Switching GAN-Based Image Filters to Improve Perception for Autonomous Driving

by

Zarif Masud

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2019

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Autonomous driving holds the potential to increase human productivity, reduce accidents caused by human errors, allow better utilization of roads, reduce traffic accidents and congestion, free up parking space and provide many other advantages. Perception of Autonomous Vehicles (AV) refers to the use of sensors to perceive the world, e.g. using cameras to detect and classify objects. Traffic scene understanding is a key research problem in perception in autonomous driving, and semantic segmentation is a useful method to address this problem. Adverse weather conditions are a reality that AV must contend with. Conditions like rain, snow, haze, etc. can drastically reduce visibility and thus affect computer vision models. Models for perception for AVs are currently designed for and tested on predominantly ideal weather conditions under good illumination. The most complete solution may be to have the segmentation networks be trained on all possible adverse conditions. Thus a dataset to train a segmentation network to make it robust to rain would need to have adequate data that cover these conditions well. Moreover, labeling is an expensive task. It is particularly expensive for semantic segmentation, as each object in a scene needs to be identified and each pixel annotated in the right class. Thus, the adverse weather is a challenging problem for perception models in AVs. This thesis explores the use of Generative Adversarial Networks (GAN) in order to improve semantic segmentation. We design a framework and a methodology to evaluate the proposed approach. The framework consists of an Adversity Detector, and a series of denoising filters. The Adversity Detector is an image classifier that takes as input clear weather or adverse weather scenes, and attempts to predict whether the given image contains rain, or puddles, or other conditions that can adversely affect semantic segmentation. The filters are denoising generative adversarial networks that are trained to remove the adverse conditions from images in order to translate the image to a domain the segmentation network has been trained on, i.e. clear weather images. We use the prediction from the Adversity Detector to choose which GAN filter to use. The methodology we devise for evaluating our approach uses the trained filters to output sets of images that we can then run segmentation tasks on. This, we argue, is a better metric for evaluating the GANs than similarity measures such as SSIM. We also use synthetic data so we can perform systematic evaluation of our technique. We train two kinds of GANs, one that uses paired data (CycleGAN), and one that does not (Pix2Pix). We have concluded that GAN architectures that use unpaired data are not sufficiently good models for denoising. We train the denoising filters using the other architecture and we found them easy to train, and they show good results. While these filters do not show better performance than when we train our segmentation network with adverse weather data, we refer back to the point that training the segmentation network requires labelled data which is expensive to collect and annotate, particularly for adverse

weather and lighting conditions. We implement our proposed framework and report a 17% increase in performance in segmentation over the baseline results obtained when we do not use our framework.

Acknowledgements

I would like to thank my supervisor, Krzysztof Czarnecki, for granting me the opportunity to work as a reseach assistant in the Waterloo Intelligent Systems Lab. My experience working there, and indeed my entire Waterloo experience has been an incredible one that has contributed to my growth both academically and at a personal level.

I wish to thank Sean Sedwards for his immense support, and invaluable feedback every step of the way. This thesis would not be possible without his help.

I am also fortunate to have a group of colleagues who have been always available to me with their array of resources and experiences that have helped me complete this project.

Lastly, and very importantly, I would never have been able to do this if not for friends and family who have been with me through the good times and the bad. I am eternally grateful.

Table of Contents

Chapter 1

Introduction

Autonomous driving research is one of the most lucrative areas of research in present times. Although there has been interest in the field since the 80s [17], in the past decade, the field has gained significant momentum. Currently, companies like Waymo, Tesla, Uber, etc. are on an all-out race to deploy fully autonomous cars on public roads, where the first company to reach market can stand to make significant profits by dominating market share. Both academia and industry are engaging in various aspects of Autonomous Driving Systems (ADS), from mapping and routing, to behaviour planning, tracking, perception, controls, etc.

Automated Vehicles (AV) hold the potential to increase human productivity, reduce accidents caused by human errors, allow better utilization of roads, reduce traffic accidents and congestion, free up parking, generalize valet services, etc. [1]. AVs can open doors for completely new paradigms for traffic control, routing for public transits, or change the modes of these transits themselves. Avoidable human-errors resulting from completely human conditions such as carelessness, drowsiness, or even momentary physical disturbance, etc. very often lead to loss of life and property of the individual responsible, as well as other parties who might have been in the vicinity. Autonomous driving can thus save value by both reducing loss of lives and cost significantly. Thus there is motivation for the market as well to invest heavily in autonomous driving research.

One of the major areas of research in autonomous driving is perception of AVs. Perception tasks in autonomous vehicles include identifying objects on the road, identifying other vehicles and pedestrians, detecting road signs, etc. Traffic scene understanding is thus a key research problem in perception in autonomous driving. Perception is carried out through sensory devices, primarily through camera, RADAR and LIDAR. In recent

times, advances in deep learning has resulted in unprecedented performance gains in all computer vision tasks, including perception in AVs. Deep learning can be used to detect or classify objects in 2D or 3D data.

1.1 Deep Learning and Semantic Segmentation

Deep learning research has been one of the most influential and fast-moving fields of research in the past decade. Since 2012, there has been some major milestones reached in the field of deep learning [39, 66, 67, 13, 12]. Computer vision algorithms for image detection or classification have reached and even surpassed human-level performance [67, 13]. This boom in deep learning research has also found application in various domains such as healthcare, natural language processing, finance, advertising, etc. One of the areas of application for computer vision has been in the field of autonomous driving.

One important field of research for autonomous driving and traffic scene understanding is semantic segmentation. Semantic segmentation is the task of identifying pixel-wise semantic categories. There are specialized deep learning models that are used for segmentation such as Deeplab [10, 11], PSPNet [81], FCCN [76] etc. Labelling data for segmentation is an expensive task [37], as each pixel in a scene needs to be labelled manually. Cityscapes [14] has been one of the most popular datasets for training and testing models for semantic segmentation. Other popular benchmark datasets for segmentation include Pascal VOC dataset [18], the ADE20K dataset [82], etc. These datasets contain traffic scenes, mostly urban, with relevant objects labelled. An obvious shortcoming of these datasets is that they mostly contain clear images under ideal weather conditions.

1.2 Adverse Weather and Lighting Conditions

Adverse weather conditions are a reality that AV must contend with. Conditions like rain, snow, haze, etc. can drastically reduce visibility and thus affect computer vision models. Models for perception for AVs are generally designed for and tested on ideal weather conditions under good illumination [30, 64]. Even when models are trained for bad illumination and visibility, there are edge cases that cannot be adequately represented by training or testing data. Edge cases can arise from any kind of unusual weather conditions, and from any kind of sensory noise. For example, camera input can be noisy and unreliable due to foggy weather, rain or other wet weather conditions, etc. [15].

AVs are safety critical, and thus they need to be robust to these conditions to be allowed on public roads. Adverse weather conditions can include rain, to snow, hail, fog, haze, clouds [51]. Lighting conditions can also have a strong negative impact on computer vision models. Visibility is maximum in clear, sunny weather during daytime. At night, light sources include artificial sources such as street lamps, car headlights, etc. Due to the differences in lighting, a model trained on daytime data will likely not perform well at night. Object detection using cameras is also affected during nighttime as color information is lost due to lack of illumination. Moreover, cameras suffer more from underexposure, noise, and motion blur under low light [15, 64]. Lighting conditions can also vary depending on cloud, fog, shadows, etc. Visibility can also be impaired by high illumination, such as when sun is directly facing the camera. These conditions can greatly affect the performance of computer vision models. The position of the light source can also cause variation in lighting and changes in shadows and as such result in varying levels of performance. Weather conditions can have other implications for computer vision models. Rainy weather can lead to puddles, accumulation of water droplets on camera lens, etc. Snow can cover roads and lane markings making it impossible to correctly identify them in severe cases.

Most approaches for computer vision and autonomous driving tasks are tested for clear weather images. Datasets that are used for training segmentation and object detection models for automotive scenes are generally clean images under ideal weather conditions. In [15], [57], [64], alternative approaches to train networks or adapt networks for various lighting conditions have been suggested. There are also datasets with inclement weather conditions in literature such as [72], [53]. These datasets are however not sufficiently large to cover all cases. Other issues with the datasets are discussed in Chapter 2. Various synthetic datasets have also been proposed in literature. Synthetic datasets give us certain advantages that are not possible with realistic datasets. They are also ideal for evaluation with finer control.

1.3 Challenges

The most complete solution may be to have our segmentation networks be trained on all possible conditions. If we had a dataset that included adequate rainy data, the segmentation network would perhaps learn to ignore rain streaks well. Moreover, puddles accumulate under rainy conditions. Visibility is also affected where objects in the distance are obscured more by rain streaks as well as rain particles in the air that scatter away from the rain droplets. Objects closer to the camera are also affected, albeit to a lesser extent. So a dataset to train a segmentation network to make it robust to rain would need

to have adequate data that cover these conditions well. Again, rain is only one of the conditions that make computer vision tasks tricky. Aside from rain, there are other weather conditions such as fog, mist, snow etc. as well as varying illumination. Night images have a completely different illumination than that of day images. Even clouds in the sky can make segmentation difficult. Hence it would be very difficult to obtain a dataset that could realistically cover all cases. Moreover, it is not ideal to retrain networks necessarily. We want to be able to benchmark a network and make safety guarantees along with metrics for robustness properties. Retraining networks with adverse weather and lighting may have unintended consequences.

Moreover, labeling is an expensive task. It is particularly expensive for semantic segmentation, as each object in a scene needs to be identified and each pixel annotated in the right class. Dataset preparation methods for semantic segmentation is a slow and laborious activity which is evident by the limited availability of such finely annotated datasets for semantic segmentation. Collecting adverse weather data that is highly representative, and then the task of labelling would be a monumental task. It would be ideal to have a way to train our segmentation networks in a more unsupervised manner so we can improve robustness to these kind of challenging conditions.

Thus, the adverse weather is a challenging problem for perception models in AVs. Realistic datasets for such adverse/inclement weather are scarce, and their quality is questionable. This is particularly true for semantic segmentation, where labelling pixel-wise annotations and ensuring quality of said labels is a costly endeavour. Synthetic datasets that are generated by a simulator can be cost-effective, quick to produce, and can give us a good playground for testing and evaluating models with edge cases.

If we do have a process to synthetically generate data in a way where we can define cases, follow some set of predefined rules that can deterministically generate adverse weather and lighting conditions, and also define location, camera position and angle, etc. we can test models and determine edge cases to be tested on. This will allow us to design specific scenes that we can test for. An example is a particularly complex intersection with heavy snow. We could potentially test a new approach to evaluate how well it performs for various subsets of data. This can give valuable insight into robustness of different models, and tell how well new techniques work to improve robustness work.

1.4 Contributions

We want to improve semantic segmentation for autonomous vehicles in the face of adverse weather and illumination, as well as evaluate our proposed approach systematically. We

overcome the data scarcity problem by working with a synthetic dataset. The approach we use to improve semantic segmentation is a filtering approach where filters are dynamically chosen using an Adversity Detector. We investigate how we can use Generative Adversarial Networks (GAN) to develop filters for our purposes which will result in improved semantic segmentation when used in our proposed framework. As the filters can work independently of the segmentation network, this gives us the opportunity to have a modular architecture for our framework, where filters can be replaced, added or removed. A rain filter can be replaced with a better rain filter. In this work we only explore the use of software approaches to improve semantic segmentation performance, as opposed to hardware filters. We also restrict our work to single-frame camera images.

We design a simple image classifier to work as the adversity detector. This module is responsible for choosing filters on-the-fly at inference time. The classifier can be a simple image classification network trained on our adverse weather data. Our dataset has metadata that gives us information such as levels of adversity, as well as type. So we can train the detector to be able to classify rainy images with light rain and heavy rain or high number of puddles or low, etc.

The choice of filter is motivated by the success of GANs in image-translation tasks. GANs [25] are a fairly new algorithm that has gained popularity for its ability to generate new, unseen data in a particular distribution by sampling an input from a different distribution. It does not need much in the way of priors. GANs have been one of the more promising fields of research in deep learning according to Yann LeCun [42]. GANs have since been used for a number of tasks including generating human faces [36], image-to-image translation [33], text-to-image translation [77], photo in-painting [56], etc. One advantage of using GANs is that we do not need pixel-level annotations in order to use them for our work. For example, GANs can take rainy images as input and give a target that is clean. Thus, we propose a modular approach that uses a filtering technique to improving segmentation performance for bad weather and lighting conditions. Our contributions are summarized as:

1. We propose a modular and unsupervised approach to improving semantic segmentation. The idea is that a core computer vision network will be used that is robust and we do not wish to retrain the network unless required. Separate modules can be installed on top of the network depending on what is necessary. These modules can be bench-marked and can be used to preprocess input data to translate to the domain the core network is trained on.

2. We explore the use of GANs to take images with adverse weather and lighting conditions– rain, puddles, clouds, low lying sun angles– and improve segmentation

performance on those images. We also explore how well GANs work with varying levels of rain, puddles and clouds. Additionally, we examine GANs that do not require paired data. We implement these filters and conduct a systematic evaluation of their performance on synthetic data. These GANs act as the modules that we want to use on top of the core network.

3. We train a model to detect adverse weather conditions and the kind of condition along with levels. This model is used to decide whether to apply a GAN filter, and if so, which particular filter to apply. We implement this framework and evaluate its performance on synthetically generated data.

1.5 Organization

Chapter 1 is the introduction which gives a rough idea of the domain, highlights the research questions and discusses the motivation for this work, as well as outlines the approach taken to solving the research problem. Chapter 2 discusses backgrounds and concepts that are relevant to understanding the thesis. The papers discussed should be enough to follow the state-of-the-art, as well as survey a brief history of the literature on relevant topics. We introduce the proposed framework in Chapter 3, where we discuss the overall framework, and proceed to details regarding each of the components. We discuss methodology of our research, and some parameters constraints we establish. Chapter 4 describes the experimental setup, the results obtained from our experiments and a brief discussion on the results. Chapter 5 concludes the thesis with relevant discussion and identifies future work on this research.

Chapter 2

Background

Advances in deep learning in the past decade have opened various avenues in computer vision research, as well as robotics and automotive research. Semantic segmentation is an important computer vision task that has seen significant use in perception for Autonomous Driving Systems (ADS). Most systems are trained for ideal weather conditions and are not found to be robust to variations from such ideal conditions. Adverse weather and lighting can drastically affect performance of these computer vision models. There have been research efforts on making them robust to various weather conditions and adverse illumination. Training with more data usually improves performance but it is often difficult to find high quality data in all kinds of conditions that may affect these models. It is particularly hard to find labeled data for segmentation as labeling is an expensive task by itself. We discuss these concepts in more depth in the following sections.

2.1 Autonomous Driving

Autonomous driving is an active area of research with widespread interest for the potential it holds. Automated Vehicles (AV) not only has the potential to increase human productivity, but improve safety by reducing accidents caused by human errors. According to [34], the average American spends 10,900 miles and more than 290 hours on the road every year. This translates to seven 40-hour weeks. The cost of driving is not only borne by the driver herself, but also has negative implications for external parties. According to [1], these costs to external parties are estimated to be 13 cents per mile approximately, which results in $1,300 worth of costs for every 10,000 miles driven. AVs will also allow for better utilization of roads, by reducing traffic accidents as well as congestion, free up

parking, etc. [1]. Hence there is a huge interest in autonomous driving research driven by both academia and industry, to make AVs safe and robust enough for deploying on roads.

Autonomous driving research has been around since the 1980s and 1990s [17], with works such as [71] displaying the viability of AV or robots to control their own navigation on roads and highways. During early years, only very simple driving scenarios such as distance keeping, lane changing, and simple intersection scenarios were possible. In 2005-2007, the US Department of Defense, specifically the Defense Advanced Research Projects Agency (DARPA), sponsored a series of competitions where 35 self-driving vehicles competed from both academia and industry. These vehicles, however, were not feasible or safe for public roads, having heavy modifications and used expensive, non-automotive-grade sensors [74].

Since then, particularly with the advancements in deep learning, there have been significant leaps in AV research [30]. Recent approaches to autonomous driving consists of multiple tasks such as scene parsing, lane following, path planning etc. AVs must have accurate and robust perception being able to carry out these tasks. Some of the core tasks in computer vision that are required for perception of AVs are discussed in the following section. We also discuss how adverse weather can affect the performance of computer vision networks and how approaches have been developed in literature to improve the models.

2.2 Advances in Deep Learning

Deep learning refers to hierarchical learning. It is generally achieved by using Multi-layered Perceptron (MLP) with very deep architectures comprising of many different layers with thousands of learned parameters. They are also referred to as Deep Neural Networks (DNNs). DNNs can learn hierarchical feature representation using backpropagation and can fit very complex functions due to the high degree of non-linearity it offers.

While neural networks have been around since the 1940s [41], it was not until Hinton et al. [63] showed how to train multi-layered neural networks using back-propagation that the field gained any real momentum. Using the same back-propagation technique, it is now possible to train very deep neural networks that can fit any arbitrary function sufficiently well. Nowadays, many complex research questions, particularly in computer vision and natural language processing, have been attempted with variants of DNNs such as the Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN) and Long short-term memory (LSTM), Generative Adversarial Networks (GAN), Autoencoders.

There has been a significant boom in the field of Computer Vision in the past decade, largely due to widespread use of CNNs. There are primarily two reasons for the shift

towards CNNs.

Availability of Data

In recent times, there has been a huge volume of data available for research. This data explosion has opened doors for many complex models to learn from the data to fit complex functions and give good predictions. Deep neural networks are a good example of this. A particular type of Deep Neural Network (DNN), called the Deep Convolutional Neural Network (DCNN) have found use in tasks such as image classification, object detection, semantic segmentation etc.

Powerful Graphical Processing Units (GPU)

Another reason for the emergence of DNNs is the availability of powerful GPUs that enable fast, multi-core hardware for parallel training. GPUs, paired with highly-optimized implementation of 2D convolution allow training of very large CNNs.

2.2.1 Convolutional Neural Networks

CNNs are a variant of MLP that uses hidden layers typically consisting of convolutional layers, pooling layers and fully connected layers [44]. CNNs can learn hierarchical representation from data, particularly spatial data, which makes it ideal for computer vision task. Compared to DNNs with fully connected layers, CNNs also have fewer parameters, as weights are shared. Computer Vision tasks such as image classification, object detection, and segmentation. have been around for a long time, but they were solved using conventional machine learning techniques. Such techniques required rich features to work correctly. Features for images are particularly difficult to extract. A common practice had been to extract features by hand using various statistical techniques. Feature extraction itself used to be a prominent field of research. But the process is difficult and costly. It would take a group of highly trained researchers and engineers many months to design and implement a fully functional image classification model from scratch [39].

Since the 1990s, Convolutional Neural Networks (CNN) have been shown to both work as good feature extractors as well as good classifiers [43]. Convolution layers could learn features by a process called back-propagation [45]. But at the time they were very expensive to train on high quality images. In [39], the authors proposed the AlexNet model that introduced much deeper networks on the model proposed by LeCun [43]. This was an

important milestone and a very influential publication as it started the deep learning boom for computer vision. Convolutional neural networks are highly suited for visual recognition tasks. CNNs are good feature extractors as well as classifiers [43]. This reduces the need for complicated and time consuming feature extraction and processing. As such, computer vision research has been heavily directed towards use of CNN and employing more advanced CNN architecture. Nowadays, it is common to train on millions of images for 1000s of classes using 10-20 Rectified Linear Units (ReLU) layers, with millions of weights and billions of connections between units. Training such a network a few years ago would have taken weeks but due to the advancement in hardware and software, it is now a task that takes no more than a few hours [44].

Since 2012, there have been a number of very powerful CNNs that have achieved human-level performance [66] and better [29] [68] [67]. CNNs have widely been used for object detection, classification and segmentation [16], as well as video and audio. These developments, particularly in the field of computer vision, has opened doors in various applied fields of research. One particularly important among them is perception for robotics and autonomous driving.

2.2.2 Generative Adversarial Network

Generative Adversarial Networks (GAN) have been proposed for the first time by Goodfellow et al. in [25]. The model contains two functions. One is the generator, G, which is used to generate a sample in the data distribution, and a discriminator, D, which is tasked with being able to tell whether the given sample is from the original distribution. The generator and the discriminator functions are generally two neural networks that are trained together by contesting with each other following ideas from game theory.

GANs have been one of the more promising fields of research in deep learning according to Yann LeCun [42]. In recent years, GANs have been used for a number of tasks including generating human faces [36], image-to-image translation [33], text-to-image translation [77], and photo inpainting [56].

2.3 Computer Visions Tasks in AVs

In the following sections, we discuss some key tasks in perception for AVs, but will mostly focus on semantic segmentation, which we argue is the more difficult task.

2.3.1 Image Classification

Image classification is the task of categorizing an image into classes based on some context. An example of the use of image classification is identifying time of day like early morning, noon, or evening or nighttime. Image classification could also identify scenes, places, etc.

2.3.2 Object Detection

Object detection in Computer Vision is the task of identifying objects in an image correctly by drawing bounding boxes around them to detect them, and then the task of identifying what class the object belongs to. For traffic scenes, a common problem is to detect vehicles and pedestrians in an image.

2.3.3 Semantic Segmentation

In computer vision, the task of identifying pixel-wise semantic categories is called semantic segmentation or scene parsing. It is a very useful technique for image understanding and analysis with applications in various domains of computer vision and artificial intelligence. A few applications include autonomous driving, robot navigation, industrial inspection, remote sensing, in cognitive and computational sciences, agriculture and medical imaging. Prior to the boom in deep learning research, traditional machine learning techniques such as random forests were used.

Segmentation is useful for perception in autonomous driving for a number of reasons. It is useful for scene understanding, identifying regions of interest, pedestrian and traffic sign detection, etc. In the past few years, like most computer vision tasks, deep learning has taken hold and drastically improved the state-of-the-art. There have been a number of high quality datasets for semantic segmentation such as Cityscapes [14]. More and more semantic segmentation models have been developed to achieve high scores on these datasets. We cover a few state-of-the-art models used for semantic segmentation.

DeepLab V3

The DeepLab [10] network is currently state-of-the-art for Cityscapes semantic segmentation task with a Mean Intersection Over Union (mIoU) of 81.3%, and 86.9% on Pascal VOC. The network uses atrous convolutions, and applies depth-wise separable convolution

to the atrous spatial pyramid pooling and decoder modules. It improves over previous DeepLab implementations by fine-tuning hyper-parameters, including batch normalization, varying batch size, larger crop size and changing output stride, among others. The DeepLab V3+ [11] achieves 89.0% IoU on Pascal VOC and 82.1% on Cityscapes. This model is an improvement on the V3 using output stride of 16 or 8 instead of 32, and using Xception modules.

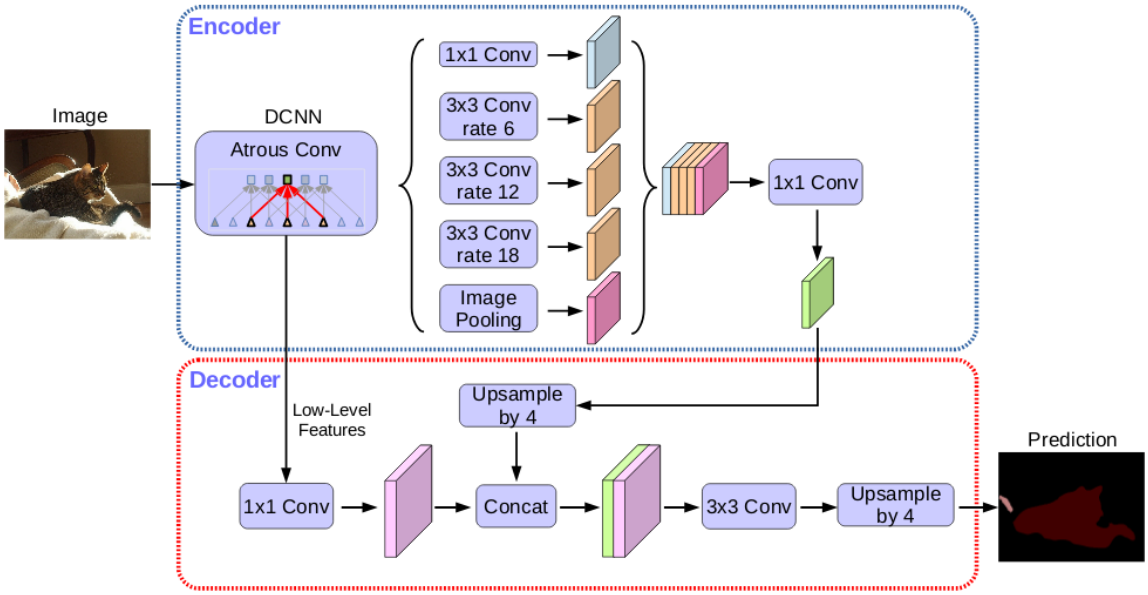


Figure 2.1: DeepLab V3+ [11]

PSPNet

PSPNet [81] stands for Pyramid scene parsing network. It uses a deeply supervised loss as an optimization strategy for deep ResNet-100. This improves learning and convergence. The loss functions include a main softmax loss to train the final classifier and an auxiliary loss that is applied after the fourth stage. PSPNet achieves an IoU of 80.2% on Cityscapes dataset, 85.4% on the Pascal VOC and 44.8% on ADE20K.

FCCN

FCCN [76] (highly fused fully convolutional network) uses a different cost function that calculates cost on each layer till the final output layer. This was shown to greatly improve segmentation performance. The network has a top score of 69.94% IoU on CamVid and a score of 44.23% on ADE20K dataset.

2.4 Adverse weather and lighting conditions

Weather conditions can drastically influence visibility and thus computer vision models are also generally affected. Perception algorithms are generally designed for and tested on ideal weather conditions under good illumination [30, 64]. One of the big challenges for autonomous vehicles is dealing with edge cases, where weather and lighting conditions are not optimal; camera input can be noisy and unreliable due to foggy weather, rain or other wet weather conditions, etc. [15]. AVs are safety critical, and thus they need to be robust to these conditions to be allowed on public roads.

These weather conditions can differ based on the types and sizes as well as the concentrations of particles in air. Clear weather is ideal for visibility, but under adverse conditions, light gets scattered by the particles in air. [51]

Haze: Haze is formed by small particles that can come from a variety of sources (such as volcanic ash, foliage exudation, combustion products) in air. Haze particles generally produce gray or bluish hue and affects visibility.

Fog: Fog is similar to haze in some ways, and forms when the humidity reaches a point of saturation. Fog generally has a more drastic effect on visibility than haze. In fog, the particles are larger and hence they cause more disturbance in visibility.

Rain: The above conditions are static in the sense that they do not change over a short period of time. Rain on the other hand causes temporal variations in scenes. The effect of rain on visibility depends on the levels of rain, the size and orientation of the droplets, wind direction, etc. Rain particles are also much larger and hence has a high amount of scattering. Visibility can be heavily affected by rain. Rain has other effects on scenes as well, such as, accumulation of puddles. Puddles can form reflective surfaces that can affect computer vision models.

Snow: Snow is similar to rain in that it is different from the static conditions. Snow particles are ice crystals that precipitate from the atmosphere. They cause high amount

of scattering and hence significantly affect visibility. Snow also accumulates and covers surfaces partially or completely and introduces other difficulties in object detection tasks.

Aside from weather conditions, we also have varying lighting conditions. The sun is the primary source of light in outdoor settings and as such, during daytime, visibility is maximum, given other conditions remain the same. At nights, there is no sunlight, and light sources include street lamps and car headlights. Due to the differences in lighting, a model trained with data from daytime, under ideal conditions will not perform well on night images. Object detection using cameras is also affected during nighttime as color information is lost due to lack of illumination. Moreover, cameras suffer more from underexposure, noise, and motion blur under low light. [15, 64]. Lighting conditions can also vary depending on cloud, fog, shadows, etc. Aside from lack of illumination that is found in nighttime, as well as twilight hours, visibility can also be impaired by high illumination. One such example is when the sun is directly facing the eyes, or the camera. These conditions can greatly affect the performance of computer vision models. The position of the light source can also cause variation in lighting and as such result in varying levels of performance.

Most approaches for computer vision and autonomous driving tasks are tested for clear weather images. Datasets that are used for training segmentation and object detection models for automotive scenes are generally clean images under ideal weather conditions. In [15], [57], [64], alternative approaches to train networks or adapt networks for various lighting conditions have been suggested. We discuss some of them in Sections 2.7 and 2.8.

2.5 Datasets for Semantic Segmentation

Datasets for semantic segmentation are expensive to collect as labelling for semantic segmentation is a costly task. In 2008, Brostow et al. [7] published the first segmentation dataset for road scenes. In 2015, the Cityscapes dataset [14] was published which became the de facto benchmark dataset for segmentation tasks. A variety of segmentation models have been designed and tested on the Cityscapes dataset. The dataset is of high quality and is representative of urban road scenes. Other important datasets for semantic segmentation include the Pascal VOC dataset [18], the ADE20K dataset [82], but these datasets suffer from a few criticisms. One is the lack of quantity of labeled images. Another is that most of the data is of clean images, under ideal weather and lighting conditions.

2.5.1 Realistic Adverse Weather or Lighting Datasets

Raincouver dataset [72], published in 2017, sought to tackle the problem of scarce adverse weather data for segmentation by including challenging rainy driving conditions in their published dataset. The dataset is meant to supplement existing datasets such as the Cityscapes dataset. The images ranged from day images, to that of dusk and nights. The published dataset contains 326 frames with hand-annotated pixel-wise semantic labels as well as half an hour of driving video captured on the roads of Vancouver, Canada.

In 2018, the Mapillary Vistas dataset [53] was published which contain 25,000 high-resolution images, 152 object categories, 100 instance-specifically annotated categories across a wide range of locations. The dataset contains different weather and lighting conditions, across different seasons and time of day. However, the dataset lacks metadata to identify these conditions and is hence unsuitable for evaluating impact of such conditions as well as training for certain conditions specifically or testing models that are designed to improve performance for a specific kind of condition.

Another noteworthy dataset for driving scenes is the Berkeley Deep Drive dataset. The dataset contains more variation in weather and lighting conditions than Cityscapes, but it suffers from the same issue that Mapillary dataset has in its lack of metadata. Moreover, the dataset is known to contain labeling inconsistencies in dark regions [64].

2.5.2 Synthetic Datasets

Synthetic data generation gives the advantage of being able to control conditions for scenes, as well as label pixels and assign metadata accurately.

Virtual KITTI [22] was released in 2016, which is a recreation of the KITTI dataset [23], but the images are replicated in 8 different weather and lighting variations. The dataset contains 21,260 frames of semantically labeled data. The authors provide impact analysis of weather and imaging conditions for object tracking but not for semantic segmentation.

Another example of synthetic dataset is the SYNTHIA dataset [62] published in 2016. The dataset contains 13,400 annotated images with varying weather and lighting conditions. In [31] the dataset was released with 2224 new images. One issue with the dataset is that there are no meta-data that in necessary for evaluating or developing models for different weather conditions separately.

Another approach to generate synthetic data is to use an existing game environment such as Grand Theft Auto V (GTAV) [61], [60]. The dataset released by Richter et al.

contains 254,064 images. In [2], the authors proposed an easier way to generate data from GTAV and published a total of 1,000,000 annotated images. However, games like GTA are generally closed-source and it is difficult to control for the conditions or obtain specific images as required.

In [37], the authors developed semantic segmentation dataset that they synthetically and procedurally generate using the Unreal engine. The scenes are generated from a real-world representation of an urban environment in CARLA, with a range of variable weather and lighting conditions which the authors refer to as influence factors. This dataset contains metadata pertaining to the type of weather condition as well as the magnitude. Influence factors in the dataset include rain, puddles, cloud and angle of the sun.

2.6 De-noising and Recovering Clean Images

In the Section 2.4, we discussed how the adverse weather and lighting conditions can affect performance of Computer Vision (CV) models. The simple method to improve this performance would be to obtain sufficient quantity of images that cover all possible weather conditions that the models are likely to encounter. This is infeasible for a number of reasons.

Firstly, there can be a huge variety of possible combinations of these conditions. It is infeasible to collect a dataset that is of sufficiently high quality that will also cover all of these cases. Secondly, to train a semantic segmentation model, the images need to be finely annotated at a pixel level. This is an expensive task and datasets are not publicly available yet that sufficiently cover all of these cases [37].

The other approach to making these models work on such conditions would be to somehow filter out the effects of these weather and lighting factors in order to obtain an image that is close to the clean images the network is well-trained on.

Thus we can think of a scene under adverse weather conditions to be a simple sum of a clean image and some kind of perturbation. In the case of rain, Zhang et al. [80] decompose a rainy scene into a clean weather scene and rain droplets. This can be represented thematically as $x = y + w$, where x is a rainy image, y is a clean image, and w are the rain streaks.

We see similar approaches in literature for recovering from images affected by haze [8]. Atmospheric scattering models from [51] are often used, e.g. $I(x) = J(x)t(x) + \alpha(1 - t(x))$, where, $I(x)$ is the hazy input image, $J(x)$ is the clean image we want to recover, $t(x)$ is

the medium of transmission, α is the global atmospheric light, and x is an arbitrary pixel index in the image.

Thus, we can at least suggest there exists a function to recover a clean image from an image affected by adverse weather conditions. Same can be said for illumination. Note that we only wish to recover enough of the original image that we can improve scene understanding of such images, and do not necessarily need to recover the exact image.

In this section, we discuss some of the methods for doing so that are found in the literature.

2.6.1 Software vs Hardware

It is worth mentioning that hardware can have an important role in performance of perception for AVs. Different sensors have their own unique strengths and weaknesses. Some of the most widely used sensors in ADS include camera, Lidar and Radar. Other sensors that are also used include ultra-sonic (US) sensors and contact sensors. A common approach is to use both camera and Lidar. These two sensors can complement each other and improve perception without being burdened by size of the devices. Camera and Radar is also often used together.

Lidar works by emitting laser pulses, and the time it takes from departure to hitting an obstacle and returning back to source is monitored. It works well at night or under poor lighting as it does not require illumination from the surrounding environment. However, Lidar is adversely affected by bad weather [4]. Lidar is also quite expensive compared to camera.

Radar operates in a similar fashion by emitting waves and receiving the echo. Where Lidar typically works in infrared, Radars work with radio waves, and each frequency has different absorption and scattering properties in precipitation. Radar work well under bad weather and lighting, but offer little object identification capabilities. Radar is also more expensive than camera and moreover, it is difficult to integrate in regular cars due to the size and shape. While Google has opted to use 3D Lidar for most of its perception tasks, and use cameras in limited capacities [9], it is seen in [59] that the costs for these devices are significantly high.

Moreover, in various driving assistance technologies such as Automatic Braking Systems (ABS), lane detection and lane keeping, Tesla's Autopilot, cruise control and adaptive cruise, cameras are often used since they are cheaper. Tesla has opted to not use Lidar for their cars [27].

Criteria	Lidar	Radar	Camera	US
Very short range (0-1m) detection	Poor	Only for short range radar	Ok	Very good
Short range (1-30 m) detection	Very good	Very good	Good	Poor
Long range (30-100+ m) detection	Medium	Very good	Poor	No
Operation in adverse weather conditions	Poor	Very good	Poor	Good
Operation at night	Very good	Very good	Limited	Very good

Table 2.1: Comparison of main technologies for environment perception [32]

For this thesis, we scope our work to camera only since we want to improve semantic segmentation under adverse weather using RGB images. Even when we discuss cameras, there are different kinds of cameras with different specifications. Some lenses are better for long range, some are better for short ranges. There are powerful cameras that have high dynamic range that can capture values in a wide range. Hence, such cameras could be used for low illumination settings. Moreover, there are time of flight cameras that use their own illumination to capture scenes. Camera technology is an entirely different research domain and as such we restrict our research to only software implementations (algorithms).

2.6.2 Single-shot vs Multi-shot

In literature, there are various methods for removing rain, haze, etc. Most of these methods can be divided into two categories based on whether they use temporal information or not. Using temporal information is often advantageous and is considered an easier problem.

For removing dynamic conditions such as rain or snow, it is easier to work with multi-shot images, or video because it is possible to use the temporal information to help the denoising process. For example, it is an established fact that video based methods will generally be better than single-shot denoising techniques as rain is a spatio-temporal phenomenon and rain streaks do not persist on the same pixels in consequent frames [3].

For more static conditions such as fog or haze, or low illumination, having multiple shots isn't as big of an advantage. Nevertheless, for both cases, single-shot recovery of

clean images is considered the more difficult task [3]. For this thesis, we scope the work to only single-shot images.

In the following sections, we discuss some of the techniques in literature for denoising images with adverse weather conditions or lighting. This is not a comprehensive survey, but we discuss some popular approaches that are used for denoising rain, denoising haze or fog, and recovering daytime images from night or dark images.

2.6.3 Metrics for Measuring Performance

Here, we discuss some of the metrics that are often used for measuring the quality of models that are used to recover images.

Mean Square Error (MSE):

MSE measures the average of the squares of the errors. MSE has a wide array of applications across various domains. The equation for calculating MSE is given as follows,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2,$$

where, n is the number of instances, Y_i is the i$^{\text{th}}$ value, and $\hat{Y}$ is the average of Y.

MSE can be used to compare how similar the denoised image is to the original image.

Peak Signal to Noise Ratio (PSNR):

PSNR is the ratio between the maximum possible power of a signal, and the power of corrupting noise that affects the fidelity of its representation. The PSNR in dB is given by

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right),$$

where MAX is the maximum possible value for a pixel (which is 255 for regular RGB images), and MSE is the mean square error. PSNR is most commonly used to measure the quality of compression algorithms. For reconstructing an image, PSNR can be used but it does not take into account structural similarities.

Structural Similarity Index (SSIM):

SSIM is used for measuring the similarity between two images. The idea behind SSIM is that MSE or PSNR only estimate absolute errors. SSIM is designed to consider image degradation as perceived change in structural information. It incorporates luminance and contrast masking.

The measure between two windows x and y of common size NN is given as [73],

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

where, μ_x and μ_y are the means of x and y respectively, and σ_x^2 and σ_y^2 are the variance of x and y respectively, and σ_{xy}^2 is the covariance of x and y.

Blind Image Quality Index (BIQI):

This index provides a score on the quality of image that is not tied to ground truth. A higher BIQI score indicates lower image quality, while a low BIQI score indicates higher image quality. As this is a subjective measure, and is independent of any reference, it is difficult to make conclusions based on BIQI score alone.

2.7 Techniques for Rain De-noising

Rain de-noising has been an active area of research for a long time. We discuss some of the traditional techniques in literature for removing rain from images. We also discuss deep learning based approaches, which are our primary focus.

Sparsity-based Methods: Sparsity based methods work by treating images with rain in two separate sets of atoms where one represents raindrops assuming all of the drops are in the same orientation and the other set represents background. [35] A similar approach is to use codes for the same two sets of components against dictionary atoms, which might, however, result in noise around rain streaks. [48]

Low-rank Representation-based Methods: This method is based on the fact that the orientation and pattern of rain in images will be same, and based on that, a low rank is assigned prior to capturing the image [55], [65]. However as it works on ranked code, there is a high tendency of repetition which leads to blurry results. For solving this issue,

a convolutional coding based method is introduced where a set of learned filters are used for rain pixels. [78]

Gaussian Mixture Model-based Methods (GMM): Since GMM is capable of processing multiple patterns and orientations, patch based GMM method is introduced for modeling the rain streaks and background separately. [46]

Deep Learning Approaches: The recent success of CNNs has motivated its application to various computer vision tasks. There have been a few CNN-based models proposed for de-noising rain. The models learn a function to transform a rainy image to a clean image, often employing traditional image processing techniques to improve results obtained from the CNN.

In [20], Fu et al. propose DeRainNet, a CNN-based rain de-noising network. The authors implement their model by first decomposing an input rainy image into two parts- a background-based layer, and a detail layer. Mathematically, this can be shown as $J = J_{base} + J_{detail}$.

The authors suggest that the rain streaks in the images remain on the detail layer. The CNN is thus trained on the detail layer. The base layer is subtracted which leaves a sparse detail layer, with many pixels around zero. The training process is hence simpler, requiring less data and time with a higher convergence for the CNN. The authors report unavailability of rainy data along with ground truth data (corresponding clean images) and instead synthesize rain using Photoshop filters which they use as ground truth. The authors collect 350 clean outdoor images from the UCID dataset and BSD dataset, and used a Photoshop technique to synthesize rain on top of them to generate 14 images with different streak orientation and intensity from each clean image resulting in 4900 rainy images. From that set, the authors further selected 64×64 patches to obtain one million training samples. The authors demonstrate a model trained on synthetic rainy data on real world rainy images. The output of the CNN is added back to the base layer to get a hazy output which is passed through a function to enhance the image. The authors use SSIM as their metric for evaluation and report better performance over the existing state-of-the-art. Moreover, the authors showed that when previous techniques are used on different scenes than that of the training data, the models often perform worse. The proposed approach yields improved performance on all new scenes and hence is more generalizable. The authors also report improved BIQI scores on average across the unseen real-world data over existing approaches. In [21], the authors extended the previous network structure using Residual blocks to get improved performance.

Zhang et al [80] proposed the Image De-raining Using a Conditional Generative Adversarial Network (ID-CGAN) by modifying the architecture of Conditional Generative

Adversarial Networks (CGAN) [50]. The proposed model is specifically built to de-noise rain from rainy images. The CNN based approach only minimises the L2 error (Euclidean distance). The authors suggest that using only the L2 error will lead to poor visual quality of the de-rained images. GAN based approaches incorporate the discriminator error as well as the L2 error. The authors further introduce a perceptual loss term in the cost function. This loss term is shown to improve on the original loss function by making the training process more stable, reducing artifacts and noise and leads to overall better quality for generated images. The perceptual loss is calculated by using a pre-trained CNN to output high-level feature values for the generated de-rained images as well as the ground truth. The refined perceptual loss function includes all three loss terms and is defined as, $L_{RP} = L_E + \lambda_a L_A + \lambda_p L_{LP}$, where, L_E is the Euclidean loss, L_A is the adversarial loss, and L_P is the perceptual loss. λ_a is a weight for the L_A and λ_p is a weight for L_P.

The authors also propose using a multi-scale discriminator to capture both global and local context, so as to make the discriminator more robust by having a multi-scale receptive field. The idea has been previously used for other vision tasks such as object detection and segmentation. Here also, the authors use a Photoshop technique to synthesize rain as in [20]. The dataset consists of 700 training images from the UCID dataset and BSD-500 dataset, and 100 test images and rain is added with different intensities and orientation, and resizing all images to 256×256. The authors test their model using metrics PSNR, SSIM, UQI, VIF and report improved performance on all of them over existing approaches.

In [3], the authors bring up an obvious limitation from the rain removal papers discussed above. Due to the limitations of publicly available datasets, the approaches proposed in literature typically demonstrate the effectiveness of the models by evaluating their performance against synthetic rain on small datasets. The most commonly used metrics for measuring performance of these techniques are PSNR and SSIM. These metrics are not representative of real world applications. The authors instead propose testing these methods on real world data, and in some typical computer vision pipeline, such as, segmentation, instance segmentation, and feature tracking. The authors show that using state-of-the-art rain denoising techniques for single-shot rainy images that have good PSNR and SSIM, the models improve segmentation performance, but has reduced performance for feature tracking, and mixed results for instance segmentation.

On the other hand, for video based approaches, state-of-the-art techniques improve performance on all three evaluations. It is an established fact that video based methods will generally be better than single-shot denoising techniques as rain is a spatio-temporal phenomenon and rain streaks do not persist on the same pixels in consequent frames.

In the recent pre-print, [57], the authors propose a domain adaptation system where

data from any domain is converted to the domain the network is originally trained on. The network is assumed to be trained on clean weather, under ideal conditions. The models that are used to convert domains are called input adapters. Data from a selected adapter is fed to the computer vision network to train the domain adapters in a supervised method. The adapters are trained from generated data using a Condition-Dependent GAN that are trained to convert non-ideal input images (which may have rainy or snowy data for example), to a distribution that the network is familiar with. The authors make a point that the converted data do not necessarily have to be identical to the clean images, but that it should improve performance on segmentation or localization (or any computer vision task we would want to perform on the data). Moreover, the authors introduce online learning to their framework. New data that are not from the distribution the adapters are trained on are identified, and the authors use it to train a new set of parameters. The model does not require fine-tuning and is designed to work on off-the-shelf segmentation networks. The authors demonstrated their approach and showed improved results on the RobotCar dataset. The approach is mostly unsupervised as the pipeline the authors propose categorizes data in an unsupervised fashion using metadata such as time of day and weather conditions. This is very similar to our own method, but independently derived, and the authors do not go into implementation details or even a systematic evaluation of their methods.

2.8 Recovering Daytime Images From Nighttime

In [15], the authors propose a method for progressive adaptation of models that are trained for daytime scenes to nighttime scenes. The authors suggest that the domain discrepancy between daytime scenes and twilight scenes, and again between twilight scenes and nighttime scenes are smaller than that of daytime scenes and nighttime scenes. This lead the authors to use images taken at twilight and use them to progressively improve the network. The authors train a semantic model that is trained on labelled daytime images in a supervised fashion. The model is then used to predict semantic labels for intermediate stages and these predicted labels are assumed to be close to accurate. The predictions are then again used for fine tuning the network. Thus the daytime model is fine-tuned for nighttime scenes in an unsupervised manner. The results showed by the authors show slight improvements over a network trained on daytime scenes only. The authors also publish their dataset that they call Nighttime Driving which has 35,000 unlabelled images and 50 labelled images. The authors suggest this method can be extended to other weather and illumination conditions.

In [64], the authors propose a method called Guided Curriculum Model Adaptation (GCMA) where the authors build on [15] to use the twilight times. Moreover, the authors also suggest that images taken over different times with the same 6D camera pose will share a significant amount of content and thus useful for gradual adaptation. Finally, the authors use an image translation technique to stylize daytime data to darker domains. The authors use these three ideas to design a method to gradually adapt semantic segmentation models trained on daytime images to work on nighttime images in an unsupervised manner. The authors also propose a new metric for segmentation for data with indiscernible content. Furthermore, the authors publish the dataset they used for developing and testing their method.

2.9 Removing Haze

Haze removal is a challenging problem as well. Haze affects the image on varying degrees based on levels of haze, as well as the depth of objects in the image. Various image enhancement techniques have been applied to recover haze-free images in literature [75, 52, 49]. These methods have obtained varying levels of success. In recent years, better approaches using better priors have been proposed in literature. In [69], the authors propose a Markov Random Field (MRF) using the assumption that local contrast of the haze-free image is higher than the hazy image. This has the negative effect of producing over-saturated images. In [19], Independent Component Analysis is used to remove haze from images. Dark Channel Prior has been used in [28] to subtract the haze. These methods are computationally expensive and information is often lost due to the static transformations. In [54], factorial MRF is used to improve on previous approaches. Other approaches have also been proposed in literature to improve computation costs [24, 70].

A CNN-based approach called DehazeNet for removing haze from images is proposed in [8]. The model uses an atmospheric scattering model to recover the haze-free image as in [51]. The model uses convolution and maxout [44] layers which the authors suggest can generate almost all haze-relevant features. The authors also suggest a bilateral rectified linear unit as activation functions, which improve the quality of the recovered image. The model is shown to perform better than the existing approaches and much easier to use than traditional techniques. The authors in [79] propose a Densely Connected Pyramid Dehazing Network (DCPDN). The atmospheric scattering model is directly embedded into the network, to ensure that the model follows the physics-driven scattering model for dehazing [51]. The authors also propose an edge-preserving loss function, and used a generative adversarial network (GAN) to generate haze-free images. In the last few years,

various other deep learning based approaches have been proposed to remove haze from images.

Chapter 3

Framework

In this chapter, we discuss our key idea, the framework we propose and implement, and our methodology. We discuss the dataset that we used, and our motivation behind using synthetic data. We discuss the models used for segmentation, and motivation for choosing the particular model. Moreover, we discuss the generative adversarial network architectures used, why we choose them, and the differences and significance of the architectures. We also discuss the architecture of the classifiers we used for detection of weather in our work. We discuss in brief each of the components that we use in our framework, how we implement them, and details specific to the components and we discuss the methodology of our research.

3.1 The Approach

We propose the idea of a modular architecture for improving semantic segmentation under adverse weather and lighting conditions. Our idea can be thought of as pre-processing step during inference time before feeding the processed input to the segmentation network.

The key advantage of our method is that we do not require labelled data for segmentation for these adverse weather conditions that we would typically require if we used the conventional approach to make a perception model more robust to edge cases. Instead, we only require metadata such as the presence and levels of rain, snow, etc. Thus, we do not retrain the network with expensive labelled data, or risk changing a network that is already tested and bench-marked in order to adapt it to a new environment with adverse weather conditions. We have previously mentioned that it might be useful to have a core network

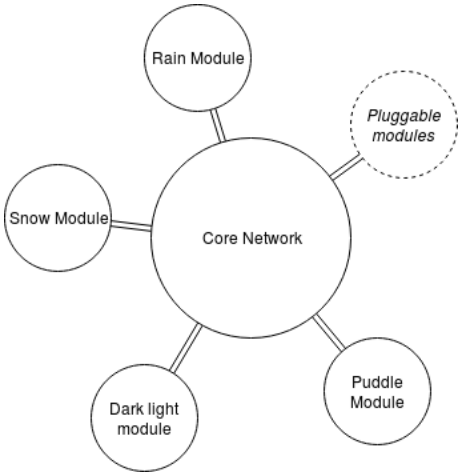


Figure 3.1: Modular Approach to improve segmentation

that we don't easily change or retrain as retraining can sometimes lead to 'forgetting'. It also allows us to quickly and easily extend a core CV network to work on varying visibility conditions.

The other benefit from our approach is that we want to be able to plug in or out these modules. This allows for a more flexible framework, where modules that are not necessary are not installed into the system. For example, a snow module is unnecessary in tropical regions. Figure 3.1 shows the idea of a several pluggable, and unpluggable components attached to a core network. In the next few sections, we discuss how we implement this idea.

3.2 Dataset

We used the model in [37] to generate data for our use. Data is generated synthetically, and procedurally following a set of grammar rules, which specify camera location, location of sun, presence and degree of influence factors etc. The procedural modeling tool uses CityEngine, with OpenStreetMap as prior input. This allows the engine to model a 3D environment based on an actual urban location in Canada. Shapefiles are also used as input from municipal open databases, which contain information such as building heights and trees. The authors used CARLA to streamline the process of dataset generation. The dataset contains RGB images, with semantic annotations, along with depth information, occlusion maps, etc.

For our purposes we use the RGB images with semantic labels only. The data points in the original work are selected in a way that they all contain road-scenes, and they face the roads to make it relevant for automotive scenes. The dataset contains a total of 8,607 images that contain scenes with different weather and lighting conditions.

Of the total 8,607 images, there are 515 images with 25% rain, 576 images with 50% rain, 522 images with 75% rain and 538 images with 100% rain. Again, there are 541, 561, 520 and 529 images with 25%, 50%, 75% and 100% cloud. For puddles, there are have 565, 491, 530, 565 images with 25%, 50%, 75% and 100% puddles, respectively.

The original images all have resolutions of 2048×1084. Our DeepLab implementation takes in crops of 513×513 pixels, so we use smaller crops to train our GANs. This is due to the fact that training a GAN network with higher resolution than this will exhaust the memory and be computationally more expensive. If the crop size for the segmentation networks is only a fraction of the original resolution, we can use smaller crops. Hence, for our purposes, we take random crops of 512×512 to create subsets of our data. In later experiments, instead of completely random crops, we decided to crop only from relevant parts of the images.

3.3 Overall Architecture

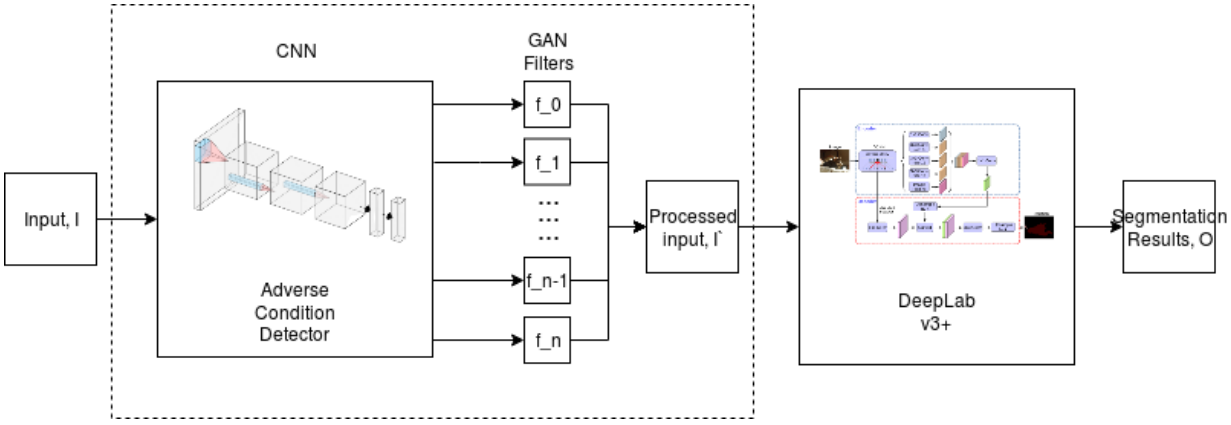


Figure 3.2: Proposed framework

The proposed framework as shown in Figure 3.2 has two main components. One is the adversity detector, and the other component is the filter. There can be any number of filters that we can add or remove on top of this framework. Thus, we call them pluggable

filters. In order to add or remove a filter, we also have to modify the adversity detector to be able to tell when to use the new filter, or to know not to use a discarded filter.

We implement the proposed network by designing and implementing the individual parts and then using them together. In the experimental results section, we show how each of the components perform, and we then show how the entire pipeline performs. Discussion on the individual components are given in the following sections. In order to demonstrate the application of the framework, we sample input uniformly at random from the entire dataset to obtain experimental results. The random input is passed through the adversity detector which gives a prediction for what condition is present in the image. Based on this prediction, we apply a filter and then process the input. If the detector finds no significant conditions that impair segmentation performance, no filters are applied. These processed inputs are collected and we run it through our DeepLab implementation. To demonstrate out framework works, it is sufficient to show that the mIoU obtained from the preprocessed images in DeepLab are higher than that without our preprocessing model.

As we mentioned earlier, our framework is meant to sit between the sensor (camera) and the segmentation network as a preprocessing step. The advantage of this is that it is extensible and highly modifiable. We discuss these components in more detail in the following sections.

3.4 Segmentation network

For semantic segmentation, we use the same DeepLab v3+ [11] implementation as [37]. DeepLab v3+ is among the state-of-the-art with a peak mIoU of 82.1% accuracy on Cityscapes semantic segmentation task. The network uses a backbone of ResNet-50 [29] that has been pretrained on the ImageNet dataset [16]. Three different DeepLab v3 checkpoints are used. One that is obtained from training only on clean weather and lighting images, one that is obtained from training on additional 3% of the adverse conditions, and one that is obtained from training on additional 10% adverse condition images.

3.5 GAN Filters

There are currently a wide variety of GAN architectures being used in research. Each of these techniques has their strengths and weaknesses. For our research, we pick two models, one that requires paired data, and one that does not. We discuss our choice of networks, and their strengths and weaknesses below.

Pix2Pix

The authors in [33] explore the use of a conditional adversarial network as a general purpose solution for image-to-image translation problems. The idea behind the work is that we can let the network learn a loss function rather than having to provide specific loss function formulations for different problems. The authors have demonstrated its flexibility in terms of application in their work. Moreover, many people have used the model for a variety of applications including art and illustrations. The authors achieved their results using conditional GANs. Previous efforts that have used image-conditional models have only done so for particular applications. The authors used their approach for all kinds of tasks. The authors also use a 'U-Net'-based architecture for the generator, and a convolutional PatchGAN as a discriminator. The authors argue that skip connections in the U-Net will improve performance because for image translation task, the generator needs to take in high resolution input and produce high resolution output, and while the input and output are designed to be different, they share an underlying structure that skip connections will help preserve. The PatchGAN in the discriminator works by only penalizing structure at the scale of patches, i.e. it tries to classify if each $N \times N$ convolution patch in an image is fake or not.

$$G^* = arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

where, G is the generator function, and D is the discriminator function.

In this work, we consider the problem of denoising an input that has reduced visibility due to adverse weather or lighting conditions as an image translation problem from a 'noisy' input to a 'clean' output. The Pix2Pix model is ideal for our purposes for a number of reasons. One is that it works reliably well without too much modification to the core architecture. It is also easy to set up requiring little oversight. Most other GAN models are very difficult to train and require heavy modification at times to make them work. Another reason for using Pix2Pix is that we can use the same model for all our filters, just with varying weights. The same model is separately trained on different conditions and the best filter is picked at inference time to preprocess the input data to force it to a distribution that the segmentation network is familiar with (clean weather images with ideal lighting).

One limitation of the Pix2Pix network is that the model requires paired data to train, which means that for every input, there has to be a mapping to a specific output. For traffic scenes, finding paired sets of data is challenging, though not impossible.

Cycle GAN

The authors in [83] proposed an image-to-image translation GAN that does not require paired training data to learn a mapping from an input image in a source domain to a target domain. This mapping can be expressed as $G : X \rightarrow Y$, where the source domain is X, and target domain is Y. This mapping is under-constrained and intractible to learn as there can be infinitely many possible mappings from X to Y. The authors couple this with an inverse mappping $F : Y \rightarrow X$ and introduce a cycle consistency loss to enforce $F(G(X)) \approx X$. The authors demonstrate their method for image-to-image translation tasks without the availability of paired data and show good results.

While the approach can sometimes yield very good results, often it does not. The authors concede this point in the original paper, where they mention that on image translation tasks where geometric changes are necessary, the learned translation degenerates into making minimal changes to the input. The authors also compare their method to that where paired training data is used and acknowledge the gap in performance, mentioning that the gap in many cases will be impossible to close. What the authors feel will lead to improved results is a way to incorporate some form of semi-supervised learning into GAN models.

3.5.1 Training and Testing the filters

In order to train each filter, we collect data for rain, puddles, clouds and low lying sun separately. We also create subsets of data for the different levels of rain, puddles, clouds, e.g. 25% Rain, 50% Rain, 75% Rain, 100% Rain. We discuss this in more detail in the following sections. The subsets are obtained by picking images relevant to the subset and then taking a random crop (with some constraints).

For each subset of data that we make, we create a training set, and a test set. For training, we pick 1500 training images, and 500 test images. We use a simple naming convention to refer to these subsets- e.g. images containing 25% rain is called *rain_25*. If a subset contains both 25% rain and 50% rain, we call the set *rain_25_50*. A set containing 25%, 50%, 75% rain is referred to as *rain_25_50_75* and so on. Similarly, for puddles or cloud, we follow the same convention where *puddles_25* refers to the set containing only 25% puddles and *cloud_25* refers to the set containing 25% clouds. For low lying sun angles, we simply refer to the only set as *lowsun*.

We train the individual filters using the training set for corresponding data subset. A filter trained on *rain_25_50* is referred to as *gan[rain_25_50]*. We train the filters on 150

epochs of the training set.

We test our filters, and eventually our framework on the test set which we assume to be in the same manifold. The filters are tested by generating denoised images by the GAN which take as input the test images. The generated set obtained from *gan[rain_25_50]* tested on *rain_25* is referred to as *[rain_25_50]_[rain_25]*. The generated set is then used to measure the mIoU in the DeepLab network. We compare the mIoU of the denoised set compared to that of the input set.

3.5.2 Normalized mIoU score

Because we are interested in how well the GAN network we train works to denoise the images, and in particular, how much better the filter works with respect to the mIoU of the adverse weather set, we define a score that calculates exactly that. We define the normalized mIoU score for a GAN as

$$nmIoU\_score_{GAN} = \frac{mIoU_{GAN} - mIoU_{unfiltered}}{mIoU_{clean} - mIoU_{unfiltered}} \tag{3.1}$$

where $mIoU_{GAN}$ refers to the mIoU for the denoised images generated by the GAN filter, $mIoU_{unfiltered}$ refers to the mIoU for the adverse weather data, and $mIoU_{clean}$ refers to the corresponding clean images

Thus, an nmIoU score of 0 means that the filter makes no positve improvement on semantic segmentation, or whatever improvements it does make is offset by the performance hit the network takes by image quality degradation and/or artifacts introduced by the GAN.

An nmIoU score of close to 1 means that the filter is able to achieve close to perfect score. A negative score means that the filter is doing more harm than good and is useless for all intents and purposes.

3.5.3 Rain Filter

In this work, we use the different variations of rain possible in ProcSy [37]. It is worth noting that the 100% rain samples from the synthetic dataset is only medium to heavy rain in terms of realistic rain. 25% to 50% rain would be light rain, and 75% to 100% would be moderate rain.

We sample randomly from the entire dataset to create subsets with 25%, 50%, 75% and 100% rain. Because it is difficult to train our generative networks with high resolution images due to memory issues, we decided to crop the samples randomly to a size that the segmentation network could also work with. For our initial experiments, we decided to create 11 sets of data for each of them.

It does not make sense to crop completely randomly for a number of reasons. For traffic scenes, useful information would generally be in the more central regions. For example, the sky will often not contain useful information and can introduce randomness in experimental results. So we made a second subset of rainy images with 25%, 50%, 75% and 100%. We also decided to have a mixture of these subsets so we have one set containing all levels of rain randomly picked, and another two sets 25% and 50%, and one that contains 75% and 100%.

Using our selected datasets, we train a series of rain filters.

3.5.4 Puddles

Puddle images also have varying levels ranging from 0% (which are basically clean images), to 25%, 50%, 75% and 100%. The puddle generation in the synthetic data is quite realistic, with real reflections on the water. 25% puddles look like small accumulation of water on roads. 50% to 75% are medium to heavy levels of accumulation. This can cause significant degradation in performance for semantic segmentation. 100% puddles have water accumulation akin to floods.

Similar to the second approach for creating a rain subset, we create subsets of the data with 25% puddles, 50% puddles, 75% puddles and 100% puddles. We also have a set that contains all levels of puddles sampled uniformly at random from each of the different subsets.

3.5.5 Cloud Filter

We use clouds of levels 25%, 50%, 75% and 100%. Varying levels of cloud have varying illumination. 0% has the highest illumination and 100% will have the lowest. Ideally, we would have also liked to work with day and night, as well as twilight hour images, but ProcSy at the moment only has varying clouds and angles of sun for changing the lighting conditions.

Similar to the approaches above, we train filters for each cloud levels and we show the improvement in mIoU.

3.5.6 Sun Angle Filter

The sun directly hitting the camera can lead to significant degradation for computer vision networks. For this filter, we create a dataset sampled and cropped randomly to include images where the sun is at an angle of 10° with the horizon. This corresponds to the 'golden hour', where the sun is just slightly above the horizon and is facing the camera. As for the vertical angles, we include all combinations.

Similar to the approaches above, we pick a subset of the data with sun at an angle of 10° with the horizon, cropped to 512×512 with crops from the bottom part of the image to include more relevant crops. The filter is trained with the subset.

3.6 Adversity Detector

We use a CNN for detecting whether a scene has a specific adverse weather or lighting condition we need to account for. We trained several state-of-the-art CNNs to be able to detect conditions ranging from 25% rain to 100% rain, 25% to 100% puddle, 25% cloud to 100% cloud, and whether the sun is at a low enough angle to adverse affect segmentation performance. We found Xception [13] to perform the best, so we stuck with this classifier.

For our initial tests, we use a simple convolutional neural network with a few convolution layers using ReLU activation functions, with dropout of 0.25 and maxpooling layers between every other convolution layer. We use categorical crossentorpy as our loss function, with an adaptive learning technique called RMSProp. It scales the learning rate so the algorithm goes through saddle point faster than most other techniques. This network worked moderately well when we only considered 3 to 4 classes (Rain 50%, Rain 100%, Puddle 50%, Puddle 100%). But on trying to scale it to all 14 classes, the accuracy dropped too much. So we opted to use the Xception network.

We used the available Keras implementation of Xception as our core network. We added our own input layer that takes in a tensor of shape $128 \times 128 \times 3$. In order to train the network, we used a nearest neighbor technique to compress the images from our crops of 512×512 to 128×128. For the output, we added a Global Average Pooling Layer at the end of the base network, and added a densely connected layer with 1024 neurons. The softmax layer with 14 nodes for our 14 output classes are added on top of the densely connected layer. For training we used data augmentation techniques such as shear, zoom and horizontal flips. The images are also normalized to a range of 0 to 1.

It is worth noting that weather conditions, as well as lighting conditions can be predicted even better if we use other priors such as time of day and weather data. Since we used

synthetic data and there are no associated priors with them other than just the meta data that we actually use for labels, there is no way to test how well the network would perform given such priors. But it is almost certain to improve prediction accuracy.

In the next chapter, we discuss implementation details, and experimental setup and give results obtained from our experiments.

Chapter 4

Experimental Results

4.1 Experimental Setup

For our experiments, we used Tensorflow and Keras. We used other Python libraries such as Numpy, Pillow, Scikit-learn, etc.

For training, we used machines with configurations having Intel Core i7, Nvidia GTX 1080Ti, 16GB RAM. We also used our training server containing 4 GPUs but similar specifications.

We design a series of experiments. We first Our experiments start with first exploring the use of GANs as denoising filters.

4.2 Dataset

We used the dataset from [37]. From the entire set of RGB images, we create subsets where we denote them in the format $< Type >\_ < Percentage/Quantifier >\_ < Percentage/Quantifier >\_...$. Thus for a subset containing only 25% rain, we refer to the set as *rainy_25*. The full list of subsets are as follows:

Rainy	Puddles	Clouds	Low Sun
rainy_25	puddles_25	cloud_25	lowsun_10
rainy_50	puddles_50	cloud_50	
rainy_75	puddles_75	cloud_75	
rainy_100	puddles_100	cloud_100	
rainy_25_50	puddles_25_50	cloud_25_50	
rainy_75_100	puddles_75_100	cloud_75_100	
	puddles_25_50_75_100		

The dataset contains training, test and validation splits with 1500 images in training, 500 in validation or test. We use the same set for test as validation, as we are not concerned with a separate validation split. Each image is obtained by sampling from the entire set with relevant properties. The images are cropped using a simple scheme where we pick a random pixel.

4.3 Segmentation Network

We used DeepLab V3+ [11] for our two semantic segmentation model. We used the same process of training as done by [37]. First, which is referred to as Model A, is trained on 8000 clean images only. The mIoU for this network for our clean data which are cropped to 12×512 pixels in a way to capture most of the street and less sky, is **0.787**.

The second, Model B, is trained on 8000 images that contain equal portions of clean, rainy, cloudy, and puddle images. Even for the set containing rainy images, the rainy image set again contains 25%, 50%, 75% and 100% rainy images. Same is true for cloud and puddles set. The mIoU for the clean images for our datasets is **0.670**.

4.4 Filters

A key component of our framework are the filters. We performed a series of experiments to first answer the question as to whether GANs can be filters in the first place. Existing work in literature has motivated us to try them as denoising techniques. We also design a method to train GANs as filters to remove adverse conditions. This is discussed in the next section. We also explore the use of completely unsupervised GANs which does not require paired data and compare it to a network that does. In this section, we discuss the experiments we performed on implementing filters. We discuss the training processes we

followed, compared our choice of GAN architecture, and then we discuss results obtained for each filter we trained.

4.4.1 Training and Testing Filters

We use the Pix2Pix network and CycleGAN as our GAN architectures. We train the filters using each data subset. The trained networks are denoted as $< GANtype > \_ < Type > \_[< Percentage/Quantifier > \_ < Percentage/Quantifier > \_...]$.

We train the filters trained on 1500 images. On a single GPU, a filter trained on a set of 1500 images with 150 epochs take about 12 hours to train. We used a batch size of 8 and used an input dimension of 512×512. We discussed earlier how data is selected and cropped.

It is difficult to measure the performance of a GAN as accuracy is not necessarily a good metric for our purposes. There are other metrics that can be used to measure similarity between the output of a trained GAN and the target image such as MSE, PSNR, or SSIM. Again, the purpose of the network is not that the output images need to be identical to the target image, but to actually improve segmentation performance. It may be the case that color information, or lighting changes may result in high MSE or PSNR, but can actually be very useful filters that improve segmentation performance.

So instead of using similarity measures, we run a series of tests with varying numbers of training images and epochs and then evaluate the mIoU on the segmentation network. This also gives us charts for training where we plot mIoU against the number of epochs. Since our purpose is to improve segmentation performance, this gives us a measure of how well our GANs work to improve semantic segmentation.

We also use a method to measure how well the GAN works with respect to the mIoU of the set it is tested on for both the clean images and the adverse weather images. We give this a score that we call normalized mIoU score. This is discussed in more details in Section 3.5.2.

4.4.2 Paired vs Unpaired data

We picked two different GAN architectures. Our motivation for these choices have been discussed in a previous chapter. In short, CycleGAN has the advantage that it does not require paired data to work, whereas Pix2Pix does.

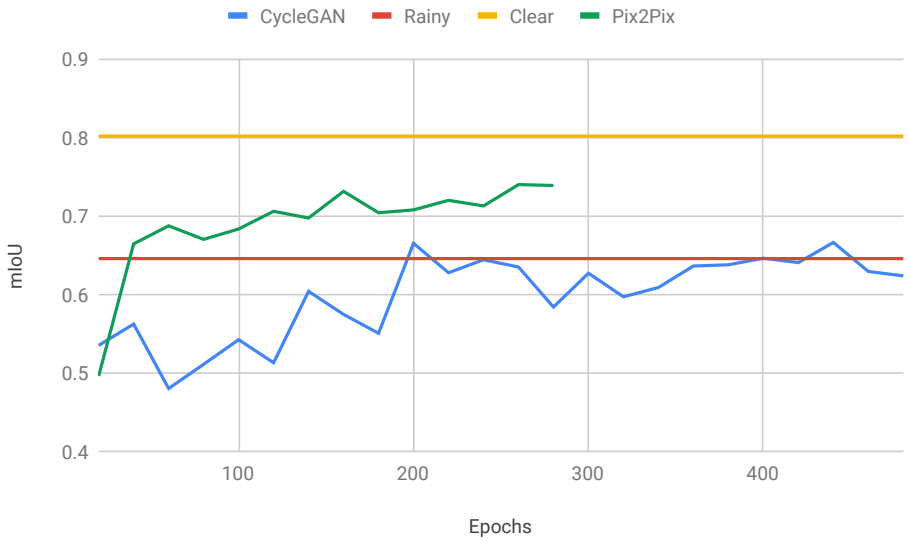


Figure 4.1: CycleGAN vs Pix2Pix - Rain 100%

We test feasibility of the two networks. For this experiment, we pick data with 100% rain. Experiments we performed show that the best performance can be shown for highest levels of any perturbation. We show these results in consecutive sections. But this point is useful for justifying our choice of data. The idea is that a denoising technique is only feasible if it works for 100% rain, as it is likely to be worse for other levels of rain.

We pick 11 sets of 500 rainy images all containing 100% rain. We calculate the mIoU for the rainy images, as well as the corresponding clean images and we pick the median set in terms of clean accuracy. We make the assumption that this set is representative of the entire dataset. For 100% rainy data, the median set has an mIoU of **0.802** for the clean images, and an mIoU of **0.6457** for the rainy images.

We train two GANs on the median set and test it against all the sets. We initially train the networks to 280 epochs. This was sufficient for Pix2Pix to get some meaningful results, but seemed inconclusive for CycleGAN. So, we further trained it to another 200 epochs.

Figure 4.1 shows the mIoU for both networks tested against the median dataset. We plot the mIoU against training epochs, where we record the mIoU every 20 epochs. The figure shows a clear improvement on baseline by Pix2Pix, whereas CycleGan struggles even after 200 additional epochs.

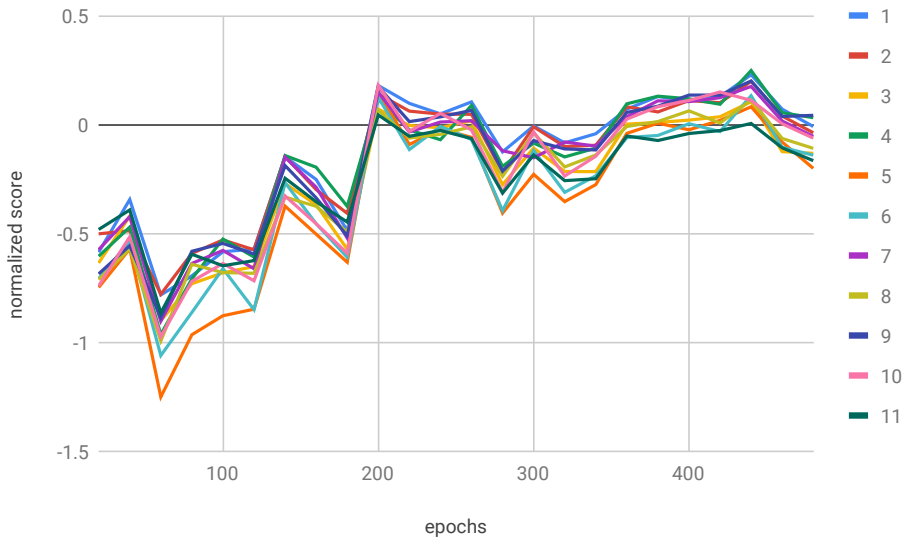


Figure 4.2: CycleGAN normalized score for all sets - Rain 100%

Figure 4.2 shows the normalized scores for all sets of rain 100% for CycleGAN plotted against training epochs. The highest score we found was a score of 0.2474 that we found on 440 epochs tested on the 4th set. Other scores are much lower in most cases, and hardly goes over the baseline score 0.

In contrast, Pix2Pix performs significantly better. Figure 4.3 shows the normalized score against training epochs for our Pix2Pix network for all 11 sets. The highest score by Pix2Pix within 280 epochs is 0.7361– close to the maximum score of 1– and the lowest score after 200 epochs is 0.2487, which is still higher than the maximum possible score we found from CycleGAN even after 480 epochs.

For this reason, we conclude that CycleGAN does not pass the feasibility test and we do not continue any further experiments with it. The authors for CycleGAN did mention that the performance of the network will never be as good as a GAN network that uses paired data. Our results are aligned with their remarks, and we also find the network to be a poor choice for a filter.

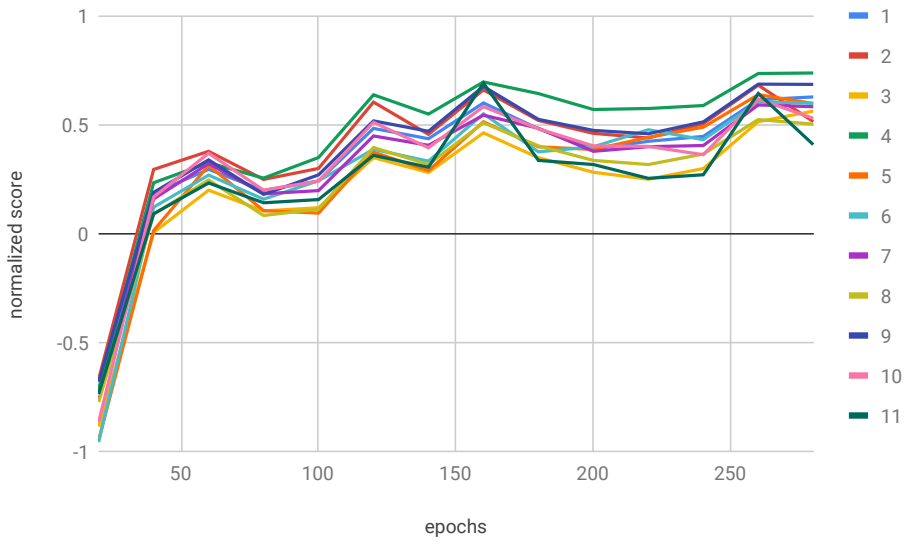


Figure 4.3: Pix2Pix normalized score for all sets - Rain 100%

4.4.3 Rain

We trained an array of rain denoising filters, using the method outlined in Section 4.4.1. In this section we discuss the results for Pix2Pix network only. Figure 4.4 shows the mIoU increase with number of training epochs for the different rain filters. The X-axis shows the number of epochs from 10 to 150, and the Y-axis denotes the mIoU.

It is worth noting that the mIoU for *rainy_25* i.e. data containing 25% rain, is **0.7950** for clean images, and **0.7743** for rainy images. This is a very close gap and difficult to improve upon. On training our GAN with a fixed network architecture and without changing our hyper-parameters much, we have an mIoU of **0.76** when we use the filter. This has the implication that the filter is doing a worse job than no filter at all. This is because GANs are known to produce artifacts sometimes. Moreover, the quality of the output images from the GAN are questionable with reduced quality in terms of sharpness, as well as random artifacts that can mess up segmentation performance.

For the set *rainy_50*, the Pix2Pix network trained on it show very marginal improvements over not using the GAN filter. The mIoU for rainy set is **0.694**, and that for corresponding clean images is **0.748**. The best we could achieve is **0.699**, which is only a fractional improvement.

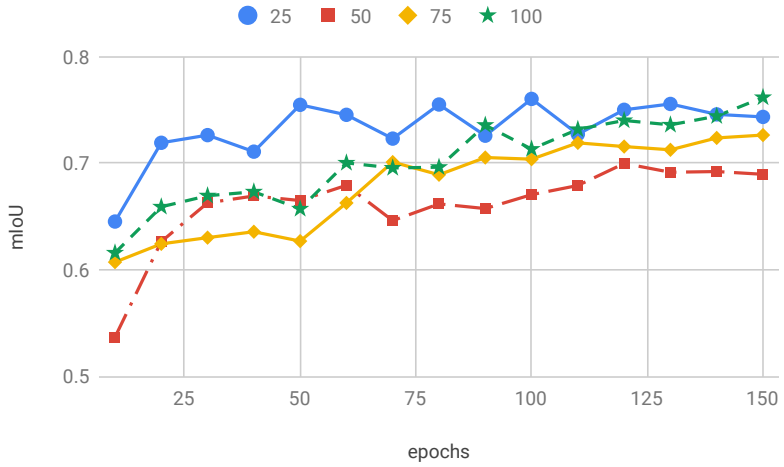


Figure 4.4: Rain: mIoU vs Epochs

We, however, have significant improvement on *rainy*_75 and *rainy*_100 using the training process we outlined for Pix2Pix. Figure 4.5 shows the normalized mIoU score for the rain filters we implemented, with the X-axis showing training epochs from 10 to 150, and the Y-axis show the normalized score.

4.4.4 Puddles

We trained a series of puddle filters following the training and evaluating technique outlined in 4.4.1. We also trained a puddle filter with all levels of puddle together. Since puddles share similar characteristics, we wondered if training a filter with all kinds of puddles would make it perform better.

Figure 4.7 shows the mIoU increase with number of training epochs. Each series in the chart represent a different GAN trained on either 25%, 50%, 75%, 100% or all of them together where we pick images from all levels of puddles uniformly at random. The different series in the chart are not directly comparable as they have varying levels of difficulty in segmentation. In Figure 4.8, the normalized scores with respect to training epochs are shown. We also observe that the series for the GAN trained with all kinds of puddles perform worse than the others.

We conducted further experiments to see if GANs would be easier to train with better performance if we could train it for varying levels of puddles individually or if we did

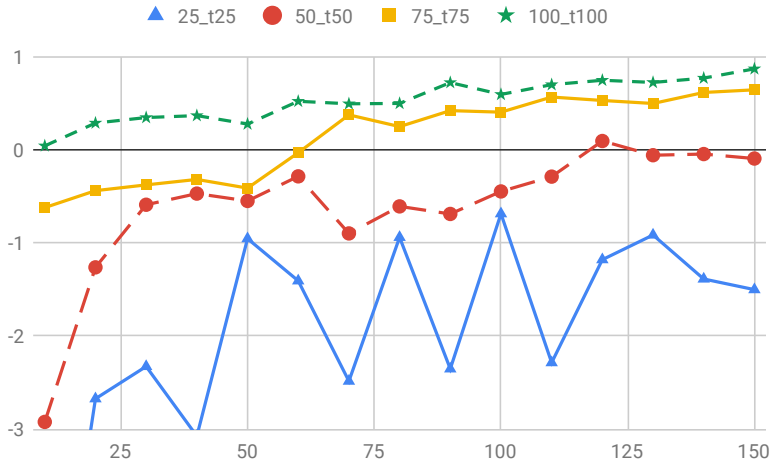


Figure 4.5: Normalized mIoU score for GAN-based Rain Filters

so together. In Figure 4.9, Figures 4.9a, 4.9b, 4.9c and 4.9d show our experiment for 25%, 50%, 75% and 100% respectively. In the figures, the Y-axis denote the normalized score, while the X-axis denote the epochs. These experiments show that generally, we do get better results from training for the different puddle levels individually than to train together.

We provide some sample images generated by the puddle filters in Figure 4.10. The top row shows the puddle images for varying levels of puddles. The row below it show the GAN generated denoised puddle images. The final row show the corresponding clean images.

4.4.5 Cloud

We trained a series of cloud filter following the training and evaluating technique outlined in 4.4.1. Clouds are the closest influence factor to vary illumination available in our dataset.

Figure 4.11 shows the mIoU increase with number of training epochs. The different series in the chart are not directly comparable as they have varying levels of difficulty in segmentation. Instead, what we are looking for are a general upward trend for the curves so we know the training is actually improving results. The relatively flat curves for 100% cloud suggest that filter requires only a few iteration to train. Figure 4.12 shows the normalized scores with respect to training epochs. This shows that the filters do improve

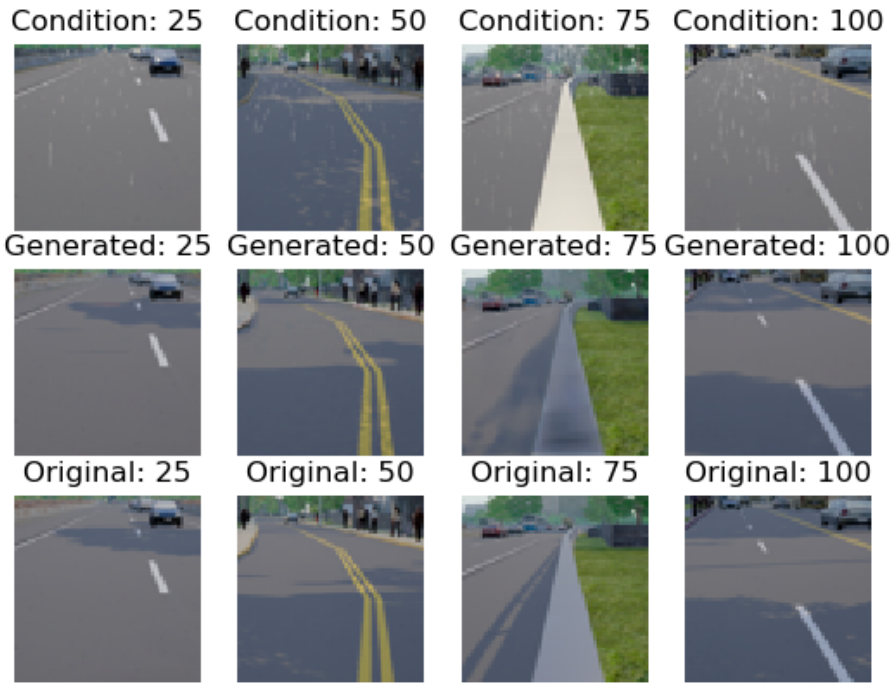


Figure 4.6: Rain denoising GAN

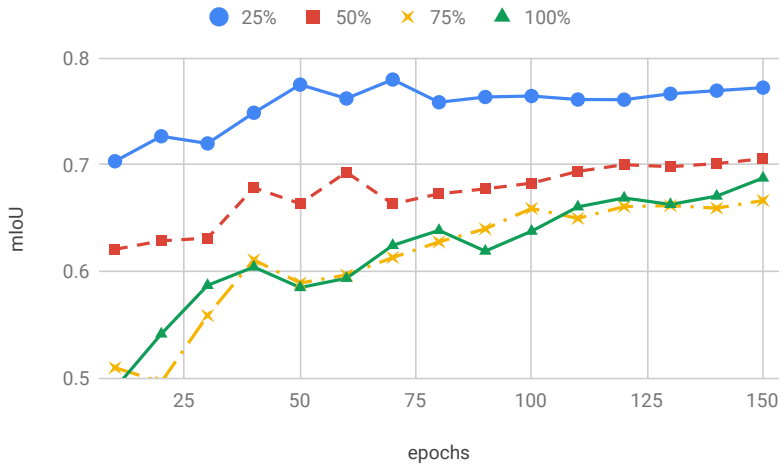


Figure 4.7: Puddle: mIoU vs Epochs

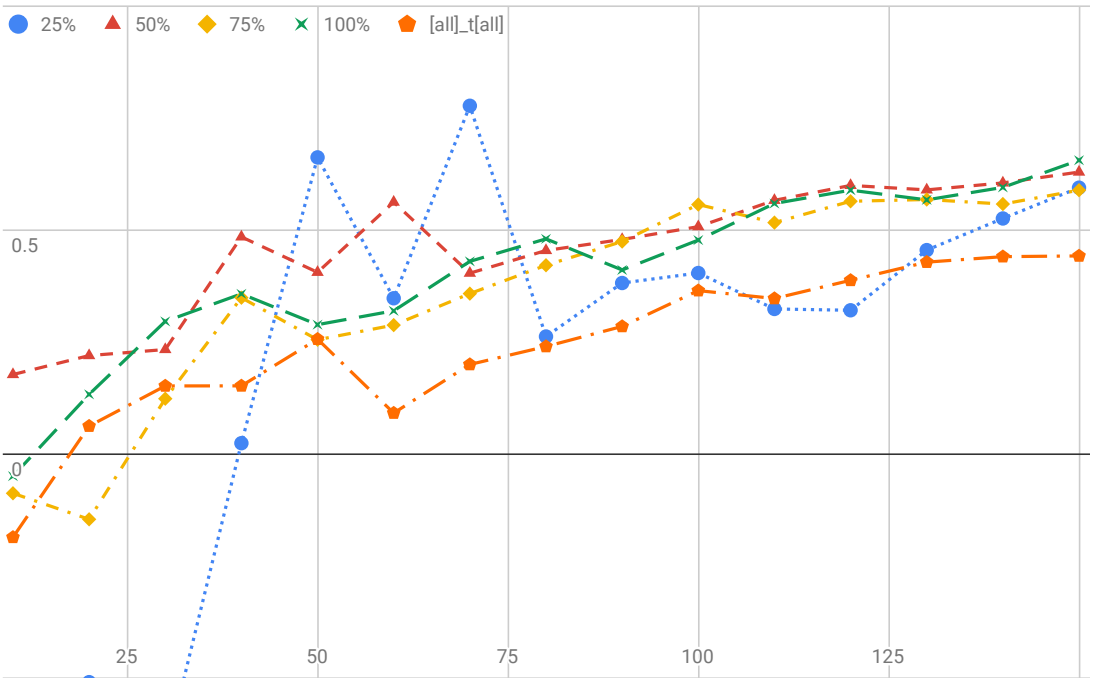
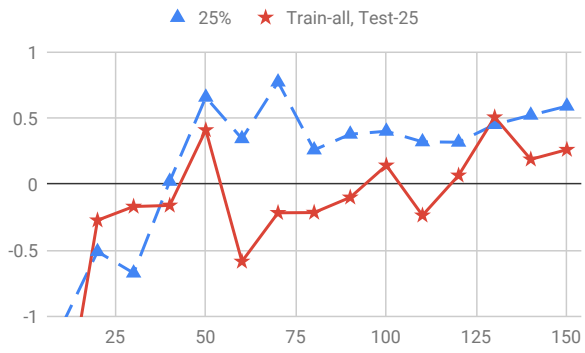
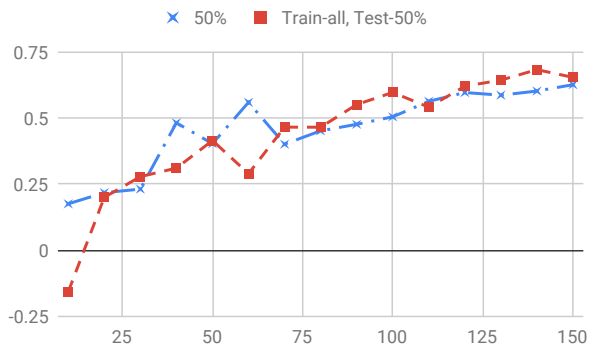


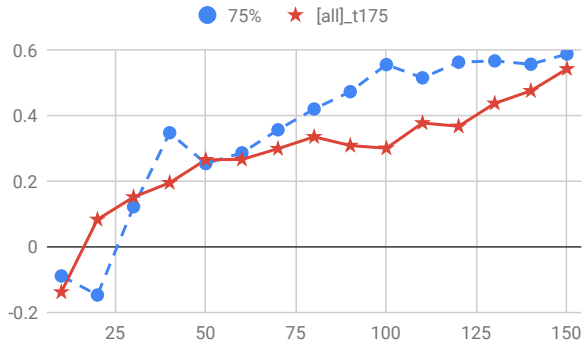
Figure 4.8: Puddle: Normalized score



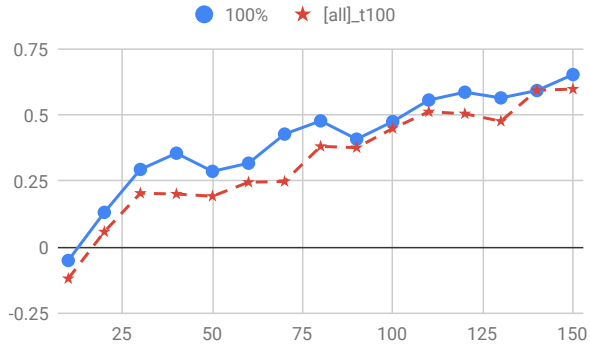
(a) Puddles 25



(b) Puddles 50



(c) Puddles 25



(d) Puddles 100

Figure 4.9: Puddles: Training with specific levels vs all levels

Figure 4.10: Puddle denoising GAN

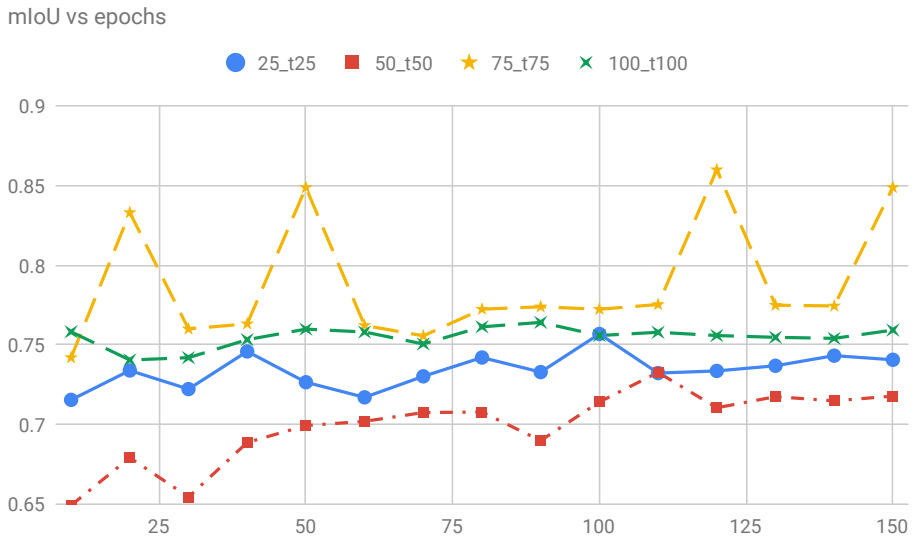


Figure 4.11: Cloud: Normalized score with respect to training epochs

performance, but it only takes the first 10 or 20 iterations to reach potentials for the 100% and 75% cloud. 25% cloud is difficult to improve on and thus the curve is consistently below the 0 score.

Cloud filters basically adjust lighting and change the distribution from dark, and overcast sky to back to the one the segmentation network is trained on. We would have liked to work with twilight to nighttime images, but our dataset does not support that at the current time.

We provide some sample images generated by the cloud filter in Figure 4.13. The top row shows the cloudy images for varying levels of cloud. The row below it show the GAN generated denoised cloudy images. The final row show the corresponding clean images.

4.4.6 Low Sun

We trained a filter to remove the effects of the sun facing the camera or hanging so low as to cast glare. The filter is trained following the training and evaluating technique outlined in Section 4.4.1.

Figure 4.14 shows the mIoU increase with number of training epochs. The bar on the top of the chart show the mIoU for corresponding images where sun is not hanging low in

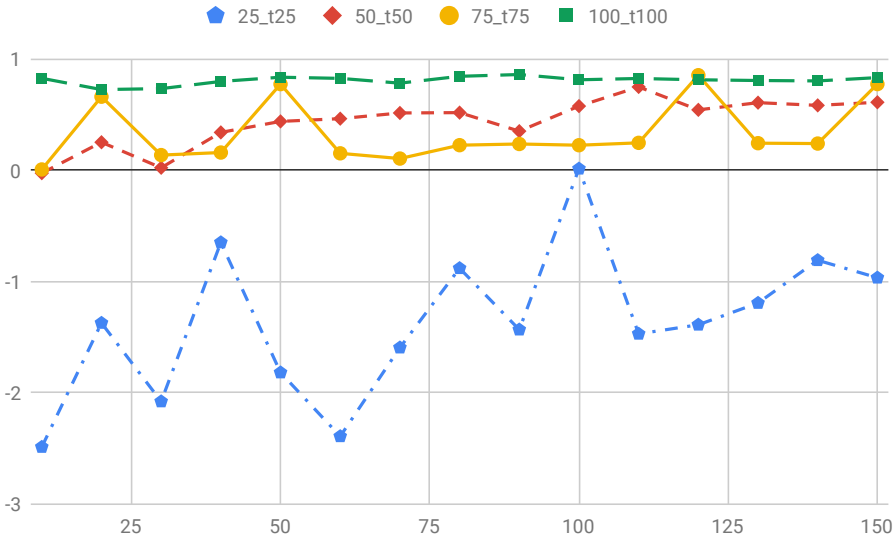


Figure 4.12: Cloud: Comparing normalized scores

the sky, and the bar at the bottom show the mIoU for data where sun is low in the sky. We can see a general trend for our curve for the filter going up. This suggests that the filter learns well. We also notice that the curve sort of starts to level off after 70 epochs. Figure 4.15 show the normalized score plotted against training epochs.

This filter also adjusts lighting and change the distribution from unusually bright light or glare back to the one the segmentation network is trained on.

We provide some sample images generated by the low sun angle filter in Figure 4.16. The top row shows the images with the sun hanging low in the sky. The row below it show the GAN generated images made to look like images with the sun overhead. The final row show the corresponding images without the sun hanging low in the sky.

4.5 Network Trained on Adverse Images

The second segmentation network, Model B, is trained on equal parts clean, rainy, and puddle images. When we test our filters on the adverse weather data, we should expect the mIoU to be much lower when we don't use filters. Our results show expected behavior.

For our experiments with the rain filter, we find the mIoU of the sets for 25%, 50%,

Figure 4.13: Cloud denoising GAN

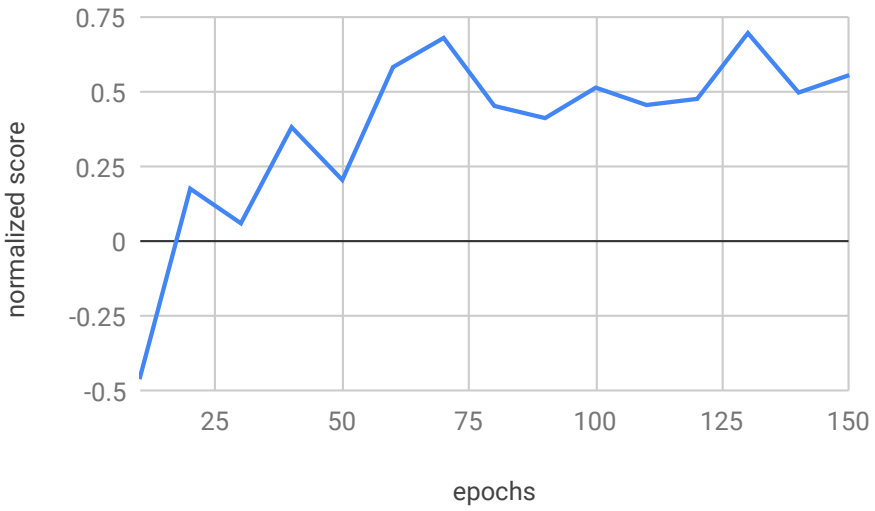


Figure 4.14: Low Sung Angle: Normalized Score

Figure 4.15: Low Sun Angle: mIoU vs training epochs

Figure 4.16: Low sun denoising GAN

	25	50	75	100
Clean	0.7072	0.6874	0.6817	0.6762
Adverse	0.7097	0.6756	0.6745	0.6594
GAN	0.6819	0.6251	0.6327	0.6594

Table 4.1: Rainy Images mIoU for Model B

	25	50	75	100
Clean	0.6566	0.6711	0.6727	0.6499
Adverse	0.6506	0.6622	0.6500	0.6205
GAN	0.6267	0.6221	0.6376	0.6231

Table 4.2: Puddle Images mIoU for Model B

75% and 100% for both clean images, and the images with the adverse conditions. We also find the mIoU for the images produced by the GAN filter. The mIoU gap between clean and rainy data is very minimal and the performance deteriorates more than it improves when we apply the filter. Table 4.1 lists down the results for rainy images for Model B.

We repeat the same experiments for the other sets as well. We only find that the performance we get from our GAN filters on 150 epochs is on par with the model trained on these adverse weather data for the highest levels, i.e. 100% rain, 100% cloud and 100% puddles. Table 4.2 shows the mIoU for the all levels of puddles tested on Model B, and Table 4.3 shows the mIoU for all levels of cloud. The GAN network in all cases is the one trained for that data distribution on the 150 epoch.

Thus, if we have the option of training a segmentation network with the data containing adverse weather conditions, it would most likely be a better alternative to using our framework. But it is worth noting that in order to do that, we need to collect finely annotated adverse weather data that may be infeasible in reality for many of the conditions we must

	25	50	75	100
Clean	0.6574	0.6541	0.6628	0.6721
Adverse	0.6592	0.6477	0.6538	0.6406
GAN	0.6323	0.6253	0.6424	0.6414

Table 4.3: Cloud Images mIoU for Model B

train our networks to be robust to.

4.6 Adversity Detector

The other major component of our proposed framework is the adversity detector.

4.6.1 The Classifier

We trained a series of classifiers for 14 classes we defined which include clean weather, four kinds of rainy data, four kinds of puddle images, four kinds of cloud, and low sun angle images. The classifier is trained with roughly 500 instances from each category. We set aside 50 images from each class for test dataset.

Initially, we experimented with a simple CNN much like the structure in [39] but found that while it works for low number of classes (accuracy of 90% for 4 classes), when we increase the number of classes to 14, accuracy goes down to about 60%.

We experimented with InceptionV3 [67] as well but the highest accuracy we obtained from our experimenting was with Xception [13] with an accuracy of **93%**. We picked random points from the training split in all the sets listed in 4.2 to create our training set. We followed the same procedure but selected points from the test split to create our test set. To train our network, we used a batch size of 32, the RMS Prop optimizer, and categorical cross-entropy as our loss function. It took roughly 10 hours to train the network. We used the Xception classifier as our Adversity Detector.

4.6.2 Combined Results

Using the adversity detector and the GAN filters we trained earlier, we implemented our proposed framework where we take an input image, pass it through the Adversity Detector, and based on the classification result, we pick a corresponding GAN, or do not pick a GAN at all if the Detector finds no presence of adverse weather conditions. We ran 11 trials as the process of picking data is random and we make the reasonable assumption that 11 trials give us a good idea of how well our framework performs. Table 4.4 gives the mIoU for when we use our framework, and when we do not. We see a noticeable improvement on Model A for mixed weather data using our approach.

Trial	Our Framework	Original
1	0.6826	0.5818
2	0.8022	0.5989
3	0.6756	0.6214
4	0.6969	0.6134
5	0.6925	0.6205
6	0.7117	0.5929
7	0.6838	0.5866
8	0.6881	0.5982
9	0.7054	0.5816
10	0.7199	0.5916
11	0.7235	0.6462
Average	0.7075	0.6030

Table 4.4: mIoU for Our Framework on adverse weather data

We note a 17.3% increase in mIoU using our proposed approach. It is also worth noting that while the Adversity Detector has an accuracy of **93.3%**, and not a perfect 100%, this still does not negatively affect performance for the framework.

Chapter 5

Discussion and Conclusion

This chapter will conclude the thesis with some final discussion, particularly with respect to the results from the previous chapter, and also some notes we want to make regarding our approach. We also outline limitations of our work, and address scope or potential for future work on the subject.

5.1 Discussion

Our experiments in the previous chapter shows that, while it would be vey useful to have a filter that does not require paired data, the state-of-the-art is not currently good enough to be used as filters for our purposes. For the experiments we conducted, CycleGAN only improves over rainy conditions for 100% rain by a small fraction after training for much longer. This improvement is also not consistent and in any way useful. The authors in [83] remarks that their method does not compare to methods like Pix2Pix or other GAN architectures that require paired data, so our results are not surprising. Once we establish CycleGAN is not viable for our work, we restrict ourselves to Pix2Pix, and conduct future experiments with it.

Better GANs

Generative Adversarial Networks have been one of the most exciting areas of machine learning research in the past five years, since Ian Goodfellow first proposed it in [25]. But despite the great results reported in various papers since, GANs have been known to be

difficult to train. The Pix2Pix architecture addresses a lot of these issues and it is shown to work for a wide variety of tasks without modification to loss function. However, we can expect the technology to make further leaps in coming years. Our proposed method offers a way to improve segmentation performance using unlabelled data, and as such, we only want to show viability of GANs as filters. It is worth noting that the GAN filters can be replaced with other filters that might work better, but can still work with our method. Other filters may include Variational Autoencoders (VAE), or other deep learning networks, or newer and better GAN architectures.

Related Work

There have been a few works in literature in the past two years that use GANs as denoising filters to remove rain [80], [21], [20], or to recover daylight images from nighttime [15]. These methods have been shown to be somewhat effective, but their results are not directly comparable for a number of reasons.

Each paper uses their own dataset, and their own method for evaluation of their technique. If we were to use the same data the authors in the aforementioned papers did, we would only be working with either rainy images, or dark images. There are some other papers on haze removal as well. However, we wanted to test our method for a number of varying weather and lighting conditions. This motivated us to use the dataset in [37]. It is also worth noting that most approaches in literature only used their method to remove rain or haze, and evaluated their approach using metrics like SSIM. We, on the other hand, use these denoising techniques to improve segmentation performance. [15] used their GAN architecture to recover daylight images, and showed improvement on mIoU, but their improvements are very minimal. Our approach is also not in competition with the methods discussed in those papers, and in fact, they can be used together. We can use the best rain removal technique in literature and use it in place of the stock GAN architecture we used for this thesis. The work by [57] is the most similar to our work. It is being developed concurrently to ours and it would be interesting to see how their work progresses.

5.2 Limitations and Future Work

Due to various constraints, we had to scope our work. This research works as a proof of concept, and our framework and methodology need to be tested on real data. This is no small task as collecting paired data with varying weather conditions would require placing

cameras at different locations and leaving them for an extended period of time. This would be a significant undertaking but it would a be a valuable contribution if it were possible.

For our synthetic dataset, we only had the option to work with rain, puddles, cloud and low sun angles. Other adverse weather conditions such as snow, or haze should be equally possible to test with out methodology, but they have not been available yet, while we have been writing this thesis. We also have limited options in terms of varying illumination. One very important experiment we wanted to perform was recovering daylight images from nighttime, but the dataset does not provide them and we found no alternative that has nighttime images with corresponding daylight images as well as rain and other conditions.

5.3 Conclusion

To summarize, this thesis explores the use of Generative Adversarial Networks in order to improve perception in adverse weather and lighting conditions for self-driving cars. For our research, we choose semantic segmentation as the perception task, primarily because it is the more difficult task, but also because we get a pixel-wise metric and hence we can judge how well the approach works for each image.

In this work, we design a framework as well as a methodology to evaluate the proposed approach. The framework consists of an Adversity Detector, and a series of denoising filters. The Adversity Detector is an image classifier that takes as input clear weather or adverse weather scenes, and attempts to predict whether the given image contains rain, or puddles, or other conditions that can adversely affect semantic segmentation. The filters are denoising generative adversarial networks that are trained to remove the adverse conditions from images in order to translate the image to a domain the segmentation network has been trained on, i.e. clear weather images. The methodology we devise for evaluating our approach uses the trained filters to output sets of images that we can then run segmentation tasks on. This, we argue, is a better metric for evaluating the GANs than similarity measures such as SSIM.

We train two kinds of GANs, one that uses paired data, and one that does not. We have concluded that GAN architectures that do not use paired data are not sufficiently good models for denoising. For our method, such a model fails to improve on the baseline and is hence of no use. We train the denoising filters using the other architecture and we found them easy to train and show good results. While these filters do not show better performance than when we train our segmentation network with adverse weather data, we make the point that training the segmentation network requires labelled data which is

expensive to collect and annotate. We implement our proposed framework and report a 17% increase in performance in segmentation.

References

[1] James M Anderson, Kalra Nidhi, Karlyn D Stanley, Paul Sorensen, Constantine Samaras, and Oluwatobi A Oluwatola. *Autonomous vehicle technology: A guide for policymakers*. Rand Corporation, 2014.

[2] Matt Angus, Mohamed ElBalkini, Samin Khan, Ali Harakeh, Oles Andrienko, Cody Reading, Steven Waslander, and Krzysztof Czarnecki. Unlimited road-scene synthetic annotation (ursa) dataset. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 985–992. IEEE, 2018.

[3] Chris H Bahnsen and Thomas B Moeslund. Rain removal in traffic surveillance: Does it matter? *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[4] Mario Bijelic, Tobias Gruber, and Werner Ritter. A benchmark for lidar sensors in fog: Is detection breaking down? In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 760–767. IEEE, 2018.

[5] Mario Bijelic, Tobias Gruber, and Werner Ritter. Benchmarking image sensors under adverse weather conditions for autonomous driving. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1773–1779. IEEE, 2018.

[6] Mario Bijelic, Paula Kysela, Tobias Gruber, Werner Ritter, and Klaus Dietmayer. Recovering the unseen: Benchmarking the generalization of enhancement methods to real world data in heavy fog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 11–21, 2019.

[7] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *European conference on computer vision*, pages 44–57. Springer, 2008.

[8] Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. Dehazenet: An end-to-end system for single image haze removal. *IEEE Transactions on Image Processing*, 25(11):5187–5198, 2016.

[9] Chatterjee. Self-driving car pushes sensor technology. *EDN Magazine*, Dec 2012.

[10] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[11] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[12] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[13] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[14] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[15] Dengxin Dai and Luc Van Gool. Dark model adaptation: Semantic image segmentation from daytime to nighttime. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3819–3824. IEEE, 2018.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[17] Ernst D Dickmanns. The development of machine vision for road vehicles in the last decade. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 1, pages 268–281. IEEE, 2002.

[18] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[19] Raanan Fattal. Single image dehazing. *ACM transactions on graphics (TOG)*, 27(3):72, 2008.

[20] Xueyang Fu, Jiabin Huang, Xinghao Ding, Yinghao Liao, and John Paisley. Clearing the skies: A deep network architecture for single-image rain removal. *IEEE Transactions on Image Processing*, 26(6):2944–2956, 2017.

[21] Xueyang Fu, Jiabin Huang, Delu Zeng, Yue Huang, Xinghao Ding, and John Paisley. Removing rain from single images via a deep detail network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3855–3863, 2017.

[22] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016.

[23] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[24] Kristofor B Gibson, Dung T Vo, and Truong Q Nguyen. An investigation of dehazing effects on image and video coding. *IEEE transactions on image processing*, 21(2):662–673, 2011.

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[26] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[27] Andrew J. Hawkins. Elon musk still doesnt think lidar is necessary for fully driverless cars. *The Verge*, Feb 2018.

[28] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2341–2353, 2010.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[30] Simon Hecker, Dengxin Dai, and Luc Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 435–453, 2018.

[31] Daniel Hernandez-Juarez, Lukas Schneider, Antonio Espinosa, David Vázquez, Antonio M López, Uwe Franke, Marc Pollefeys, and Juan C Moure. Slanted stixels: Representing san francisco's steepest streets. *arXiv preprint arXiv:1707.05397*, 2017.

[32] Constantin Ilas. Electronic sensing technologies for autonomous ground vehicles: A review. In *2013 8TH INTERNATIONAL SYMPOSIUM ON ADVANCED TOPICS IN ELECTRICAL ENGINEERING (ATEE)*, pages 1–6. IEEE, 2013.

[33] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[34] Tamra Johnson. Americans spend an average of 17,600 minutes driving each year. *NewsRoom*.

[35] Li-Wei Kang, Chia-Wen Lin, and Yu-Hsiang Fu. Automatic single-image-based rain streaks removal via image decomposition. *IEEE Transactions on Image Processing*, 21(4):1742–1755, 2011.

[36] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[37] Samin Khan, Buu Phan, Rick Salay, and Krzysztof Czarnecki. Procsy: Procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 88–96, 2019.

[38] Donald Knuth. *The TEXbook*. Addison-Wesley, Reading, Massachusetts, 1986.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[40] Leslie Lamport. *LaTeX — A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

[41] Fahad Lateef and Yassine Ruichek. Survey on semantic segmentation using deep learning techniques. *Neurocomputing*, 338:321–348, 2019.

[42] Yann LeCun. Quora: Gan.

[43] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[45] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[46] Yu Li, Robby T Tan, Xiaojie Guo, Jiangbo Lu, and Michael S Brown. Rain streak removal using layer priors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2736–2744, 2016.

[47] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[48] Yu Luo, Yong Xu, and Hui Ji. Removing rain from a single image via discriminative sparse coding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3397–3405, 2015.

[49] JE McDonald. The saturation adjustment in numerical modelling of fog. *Journal of the Atmospheric Sciences*, 20(5):476–478, 1963.

[50] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[51] Srinivasa G Narasimhan and Shree K Nayar. Removing weather effects from monochrome images. In *IEEE computer society conference on computer vision and pattern recognition*, volume 2, pages II–186. IEEE Computer Society; 1999, 2001.

[52] Srinivasa G Narasimhan and Shree K Nayar. Contrast restoration of weather degraded images. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):713–724, 2003.

[53] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4990–4999, 2017.

[54] Ko Nishino, Louis Kratz, and Stephen Lombardi. Bayesian defogging. *International journal of computer vision*, 98(3):263–278, 2012.

[55] Shunsuke Ono, Takamichi Miyata, and Isao Yamada. Cartoon-texture image decomposition using blockwise low-rank texture characterization. *IEEE Transactions on Image Processing*, 23(3):1128–1142, 2014.

[56] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[57] Horia Porav, Tom Bruls, and Paul Newman. Don't worry about the weather: Unsupervised condition-dependent domain adaptation. *arXiv preprint arXiv:1907.11004*, 2019.

[58] Horia Porav, Tom Bruls, and Paul Newman. Don't worry about the weather: Unsupervised condition-dependent domain adaptation. *CoRR*, abs/1907.11004, 2019.

[59] Priddle and Woodyard. Google discloses costs of its driverless car tests. *USA Today*, Jun 2012.

[60] Stephan R Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2213–2222, 2017.

[61] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016.

[62] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.

[63] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[64] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Semantic nighttime image segmentation with synthetic stylized data, gradual adaptation and uncertainty-aware evaluation. *arXiv preprint arXiv:1901.05946*, 2019.

[65] Hayden Schaeffer and Stanley Osher. A low patch-rank interpretation of texture. *SIAM Journal on Imaging Sciences*, 6(1):226–262, 2013.

[66] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[67] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[68] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[69] Robby T Tan. Visibility in bad weather from a single image. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[70] Jean-Philippe Tarel and Nicolas Hautiere. Fast visibility restoration from a single color or gray level image. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2201–2208. IEEE, 2009.

[71] Charles Thorpe, Martial H Hebert, Takeo Kanade, and Steven A Shafer. Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, 1988.

[72] Frederick Tung, Jianhui Chen, Lili Meng, and James J Little. The raincouver scene parsing benchmark for self-driving in adverse weather and at night. *IEEE Robotics and Automation Letters*, 2(4):2188–2193, 2017.

[73] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[74] Junqing Wei, Jarrod M Snider, Junsung Kim, John M Dolan, Raj Rajkumar, and Bakhtiar Litkouhi. Towards a viable autonomous driving research platform. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 763–770. IEEE, 2013.

[75] Zhiyuan Xu, Xiaoming Liu, and Na Ji. Fog removal from color images using contrast limited adaptive histogram equalization. In *2009 2nd International Congress on Image and Signal Processing*, pages 1–5. IEEE, 2009.

[76] Tao Yang, Yan Wu, Junqiao Zhao, and Linting Guan. Semantic segmentation via highly fused convolutional network with multiple soft cost functions. *Cognitive Systems Research*, 53:20–30, 2019.

[77] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

[78] He Zhang and Vishal M Patel. Convolutional sparse and low-rank coding-based rain streak removal. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1259–1267. IEEE, 2017.

[79] He Zhang and Vishal M Patel. Densely connected pyramid dehazing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3194–3203, 2018.

[80] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[81] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[82] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.

[83] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.