiated from real images. It is calculated by passing the generated images through the discriminator and using the discriminator's output to compute the loss. It encourages the generator to produce reconstructed images that are more realistic such that the discriminator thinks they are real.
2. **L1 Loss:** is a term that measures the difference between the generated image and the target image in terms of pixel values. It is calculated by taking the absolute difference between the two images and summing over all pixels, which is an MAE (mean absolute error) between the generated image and the target image. The L1 loss helps the generator to produce images that are similar to the target image in terms of content.

The generator loss is the weighted sum of these two loss terms [21], where the weights are determined by hyperparameters. The objective of the generator is to minimize

this loss, which encourages it to produce images that are both visually pleasing and similar to the target image.

Therefore, the generator loss can be computed by the following formula:

$$\text{Generator Loss} = \text{Adversarial Loss} + \lambda \times \text{L1 Loss} \quad (2)$$

Where,  $\lambda$  is a hyperparameter that determines the relative weight of the L1 loss compared to the adversarial loss and its value in this implementation is 100 which was decided by the authors of [18].

#### 5.1.4 Discriminator Loss Calculation

The discriminator loss for Pix2Pix GAN is based on the idea that the discriminator should be capable of distinguishing between real and fake images. It is calculated as the sum of the following two loss terms:

1. Binary Cross-Entropy Loss: is a loss term that calculates the difference between the predicted probability of the discriminator for real images and the target value of 1, and the predicted probability of the discriminator for fake images and the target value of 0. It is calculated by taking the negative log of the predicted probability for the target class, summed over all images. This loss encourages the discriminator to correctly classify the images as real or fake.
2. L2 Weight Regularization Loss: This is used to prevent overfitting of the discriminator. It is calculated by taking the L2 norm of the discriminator weights and multiplying it by a regularization coefficient. This loss encourages the discriminator to have small weights and helps to prevent overfitting.

The discriminator loss is the sum total of these two terms [21]. The objective of the discriminator is to maximize this loss, which allows it to correctly classify real and fake images, while also keeping the weights small to prevent overfitting.

The discriminator loss can be calculated by the following equation:

$$\begin{aligned} \text{Discriminator Loss} = & -[\log(D(x)) + \log(1 - D(G(z)))] \\ & + \lambda \times \text{L2 weight regularization loss} \end{aligned} \quad (3)$$

where,  $D(x)$  is the discriminator's prediction for the real image  $x$ ,  $G(z)$  is the generator's prediction for the fake image generated from the noise input  $z$ , and  $\lambda$  is the regularization coefficient.

#### 5.1.5 Attention block

The basic implementation of the Pix2Pix GAN model available on the Tensorflow website [21], yielded good results for the weather condition dataset. However, it did not yield favorable results with the lighting condition dataset. As most of the colors were overexposed in the bright-light augmentation, it was difficult for a basic Pix2Pix GAN generator to recreate the correct color and shape in bright light

images, leading to model failure. Hence, it was necessary to improve the generator performance to procure better results for both datasets. This was made possible by modifying the Generator architecture. This research attempted to modify the upsampler, downsampler, and skip connections in the generator by adding a simple attention mechanism to yield better results on both datasets.

An attention mechanism is a component often used in neural networks, particularly in deep learning models for various tasks involving sequences or spatial data. Its primary purpose is to allow the model to focus its computational resources on specific parts of the input data, effectively learning to "pay attention" to relevant information while ignoring irrelevant or noisy details. Hence, this research used a basic attention mechanism that combines average pooling and max pooling to compute attention weights and apply them to the input feature maps.

The attention block was inserted in the downsample (encoding layer) and upsample (decoding layer) functions for the Generator. This means that both the upsampling and downsampling paths of the U-Net generator will have attention mechanisms applied to them. The attention block helps the generator attend to relevant features and regions at each stage of encoding and decoding. Additionally, the attention mechanism in the skip connection contributed to the network's ability to selectively combine and emphasize important features from different scales of the input data. Skip connections help the generator maintain fine-grained details and improve the overall quality of the generated image. This allowed the model to handle lighting variations better and improve image normalization by directing its attention to relevant details and features in the presence of varying lighting conditions.

Thus, by integrating the attention block in the upsampler, downsampler, and the skip connection, it was possible to create a model that could focus on the important regions of the feature maps both during the downsampling (encoding) and upsampling (decoding) stages, as well as when combining skip connections. Enabling the model to produce high-quality, visually coherent output images while handling variations in lighting conditions and preserving fine details. It effectively enhanced the generator's capacity to translate low-resolution feature maps into high-resolution images that are similar to the desired output.

This model successfully converged for both datasets and was capable of effectively normalizing images for both light and weather variation.

## **5.2 Pix2Pix GAN model training and result analysis**

### **5.2.1 Training**

During the training, the Pix2Pix GAN model was able to reconstruct images of various driving scenes successfully for both datasets. The model was trained multiple times for each dataset by varying batch size, and epochs and by including and discluding the attention mechanism. It was determined that training the models for 5 to 7 epochs was enough to train Pix2Pix for both sets. However, it was necessary

to use batch-size of 1 for the weather condition dataset, while batch-size of 1 and 4 both showed promising results for the lighting condition dataset. The size of the training data was approximately 48,000. Based on the dataset size and epoch values presented by the various implementations of Pix2pix GAN in [18], the steps taken to train the model till convergence were relatively the same. Throughout the training, the discriminator and generator losses were monitored to ensure that one did not dominate the other. It was done to ensure that there was no model failure or collapse.

## 5.2.2 Pix2Pix Generated Results

The following images display some of the GAN-generated results for the weather condition dataset.

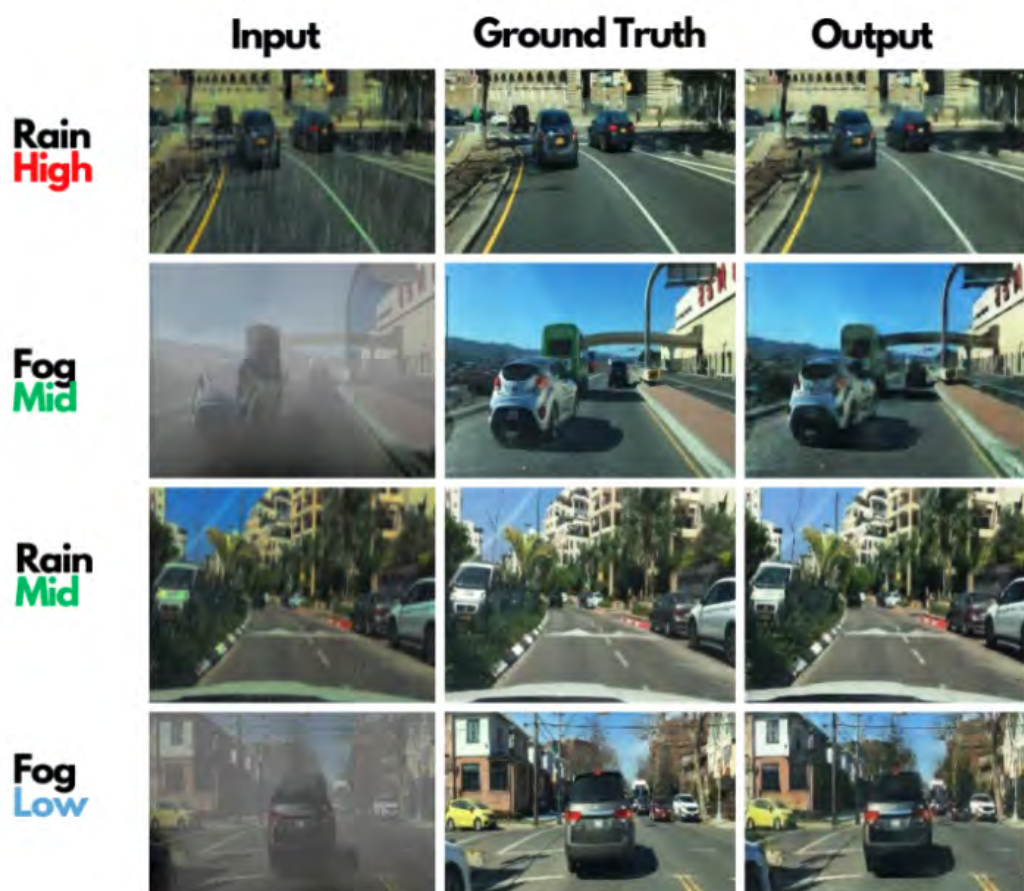


Figure 5.2.1: Pix2Pix GAN output on weather condition dataset

The following images display some of the GAN-generated results for the lighting condition dataset.

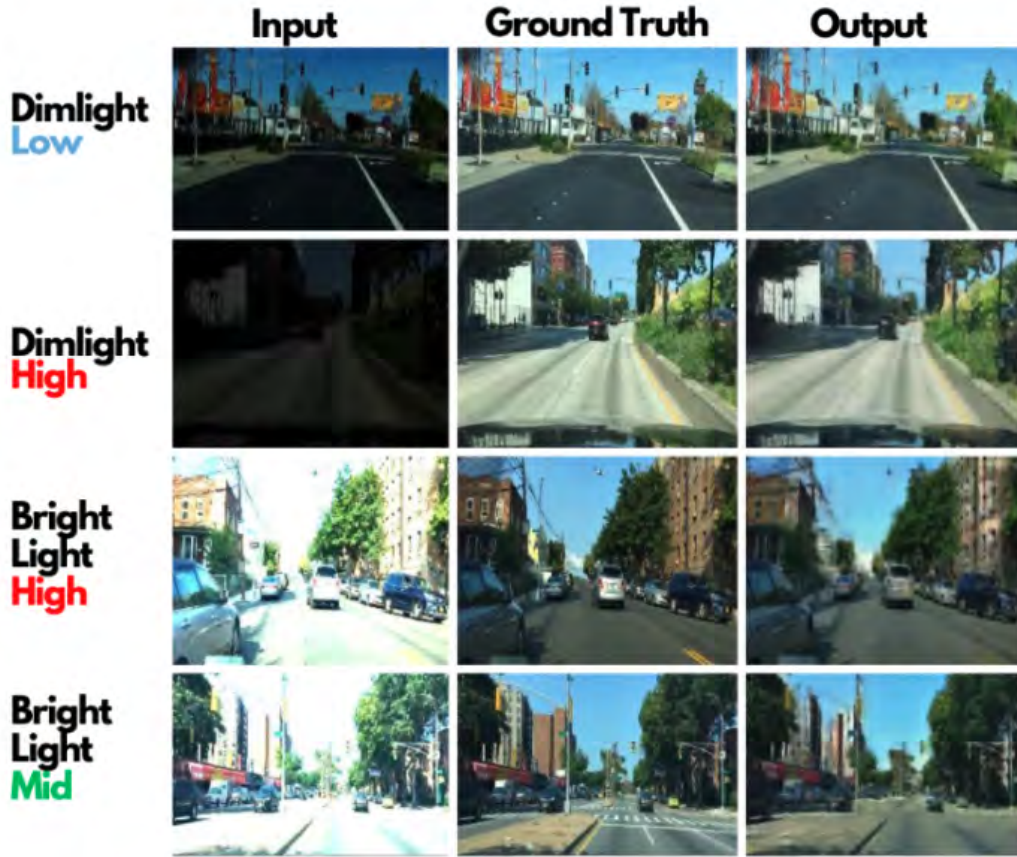


Figure 5.2.2: Pix2Pix GAN output on lighting condition dataset

### 5.2.3 Primary result analysis

Before the generated images are passed into the object detection model for testing, this research chooses to find the Structural Similarity Index and Peak Signal to Noise Ratio of the generated images to analyze the primary outputs. These metrics can effectively show how accurately the model preserves the structure and details of the target image, before the object detection model determines how the accuracy of object detection is increased after normalization.

#### 1. Peak Signal to Noise Ratio (PSNR)

Typically used in the context of digital image and video processing, PSNR is commonly used as a quality measurement technique to quantify the noise or distortion present in the reconstructed images when compared to its original, uncompressed images [28]. It does this by comparing the pixel values of the original and reconstructed signals, calculating the average squared differences, and converting this into a logarithmic scale. Thus, the formula for PSNR is given as follows:

$$\text{PSNR} = 10 \times \log_{10} \frac{\text{MAX}_I^2}{\text{MSE}} \quad (4)$$



Where  $MAX_I$  is the peak value for a pixel. This refers to the maximum possible value that a pixel or sample in the image or video can take. This value is typically 255 for 8-bit images (where each channel can take values from 0 to 255) [28]. And MSE is the mean squared error calculated as the average of the squared differences between the original and reconstructed pixel values.

PSNR is expressed in decibels (dB) where higher PSNR values indicate better image quality. Therefore a higher PSNR score means that the reconstructed image is closer to the original image, with less distortion or noise.

Although PSNR provides a quantitative assessment of quality, it still has some limitations as it does not always align perfectly with human perception of image quality, particularly when dealing with compression artifacts or other specific types of distortion as it may not capture all aspects of perceived quality. Therefore, for a more comprehensive evaluation of visual quality, other metrics like the Structural Similarity Index (SSIM) or subjective human testing may be employed.

## 2. Structural Similarity Index (SSIM)

To overcome the limitations of PSNR, this research uses SSIM as another image quality assessment tool to determine the effectiveness of normalization via pix2pix GAN. SSIM is designed to provide a more comprehensive evaluation of quality as it refers to the visual impact of changes in luminance, contrast, and structure [29].

SSIM takes into account the similarity in the luminance (brightness) between the reference and reconstructed images and assesses how well the overall brightness levels are preserved. It then evaluates the contrast of the images, which is related to the difference in brightness between various parts of the image, and measures how well the contrast structure is maintained. The structural content of the images being compared is also evaluated on how well the local patterns of pixel intensities, such as edges, textures, and details, are preserved in the reconstructed or compressed image when compared to the reference image. Finally, the three terms are combined to produce the structural similarity index.

$$S(x, y) = f(l(x, y), c(x, y), s(x, y)) \quad (5)$$

Once, all the terms are put together, the formula is represented as follows:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (6)$$

This yields a value of SSIM between 0 to 1, where 1 represents a perfect match.

Figure 5.2.3 shows the SSIM and PSNR results of the 6,000 images tested by Pix2Pix. Figure 5.2.3 (a, b) shows that after augmentation the SSIM scores (red bars) are distributed across a higher range of SSIM in the charts for both weather and lighting conditions. While after normalization, the SSIM values (blue bars) are more concentrated towards the right with a higher count. Thus the normalized bars should

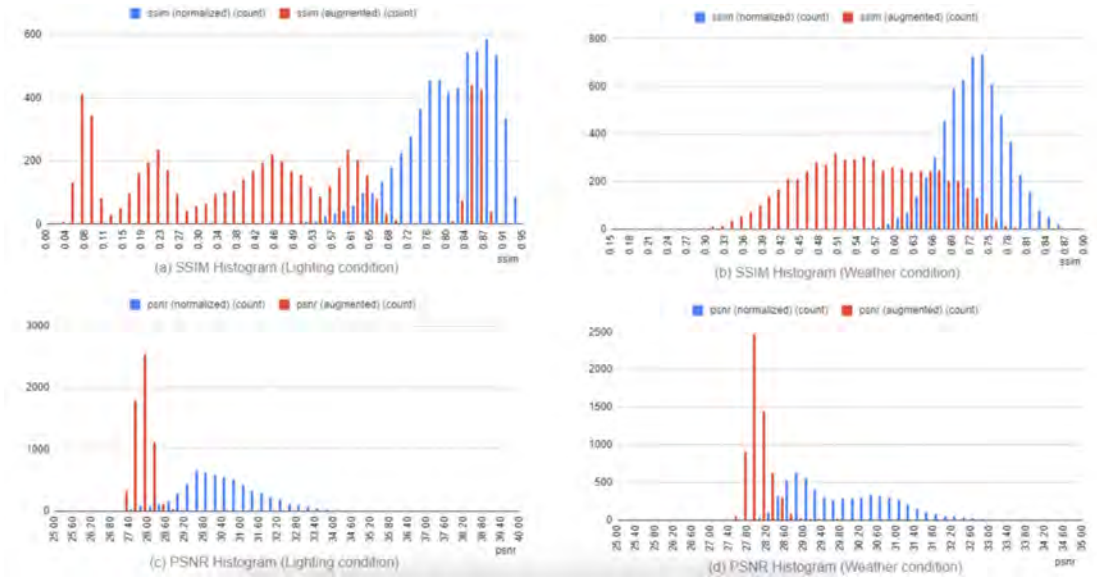


Figure 5.2.3: Histograms comparing augmented and normalized SSIM and PSNR scores for batch size 1

have an overall greater average SSIM than the augmented SSIM. The distribution of PSNR scores in Figures 5.2.3 (c, d) however, shows the opposite nature. After augmentation the PSNR (red bars) have moved towards the left, and are more concentrated within a short range, indicating a lower average. After normalization, the PSNR scores have become more distributed and cover a larger area of higher PSNR scores on the right.

Table 5.1: Mean PSNR and SSIM for augmented and normalized images

Condition	Batch size	SSIM		PSNR	
		Augmented to target	Normalized to target	Augmented to target	Normalized to target
Weather	1	0.552	0.728	28	29.9
	4		0.605		29
Lighting	1	0.433	0.807	27.8	30.5
	4		0.779		29.4

Table 5.1 represents the mean scores for the augmented to target image and the normalized image to the target image for batch sizes 1 and 4. The augmented to target image column represents the mean similarity of the clear original to the noisy augmented image. While the normalized to target represents the mean similarity between the normalized and the original image. The comparative scores of these two columns show that both the SSIM and PSNR scores increase after normalization. This means that the quality of normalized images has improved and is closer to the original image. Hence, the GAN model was successful in normalizing the images in bad weather and lighting conditions to images that are closer to clear and normalized images.

Additionally, the table suggests that the normalization results are better for a batch

size of 1 in both cases which also proves to be true by the various implementations in [18]. While the lighting condition dataset yields better results than the weather condition regardless of batch size. This could be due to the use of the attention mechanism which was used to focus on the brightness and contrast to improve the results of the lighting variation rather than to put focus on the obstruction of rain or fog.

It should be noted that the SSIM and PSNR are used to analyze the quality of reconstructed images and can not truly tell if the object detection accuracy would increase after normalization. Hence, in the next section, this research attempts to classify and compare the results to judge the improvement of object detection accuracy before and after normalization.

### 5.3 Object Detection Model Implementation

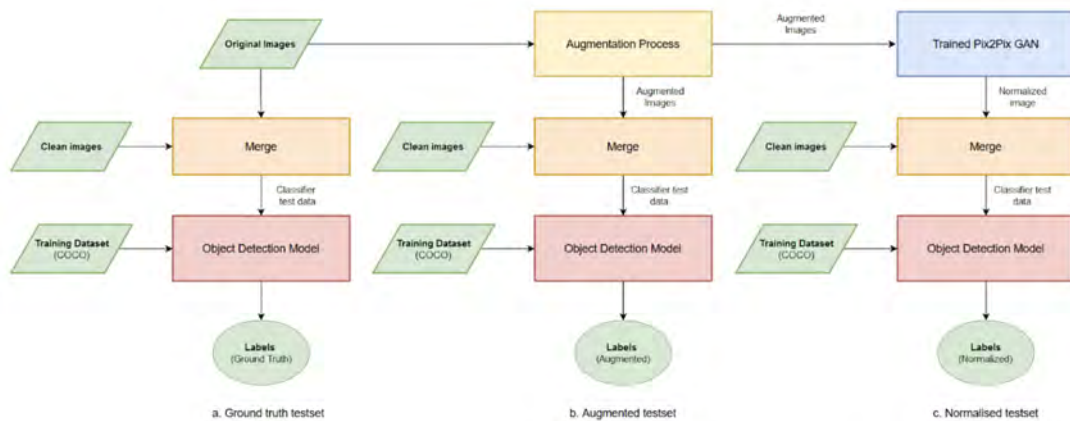


Figure 5.3.1: Flowchart representing object detection implementation

Before implementing the object detection models, the test data required to compare the results had to be created. For the 2 conditions, this study created 3 test datasets each. The 6,000 GAN testing images were used for this purpose. And an additional 4,000 images in clear weather conditions were also required to balance the 6,000 synthetic test datasets with normal images. First, the original 256x256 version of the test images were merged with the 4,000 clean images to create the ground truth test data. This is required to find the ground truth labels. Next, the 6k augmented images were merged with the same 4,000 clear images to form the augmented test dataset. Which was needed to find the detectable labels in various weather/lighting conditions. Lastly, the normalized versions of the 6,000 images were merged with the same 4,000 images to form the normalized test dataset, which would be tested to see if the label accuracy increases compared to the augmented labels.

To implement the classifier model, the COCO dataset was used. COCO stands for Common Objects in Context [31], and is a large open-source dataset of labeled images that is often used for object detection benchmarking. Although it has 80 classes, the 13 most relevant classes were selected to evaluate the models used by this research. The three object detection models that were used were imported from

the Pytorch module which were pre-trained by COCO.

Object detection was first carried out on ground truth images to find true labels produced by each model. Then the augmented test data and the normalized test data were passed through the object detection models. Since the object detection models were essentially, multi-label classifiers the models were used to detect the location of multiple objects of different classes in a single image. Objects that had a prediction probability of only 50% or above were considered. Once the label information was extracted from the 3 test sets of images, this research attempted to evaluate various accuracy measures to understand whether or not the accuracy of object detection improves after normalization with Pix2Pix.

## 5.4 Object Detection Model Evaluation

### 5.4.1 Performance Measures

To evaluate the model accuracy, precision, recall, F1 scores, IoU, and mAP were found for the object detection models. In this research, the Intersection over Union (IoU) plays an important role in finding all the metices. An IoU value of 0.5 was used as a threshold to measure the True Positives and False Positives, and False Negatives needed to calculate the selected metrics.

IoU

In object detection, the IoU is the calculation of the intersection of a ground-truth bounding box, and the prediction bounding box. It is done to find the accuracy of the predicted box compared to the ground truth.

$$\text{IoU} = \frac{\text{area of intersection}}{\text{area of union}} \quad (7)$$

Where, the area of intersection is the overlapping area between the two bounding boxes, and the area of union is the total area covered by the boxes. If the IoU is over 0.5, this object is considered a true positive and if it is below it is a false positive. While, if the bounding box was present in the ground truth labels but the same label was not found in the augmented/normalized labels then it was marked as a false negative. These values are then used to calculate the precision, recall, F1, and mAP for the models.

Precision

Precision measures the proportion of correctly identified positive predictions (true positive) to the total number of positive predictions (true and false positives). This is done to find out how many of the positive predictions are correctly identified, by comparing them to the number of false positives.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \quad (8)$$

## Recall

The next metric, recall, is the proportion of correctly identified positive predictions to the total number of actual positive predictions that were detected and not detected (true positive and false negative respectively). This is done to gauge the percentage of how often the model would detect a label belonging to a certain class. A low recall value suggests that the model is missing a significant number of positive instances while a high recall value indicates that the model is successfully capturing a large proportion of the true positive instances.

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \quad (9)$$

## F1

The precision gives an understanding of the frequency of true positives among all positives that were detected, and recall measures the true positives found from the actual positives. Both metrics have their respective flaws hence, the F1 score is used to find the harmonic mean of precision and recall and display an overall result of both.

$$\text{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (10)$$

## mAP

Lastly, mAP stands for mean average precision. This study determined the mean Average Precision (mAP) within the range of mAP at 0.50:0.95. For each class, the Average Precision (AP) at each IoU threshold within the specified range of 0.50 to 0.95 was calculated with a certain step size. This was found by plotting a precision-recall graph. Then the mean of the AP values across all classes is computed to find the class-wise mAP. Finally, the average of all class-wise mAP is determined as the mAP at 50:95 for this study. This evaluation is often used to assess the model's performance in accurately pinpointing objects within a moderate to high degree of precision.

$$mAP = \frac{1}{n} \sum_{k=1}^n AP_k \quad (11)$$

where  $AP_k$  is the AP of class  $k$  and  $n$  is the number of classes.

## 5.4.2 Comparative results of various models

These performance measures for lighting and weather conditions datasets are discussed in this section. To begin with, tables 5.2 and 5.3 display the overall performance scores for both augmented and predicted (GAN reconstructed) images.

Table 5.2: Average scores for lighting condition dataset

Model		Augmented				Normalized			
OD	Backbone	Precision	Recall	F1	mAP[0.5:0.95] %	Precision	Recall	F1	mAP[0.5:0.95] %
Faster R-CNN	ResNet50	0.923974	0.549678	0.68189	6.503907	<b>0.929177</b>	<b>0.563833</b>	<b>0.691037</b>	<b>9.009981</b>
Faster R-CNN	MobileNet v3	<b>0.892006</b>	0.54977	0.671037	11.2582	0.88082	<b>0.564672</b>	<b>0.674192</b>	<b>13.19141</b>
RetinaNet	ResNet50	<b>0.961544</b>	0.526538	0.662287	4.845578	0.95631	<b>0.54115</b>	<b>0.67208</b>	<b>7.62014</b>

Table 5.3: Average scores for weather condition dataset

Model		Augmented				Normalized			
OD	Backbone	Precision	Recall	F1	mAP[0.5:0.95] %	Precision	Recall	F1	mAP[0.5:0.95] %
Faster R-CNN	ResNet50	<b>0.949035</b>	0.441928	0.594995	3.625719	0.92669	<b>0.486941</b>	<b>0.62854</b>	<b>6.542908</b>
Faster R-CNN	MobileNet v3	<b>0.969856</b>	0.469928	0.622541	3.807608	0.91212	<b>0.542174</b>	<b>0.676287</b>	<b>9.102525</b>
RetinaNet	ResNet50	<b>0.973339</b>	0.430696	0.588384	2.487034	0.964575	<b>0.479272</b>	<b>0.629095</b>	<b>4.873164</b>

Precision, Recall, F1

The results for both lighting and weather conditions presented in Table 5.2 and 5.3 respectively are very similar. In both cases, when comparing the overall recall in the augmented and normalized columns, it can be observed that the scores increase after the image is normalized. This result is consistent for all the used object detection models. Notably, the precision scores are mostly higher for the augmented images regardless of the dataset or model it is tested on. This could be due to the presence of higher False Positives values in predicted results compared to the augmented results (ie. the model incorrectly predicts the presence of an object when it is not present). Additionally, the harmonic means of precision and recall, ie. the F1 score is higher for normalized images. F1 is a better measure of the models' performance as it takes into account both the false positive and false negative rates. Therefore, this higher value of F1 holds more weight in indicating the improved accuracy after normalization.

mAP

The mAP score is another important performance metric frequently used for object detection benchmarking as it can measure the overall effectiveness of the models. Similar to the F1 score, the AP also considers the precision-recall trade-off and includes the false positive and false negative values. This research computed the mAP at 0.5 to 0.95 IoU threshold to evaluate the model performance between a moderate and high IoU. According to Tables 2 and 3, the mAP scores of the normalized images are higher than the mAP scores of the augmented images. This indicates that the overall effectiveness in detecting objects is better after normalization.

## Class-wise IoU

To further analyze the accuracy of the models, the class-wise IoUs between 0.5 and 0.95 were calculated for the 13 observed classes which are tabulated below:

Table 5.4: Lighting condition dataset: Average class-wise IoU@[0.5:0.95]

Class name	Faster R-CNN with ResNet50		Faster R-CNN with MobileNet v3		RetinaNet with ResNet50	
	Augmented	Normalized	Augmented	Normalized	Augmented	Normalized
Airplane	0.86682	<b>0.870647</b>	0	0	0	0
Bench	0.879233	<b>0.905556</b>	<b>0.927427</b>	0.796013	0	<b>0.935652</b>
Bicycle	0.85089	<b>0.870156</b>	<b>0.905421</b>	0.88896	0.834921	<b>0.853349</b>
Bus	0.878696	<b>0.885315</b>	<b>0.898511</b>	0.895404	<b>0.883391</b>	0.877873
Car	0.879638	<b>0.884366</b>	0.881942	<b>0.884171</b>	0.885016	<b>0.889385</b>
Fire hydrant	<b>0.941558</b>	0.925862	<b>0.925368</b>	0.907019	0.879438	<b>0.895415</b>
Motorcycle	0.842383	<b>0.882135</b>	0.880548	<b>0.924062</b>	0.862823	<b>0.931129</b>
Parking meter	0.878244	<b>0.889585</b>	<b>0.918629</b>	0.883882	<b>0.936351</b>	0
Person	0.86244	<b>0.864506</b>	0.859896	<b>0.8668</b>	0.887713	<b>0.890647</b>
Stop sign	0.904481	<b>0.916561</b>	0.875711	<b>0.921946</b>	0.92293	<b>0.933787</b>
Traffic light	0.891509	<b>0.901019</b>	0.865548	<b>0.869441</b>	0.893487	<b>0.902025</b>
Train	0.819879	<b>0.839381</b>	<b>0.882938</b>	0.861759	<b>0.858915</b>	0.824244
Truck	<b>0.873993</b>	0.869834	<b>0.890742</b>	0.889501	<b>0.881398</b>	0.879983
<b>Average IoU</b>	<b>0.8745972308</b>	<b>0.8849940769</b>	<b>0.8927234167</b>	0.8824131667	0.8790032	<b>0.8877837</b>

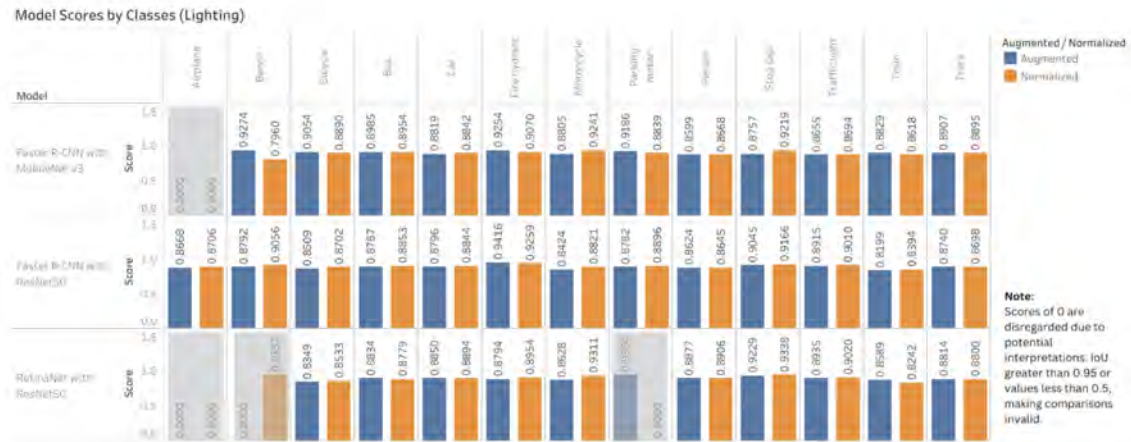


Figure 5.4.1: Class-wise IoU scores for each Model (Lighting)

According to Table 5.4 and Figure 5.4.1 comparing the IoUs of the lighting condition dataset, the Faster R-CNN with RestNet 50 backbone shows the most significant improvement since the class-wise average IoUs of most of the class (11 of 13) show an increase after prediction. This model is known to be the most accurate of the 3 models used. Similarly, RetinaNet with RestNet50 also shows an increase in prediction for the majority of the classes (7 of 10). However, Faster R-CNN with MobileNetv3 does not show much improvement; less than half of the identified IoUs show improvement (5 of 12). All the observed changes are further supported by the total mean of all classes, which shows that the average for the first and last models is higher after normalization, while for the second model, the score is lower after normalization.

Table 5.5: Weather condition dataset: Average class-wise IoU@[0.5:0.95]

Class name	Faster R-CNN with ResNet50		Faster R-CNN with MobileNet v3		RetinaNet with ResNet50	
	Augmented	Normalized	Augmented	Normalized	Augmented	Normalized
Airplane	0	0.830098	0	0	0	0
Bench	0.888203	0.917532	0	0	0	0
Bicycle	0.80964	0.879396	0	0.772661	0.896676	0.945207
Bus	0.859634	0.861385	0.847662	0.887184	0.89579	0.87189
Car	0.856944	0.871116	0.861	0.875916	0.878033	0.878473
Fire hydrant	0	0.907408	0	0	0	0
Motorcycle	0.943273	0.861622	0	0	0	0
Parking meter	0.861252	0.870457	0	0.907999	0	0.905662
Person	0.820555	0.822419	0.865467	0.872014	0.836424	0.837676
Stop sign	0.925794	0.922411	0.934955	0.941632	0.939704	0
Traffic light	0.880217	0.883596	0.867092	0.890909	0.90442	0.901798
Train	0.822354	0.804695	0.784073	0.820546	0.850484	0.808632
Truck	0.847111	0.85448	0.866989	0.869318	0.850984	0.847531
Average IoU	0.8649979091	0.8681008182	0.861034	0.8796455714	0.8693558333	0.8576666667

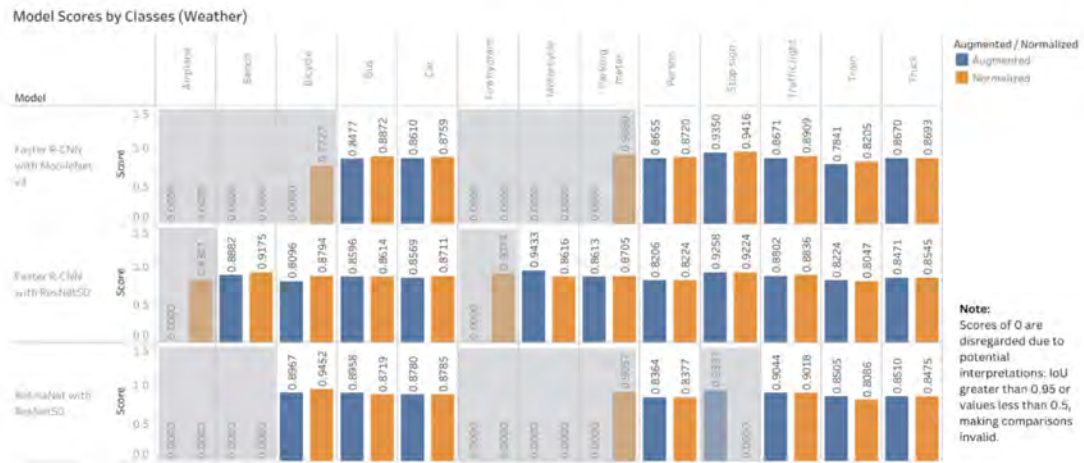


Figure 5.4.2: Class-wise IoU scores for each Model (Weather)

According to Table 5.5 and Figure 5.4.2 comparing the IoUs of the weather condition dataset, both Faster R-CNN with RestNet 50 (8 of 11) and MobileNet v3 (7 of 7), observe an improvement in a significant number of classes, as they show an increase in the average class-wise IoUs. However, in this test dataset, the RetinaNet with ResNet50 backbone does not show an improvement in IoU scores, unlike the lighting condition test data. Only 3 of 7 classes have an increase in this model. This is further supported by comparing the mean IoU of all classes in the last row, which shows that the first two models show an increase in the total mean, while the third model does not.

It should be noted that the 0 values indicate that no IoUs of the range 0.5 to 0.95 were observed for those classes. This is mostly due to the class imbalance, as very few instances of those objects exist within the test data. Hence, if a class was identified for the normalized test data but was not found for the corresponding augmented



test data in the same model or vice versa, then it was not taken into consideration for the comparison.

Of the 3 models used, Faster R-CNN with RestNet 50, is the slowest and the most accurate model and shows drastic improvement for both the dataset. Faster R-CNN with MobileNetv3, the fastest and least accurate model, shows improvement in the weather condition dataset but average results for the lighting condition dataset. Similarly, RetinaNet with RestNet50 backbone shows an improvement for lighting conditions but does not show the same for weather conditions. This suggests that the accuracy of the model itself makes a significant difference in whether or not the accuracy of the detection will improve after normalization.

# Chapter 6

## Future Work and Conclusion

The results of this research suggest that normalizing images in bad weather and lighting conditions shows an improvement in object detection accuracy. Although Pix2Pix was successful in improving the overall quality to improve detection, the scores of accuracies are still very low after normalization. Hence, more effective and real-time normalization methods should be explored in the future. Additionally, the SSIM scores of Pix2Pix are average for the dataset used by this research. However, it is yet to be determined if these values could be improved by increasing the number of variations per condition. While this research attempted to train the Pix2Pix with only 3 variations per condition (rain, fog, low and bright light), future research should try to analyze the effects of varying the number of variations per condition on the performance of Pix2Pix.

This study also came across various other challenges when augmenting the images for the datasets. The images augmented using imgaug, OpenCV, and other tools were not very realistic when producing the artificial fog and dim lighting conditions. However, GANs could be used instead to augment the images. This creates scope for further research ideas such as investigation of the use of data augmentation techniques specific to the image-to-image translation tasks. This could involve exploring geometric transformations, color augmentation, or domain-specific transformations to improve the diversity and generalization capability of the generated images. Similarly, real-time and low-resource implementations of GAN is also a unique and relevant field that needs to be explored, if it is to be used in autonomous vehicles. This could include exploring model compression, quantization techniques, or network architecture adaptations for resource-constrained environments.

To conclude, autonomous vehicles are developed with the hopes that they will help reduce traffic problems, make traveling more accessible to the public, and improve the overall safety of roads. However, there is a significant gap in their ability to function in a real-life scenario, due to the lack of precision in the models and technologies that have been developed so far. Much work is still left before they can be introduced as a viable option for transportation. Therefore, this research attempts to increase the accuracy, and hence the usability of such vehicles by employing advanced color vision techniques and using GANs.

# Chapter 7

## References

- [1] Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object Detection in 20 Years: A Survey. ArXiv: Computer Vision and Pattern Recognition.
- [2] Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11), 3212–3232.
- [3] Hassaballah, M., Kenk, M. A., Muhammad, K., & Minaee, S. (2021). Vehicle Detection and Tracking in Adverse Weather Using a Deep Learning Framework. *IEEE Transactions on Intelligent Transportation Systems*, 22(7), 4230–4242.
- [4] Kenk, M. A., & Hassaballah, M. (2020). DAWN: Vehicle Detection in Adverse Weather Nature Dataset. ArXiv: Computer Vision and Pattern Recognition.
- [5] Campbell, S., O’Mahony, N., Krpalcova, L., Riordan, D., Walsh, J., Murphy, A., & Ryan, C. (2018). Sensor Technology in Autonomous Vehicles: A review. 2018 29th Irish Signals and Systems Conference (ISSC).
- [6] Amit, Y., Felzenszwalb, P., & Girshick, R. (2020). Object detection. *Computer Vision: A Reference Guide*, 1-9.
- [7] Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Glaeser, C., Timm, F., ... & Dietmayer, K. (2020). Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3), 1341-1360.
- [8] Gupta, A., Anpalagan, A., Guan, L., & Khwaja, A. S. (2021). Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10, 100057.
- [9] Stilgoe, J. (2021). How can we know a self-driving car is safe?. *Ethics and Information Technology*, 23(4), 635-647.
- [10] Usama, M., Qadir, J., Raza, A., Arif, H., Yau, K. L. A., Elkhatib, Y., ... & Al-Fuqaha, A. (2019). Unsupervised machine learning for networking: Techniques, applications and research challenges. *IEEE access*, 7, 65579-65615.

- [11] Zlateski, A., Jaroensri, R., Sharma, P., & Durand, F. (2018). On the importance of label quality for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1479-1487).
- [12] Chadwick, S., & Newman, P. (2019, June). Training object detectors with noisy data. In 2019 IEEE Intelligent Vehicles Symposium (IV) (pp. 1319-1325). IEEE.
- [13] Raj, M. M., Tasdid, S. H., Nidra, M. A., Noor, J., Ria, S. A., & Alam M. A. (2021). A color vision approach considering weather conditions based on autoencoder techniques using deep neural networks. In 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE) (pp. 1-12). IEEE.
- [14] Gomes, P. R., Sabuj, H. H., Uddin, M. A., Reza, M. T., Faiz, R. I., & Alam, M. A. (2021). A deep learning approach for reconstruction of color images in different lighting conditions based on autoencoder technique. In 2021 International Conference on Electronics, Information, and Communication (ICEIC) (pp. 1-4). IEEE.
- [15] Klinker, G., Shafer, S.A. & Kanade, T. (1989). Color image analysis with an intrinsic reflection model. In [1988 Proceedings] Second International Conference on Computer Vision (pp. 292-296). IEEE.
- [16] Buluswar, S. D., & Draper, B. A. (1998). Color machine vision for autonomous vehicles. *Engineering Applications of Artificial Intelligence*, 11(2), 245-256.
- [17] Fabbri, C., Islam, M. J., & Sattar, J. (2018). Enhancing underwater imagery using generative adversarial networks. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 7159-7165). IEEE.
- [18] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125-1134).
- [19] Saxena, U. (2018). Automold road augmentation library. GitHub.[Online]. Available: <https://github.com/UjjwalSaxena/Automold-Road-Augmentation-Library>. [Accessed: 20-Apr-2023].
- [20] Ren, S., He, K., Girshick, R., Sun, J. (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv:1506.01497v3 [cs.CV]
- [21] Pix2pix: Image-to-image translation with a conditional GAN. (n.d.). TensorFlow. <https://www.tensorflow.org/tutorials/generative/pix2pix>
- [22] Gad, A. F. (2021). Faster R-CNN Explained for Object Detection Tasks. Paperspace.
- [23] Krishna, N. (2022). Understanding and Implementing Faster R-CNN: A Step-

By-Step Guide. Medium. <https://towardsdatascience.com/understanding-and-implementing-faster-r-cnn-a-step-by-step-guide-11acfff216b0>

- [24] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. doi:10.1109/CVPR.2016.90
- [25] Godard, C., Mac Aodha, O., Firman, M., & Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 3828-3838).
- [26] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., Adam, H. (2019). Searching for MobileNetV3. arXiv [Cs.CV]. Retrieved from <http://arxiv.org/abs/1905.02244>
- [27] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2018). Focal Loss for Dense Object Detection. arXiv [Cs.CV]. Retrieved from <http://arxiv.org/abs/1708.02002>
- [28] Johnson, D. H. (2006). Signal-to-noise ratio. Scholarpedia, 1(12), 2088.
- [29] Dosselmann, R., & Yang, X. D. (2011). A comprehensive assessment of the structural similarity index. Signal, Image and Video Processing, 5, 81-91.
- [30] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), 600-612.
- [31] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13 (pp. 740-755). Springer International Publishing.