Identifying the Performance of Pedestrian Detection on Hazy Images Cleaned Using Cycle GAN and Conditional GAN

Abstract

This study aims to answer the research question: What is the most effective GAN structure for improving pedestrian detection accuracy by removing haze from an image? The findings demonstrate a significant improvement in average precision for pedestrian detection, with the Conditional GAN achieving an average precision of 0.85, while the Cycle GAN reached 0.83, both outperforming unprocessed images at 0.53. The Conditional GAN also displayed superior processing speed at 51.32 frames per second compared to the Cycle GAN's 24.96 frames per second, suggesting its greater suitability for real-time applications in autonomous vehicles.

Despite these improvements, challenges such as color distortion in the Cycle GAN and sensitivity to fog conditions in the Conditional GAN were identified, indicating that neither model consistently produces high-quality dehazed images. The study highlights critical gaps in existing research, emphasizing the need for training on diverse datasets reflective of real-world conditions. Overall, this research contributes to the understanding of GAN-based methods for improving pedestrian detection in challenging weather conditions, highlighting their potential for deployment in autonomous vehicle systems.

1. Introduction

1.1. Significance of Research

With the rise of autonomous vehicles taking to the roads, there has been the challenge of building trust with the general public. Leading concerns include safety, liability, and performance in more challenging environments such as construction zones or extreme weather conditions (Kim et. al, 2022). With concerns of safety always being one of the leading priorities and concerns, it is worth acknowledging the high level of danger currently involved with human operated vehicles on the road. The 2023 Global Status Report on Road Safety (World Health Organization, 2023) states that there were 1.19 million road traffic deaths in 2021, 21% of which are attributed to pedestrians. Pedestrians are a vulnerable group of road users, having no protection from any impact leading to a higher chance of fatalities.

This is an area where autonomous vehicles are expected to provide significant improvement by reducing the number of overall collisions (Parseh & Asplund, 2022) through faster and more consistent reactions when compared to human drivers. Despite the expected reduction in collisions, autonomous vehicles face a challenge in public perception for any fatality. An example of this can be seen after the first pedestrian fatality in 2018 where most public perceptions of autonomous vehicles recovered to normal levels a month after the incident, but the public perception of the safety of using autonomous vehicles never recovered to pre-fatality levels throughout the full duration of the study (Tapiro et. al, 2022). While it is unlikely to achieve a level of technology that completely eliminates pedestrian fatalities, this emphasizes the importance of ensuring consistently high levels of performance in all environments to minimize the number of pedestrian fatalities, as anything less could result in a loss of faith in this technology.

1.2. Research Problem

Autonomous vehicles driving in severe weather is one of the most challenging environments, with significantly worse performance and less consistency in detecting pedestrians. Severe weather greatly limits the ability of an autonomous vehicle to detect any type of object by providing some obstruction for the vehicle's sensors, having a negative impact on the safety features used to prevent pedestrian collisions (Ogunrinde & Bernadin, 2021). The ability to effectively detect a pedestrian in hazy conditions is important in improving the safety of autonomous vehicles (Li et. Al, 2020). RGB cameras are not able to see far through heavy fog or haze, similar to a human trying to look through the fog. Other sensors such as Lidar are also found to be less reliable in these conditions (Pao et. al, 2024), leaving reliability of these systems in severe weather a real challenge.

While there have been some attempts to improve accuracy by considering data from multiple sensors at a time, this dissertation focuses on how the output from an RGB camera can be improved to better detect any pedestrians in frame during a heavy fog or haze. Generative Adversarial Networks (GAN's) are a popular method of cleaning RGB images, in which a generator is trained to create an image in the target style, and a discriminator is trained to compare the newly generated image to the expected output and predict which one is real, and which one is fake. This was first proposed by Goodfellow et. Al (2014), using game theory as a way to train the model to generate accurate information. GAN's have been expanded in more recent research so that they can be used for more complicated tasks, such as removing the fog from an image and output the same image with clear weather conditions.

One stream research uses conditional GAN's and a paired dataset to generate images in the target domain based on the input image. Other studies use a cycle GAN in which unpaired data is used to train the generation of an image for a certain target domain based on the domain of the input image, and from the target domain back to the original domain again. While both of these methods have shown promising results in recent years, few studies have been conducted to directly compare the two methods and determine which is superior for the task of better detecting pedestrians in extreme weather conditions. Given that conditional GAN's use paired data I would expect them to have an advantage over a cycle GAN in giving more accurate predictions. However, due to the lack of true paired datasets for clear and foggy road images, the cycle GAN is able to utilize real images for both the foggy and clear datasets, while the conditional GAN must use a synthetically created paired dataset.

1.4. Research Question

The purpose of this dissertation is to answer the research question: What is the most effective GAN structure for improving pedestrian detection accuracy by removing haze from an image? To achieve this, we will design and build both of these models, so that their performance in both average precision and speed can be directly compared to each other in order to determine which is better suited to removing fog or haze from an image to better detect pedestrians.

1.5. Aims and Objectives

To answer the research question in this dissertation, there are three main objectives:

1- Develop a cycle GAN and conditional GAN to remove fog from an image.
2- Achieve an average precision on the test dataset greater than the 0.86 recorded by the original authors (Li et. Al, 2020) for both the conditional GAN and cycle GAN.
3- Maintain a speed of at least 20 frames per second to show that both the cycle GAN and conditional GAN can be used in real time applications for autonomous vehicles.

By aiming for these targets, we will highlight the strengths and limitations of each model for dehazing images for pedestrian detection.

1.6. Dissertation Structure

Following the introduction, we will move into a literature review for section 2. This section will discuss other research related to the research problem, in order to identify gaps in the literature which this project can address. In section 3 we will discuss the planned methodology of the project. This will include introducing the datasets, training methods, model architectures, and testing methods for both the conditional and cycle GANs. Section 4 covers the results of the project, highlighting both the results of following the planned methodology, and discussing any significant alterations required to produce quality results. A detailed discussion takes place in section 5. The discussion centers around an explanation of the results, justification for decisions, and linking the project back to the literature review to discuss how this project was successful in addressing the gap in literature, and what further gaps this brings to light that can be addressed. Finally, we get to the conclusion in section 6, where we summarize the key findings, discuss the impact of the study, the study's limitations, and finally discuss future work to build on this study.

2. Related Work

2.1. Choice of Related Literature

Computer vision is a rapidly evolving field, and to ensure we are comparing this project to studies that are still relevant we are considering studies completed within the last 5 years. This is to ensure we are identifying gaps in current literature related to the research question that are still relevant today. There are not a large number of studies specifically dedicated to de-fogging or de-hazing with a cycle GAN or conditional GAN, so the search was expanded to include studies that use these methods to remove other types of severe weather (such as rain or snow) from an image. While these studies will have some different challenges when compared to de-hazing such as handling water drops or steaks on a camera lens in the rain, they will have similar challenges in translating the image from a severe weather domain where visibility is limited into a clear domain where pedestrians and other objects are clear and visible. The search expanded on performing pedestrian detection in these conditions to include any type of object detection, as it's expected that they will face similar challenges and obstacles. Finally other studies focusing on the quality of images de-hazed with cycle GAN or conditional GAN were included to better understand the challenges in using these models for translating from a domain with hazy weather to a clear weather domain.

2.2. Conditional GAN's

Conditional GAN's as first discussed by Mirza and Osindero (2014) relies on a paired dataset, in which a single generator is trained to generate images in the target domain based on the input image, and a single discriminator is trained to compare the newly generate image to the paired image in the target domain, and guess which one is real, and which one is fake. The generator is trained to maximize the probability of the generator making a mistake, which is at the point that the discriminator can no longer tell the difference between the fake and real images.

2.3. Dehazing With a Conditional GAN

Siddiqua et. Al (2023) designed an all-in-one image restoration system to remove multiple different weather conditions from an image. This method used a multidomain, attention based Conditional GAN, which took an input image with any domain (such as rain, fog, or snow), and generated a clear image for the output. They found that this method was successfully able to remove haze while maintaining similar contrast to the ground truth, and eliminated any distortion caused by fog. Zhang (2023) used a conditional GAN to defog images in order to detect traffic signs in foggy weather. They found that this method achieved a higher peak signal to noise ratio and the structural similarity index measure compared to other methods including a cycle GAN, while they also found a much higher average precision with this method.

Other recent studies have gone further to show how a conditional GAN can improve the performance of object detection. Razzok et. Al (2023) used a conditional GAN to remove rain from images to perform pedestrian detection. They achieved an average precision of 46% with this method, compared to 26.5% without using a conditional GAN to clean the images. They also found that using traditional noise removing algorithms such as a bilateral filter did not result in any improvement for detecting pedestrians in rainy images. In the challenge of defogging images, Chow et al (2023) modified the conditional GAN base with a non-linear activation function and convolutional blocks with progressively increasing layers. They found that this resulted in a higher accuracy of object detection than other deep learning methods, and an improvement on the base conditional GAN.

2.4. Limitations of Dehazing Studies Using Conditional GANs

While other research such as Siddiqua et. al (2023) and Chow et. al (2023) find that a conditional GAN provides much better performance than other methods, we need to consider the metrics being used to measure success and the dataset being used. These studies both used paired datasets for both training and testing their model, measuring success by calculating the peak signal to noise ratio, structural similarity index measure, and root mean squared error. These metrics measure how similar the output image is to the ground truth, requiring a paired dataset to calculate. While they both found positive results through this method, it is important to note that the tests were not performed on real images with extreme weather. The paired datasets were gathered from online sources, and they do not specify how the paired data was created, leaving uncertainty on the variability of the haze for both training and testing. The studies also did not apply any pedestrian detection, while the conditional GAN gave dehazing results closer to the ground truth it is not studied how much this would improve pedestrian detection on the dataset.

In the study done by Razzok et. al (2023) they addressed the gap of applying pedestrian detection to the cleaned images, however this study focused on de-raining instead of dehazing. While there are expected similarities between the two, similar work should be done to determine if pedestrian detection is equally improved by conditional GANs for de-hazing is it does for de-raining. Similar to the other studies on conditional GANs, the testing was done using a paired dataset with synthetic weather conditions added in. Due to this limitation the results do not provide an accurate assessment of how the system would perform in real weather conditions, as the model could be simply learning the pattern of the synthetic rain.

The results shared by Zhang (2023) find the conditional GAN to be well suited to dehazing images for road sign detection. With a high peak signal to noise ratio and the structural similarity index they found that the defogged images were similar to the ground truth, and by measuring the average precision they found this defogging method to be useful in detecting traffic signs. Similar to other studies they are limited by training and testing on synthetic data. While some sample images were shown of the model performing on real-world data, the metrics were all measured on tests with the synthetic dataset, leaving uncertainty in the level of performance actually achieved on real-world data.

2.5. Cycle GANs

Zhu et. Al (2017) proposed the cycle GAN as an alternative method of image-to-image translation using an unpaired dataset. For problems such as driving in extreme weather conditions, there is a lack of paired datasets due to the impracticality of obtaining the same photo of a road scene twice where the only difference is the weather conditions. Cycle GAN allows for image-to-image translation to be trained by training both the mapping from the input domain to the target domain, and the inverse from the target domain back to the input domain concurrently, adding a cycle consistency loss. While the authors found the results were mostly positive, there remains inconsistency in performing tasks requiring a large translation. Despite this, the ability to train with unpaired data presents cycle GAN with an advantage for tasks such as removing fog.

2.6. Dehazing with Cycle GANs

Not all studies on the cycle GAN have found it to be effective at defogging or removing other weather from an image. Gupta et. al (2024) Compared the cycle GAN to other defogging models and found it to be less reliable for detecting objects. This is because the cycle GAN introduced significant alterations and additional information, which negatively impacted the object detection performed on the processed image. To help maintain detailed information of the image, Sun et. Al (2021) embedded an iterative dehazing model into the generative process of the cycle GAN. They found this approach to improve the cycle GAN helped to maintain physical attributes when recovering high quality images while still keeping the thoroughness of the defogging.

In order to make a cycle GAN that solved for multiple weather conditions, Yang et. Al (2023) designed a modified cycle GAN consisting of four sets of generators and discriminators. One to convert foggy, rainy, and snowy images into clear images, and then three generators to transform clear images back to each of the three weather conditions. They found their model trained to work for multiple weather conditions achieved similar or better performance than dedicated de-hazing and de-raining models, while they achieved poor results for de-snowing due to pixel resolution

differences. They also found an improvement in pedestrian detection for images of pedestrians in weather conditions that were first passed through their model. Teeti et. Al (2023) also found that cycle GAN could be used to improve object detection in adverse weather conditions. They trained a cycle GAN to convert dark images at night to daytime, and to remove waterdrops on the camera lens in order to detect obstacles of the road for autonomous racing. They found it improved the detection of obstacles at night or in rain from 4 of the 5 different object detectors used, with yolov5 found to be the best object detector to use.

One method of improving the cycle GAN by incorporating paired training data in the beginning was explored by Graffieti and Maltoni (2021). They proposed curl-defog, where they began training using paired synthetic data as used in a conditional GAN to build their desired parameter-space region. They then progressively add in unpaired real data to refine the model with more complex training like a cycle GAN. They found that it weas useful to initialize their model with paired training data, and that the curl defog method was in line with other state of the art approaches.

2.7. Limitations of Cycle GAN Dehazing Studies

The study done by Gupta et. Al (2024) found that the cycle GAN was less effective than other dehazing methods. They used the DAWN dataset, which contains real hazy and foggy images with good variance between images. The poor performance was attributed to significant alterations and additional information being introduced, raising the question of how well this model could perform if improved to reduce the issue of significant alterations. Sun et. al (2021) aimed to address these issues and maintain detailed information of the original image. Despite being trained on synthetic data, the results showed that the cycle GAN was well suited to dehazing both synthetic and real hazy image. The testing was done for synthetic data by calculating the peak signal to noise ratio and the structural similarity index measure, showing that the dehazed image is close to the ground truth. To test with real data, they calculated the color natural index which assesses how natural the colors in an image are and judged the quality of the output based on visual inspection. This leaves a clear gap on quantifying the performance of the dehazing model on real data and does not include any measure of how much pedestrian detection may be improved on either synthetic or real images after being de-hazed by this model.

In the work done by Yang et. al (2023), they tested the results of the cycle GAN on both real hazy images and synthetic foggy images. For the synthetic fog they measured the log average miss rate over false positive per image for pedestrian detection, which is used in cases where false positives are critical, in order to minimize the false positive rate. They found improvement after defogging except in cases of heavy fog where no improvement was seen. Although they tested their results on real hazy images as well, this was only a quantitative analysis. This leaves two major gaps in the study: Measuring the improvement of pedestrian detection for real world images de-hazed using the cycle GAN, and how the cycle GAN can be improved to handle images with heavy fog. Teeti et. Al (2023) did test the performance of object detection on real weather conditions, though they were testing for performance in darkness and water droplets on the lens. Their model was also trained with synthetic data, though tests on real data still showed an improved average precision for object detection. While they did not directly measure the detection of pedestrians, it is likely to be similar levels of improvement as detecting other objects in driving scenes. Given that the weather

conditions in this study were darkness and water droplets, it's possible that there is a large difference for weather conditions such as haze and fog as they don't share the same traits.

The curl GAN created by Graffieti and Maltoni (2021) is a combination of the conditional GAN and cycle GAN approaches, starting with paired data and slowly introducing unpaired data. While this is not directly able to compare the two approaches to each other, it does show the potential of using these or similar approaches for de-hazing. In this study they trained the model with both synthetic hazy images for the paired data and introduced real hazy images for unpaired data in training. The testing was performed on real images taken with fog inserted by a fog machine in controlled conditions. This showed that the method improved dehazing results on these images when compared to other dehazing methods, though there are limitations to this test. Due to the requirements of creating these foggy images in a controlled environment, testing is done on a very small sample size of 55 images. Similarly, the testing is limited by the type of fog generated by the fog machines. They do generate a real fog for testing images; however, it does not include the same level of variability that you get in real world scenarios where you get fog or haze of different colors and densities. This study was the only one to measure the speed of their model, finding speeds from 30-40 frames per second depending on the size of the input image.

In Zhang's (2023) study, a cycle GAN was used for comparison with the conditional GAN they had developed. The cycle GAN showed significantly lower performance in peak signal to noise ratio and the structural similarity index compared to all other models tested, and also showed a significant decrease in average precision, with the conditional GAN getting 0.673 and the cycle GAN getting 0.149. This contradicts the other studies focusing on cycle GAN that have found it to improve these metrics when applied to a foggy image. This could be due to the fact that the cycle GAN was not correctly trained or implemented in this study, or that the synthetic dataset used for training was not sufficient for training the cycle GAN. This raises questions as to the accuracy of this model compared to other cycle GANs used for defogging, and further research can be done to verify these results and determine the significance.

2.8. Summary of Gaps in Literature

In this section we reviewed some studies that align with our research question, to identify what work has already been done and where more research is required. For both the conditional and cycle GANs, most studies tested their model on synthetic weather conditions. These don't capture the full extent of conditions seen in real life and may not be accurate representations of real world performance. Considering the conditional GANs and some of the cycle GANs were trained with synthetic weather data, by not testing them on images of real weather conditions it can't be determined if the models are effective at de-hazing in different conditions or if they've simply learned patterns in the generation of synthetic weather data. Most of these studies also focused on the ability to dehaze a paired synthetic image by measuring the difference between the ground truth and generated image. While this is useful in determining the ability of the model to de-haze an image, it does not provide any information as to what extent pedestrian detection can be improved by implementing these models.

Only the curl GAN was measured for speed, and while it showed capability of real time performance the other studies in this section did not provide any indication on the suitability of the speed of their models for real world applications. With these studies all using different datasets for

both training and testing their model, we found only found a single direct comparison between the cycle GAN and conditional GAN for dehazing, in the work done by Zhang (2023), who found the conditional GAN to offer significantly better accuracy than the cycle GAN. With this being said, the numbers recorded for the cycle GAN in Zhang's study seem abnormally low compared to the other studies considered in this section, and further research should be done to determine if these results were as significant as stated in the paper.

3. Methodology

3.1. Project Overview

With the goal of the project being to determine which GAN model is superior for detecting pedestrians in foggy weather, the process is shown in figure 1. For each of the conditional and cycle GAN's, foggy image is first input to the GAN. It then returns a dehazed image, from which we can use to detect pedestrians. The models are tested on the Pedestrian Detection in Hazy Weather dataset (Li et. Al, 2019), consisting of images of pedestrians who are heavily occluded by fog or haze. This is ideal for testing the models as it represents the extreme level of fog and haze conditions that an autonomous vehicle may have to handle in real life. Yolov5 is used as the object detector, Teeti et. Al (2023) found that yolov5 was the best detector for similar tasks at maintaining high accuracy and a fast frame rate. The results from each GAN will be compared to determine which GAN offered superior performance in both average precision and frames per second.

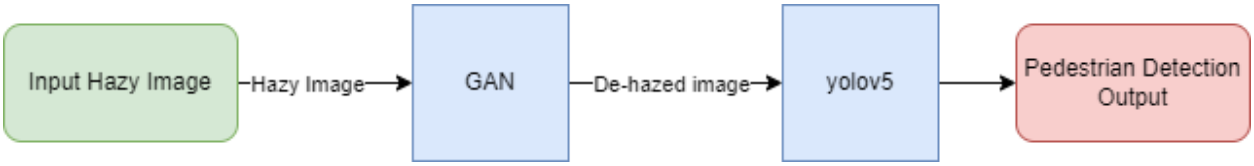

Figure 1: Flow chart of the final product.

3.2. Conditional GAN Training Datasets

To train the conditional GAN to remove haze, a paired dataset of clear and hazy data is required for training. Clear images of road scenes were obtained from the BDD100K dataset (Yu et. al, 2018), where all images were consistently sized at 720 by 1280. To create a paired dataset, an all white cloud layer was created and drawn as an ellipse to cover the image. Gaussian blur is applied to the cloud layer to smooth the edges of the ellipse, so the fog looks more natural, and it is then blended with the original image. This method of using cloud layers to create synthetic fog for dehazing models was successfully demonstrated by Chow et. Al (2023). An unpaired set of real hazy and clear images is used to train the cycle GAN. The BDD100K dataset contains images with both foggy and clear labels, which can be used as an unpaired dataset for training the model.

3.3. Conditional GAN Training

The structure of the conditional GAN is depicted in figure 2, consisting of one generator and one discriminator. The GAN was trained for 200,000 training steps, with one step consisting one paired image set being used for training. Random Jitter is applied to the images at the start of each training step to add randomness and ensure the model doesn't easily learn the training images. This involves applying random cropping and random mirroring to prevent overfitting the model. The dehazed image is generated from the generator output for the foggy image. The discriminator

outputs for the original paired image as well as the original foggy image with the dehazed image are then calculated. These are then used to calculate the generator and discriminator losses. The gradients are calculated from the losses and used to update the weights of both the generator and the discriminator. The code for each training step can be found in figure A1, found in appendix A.

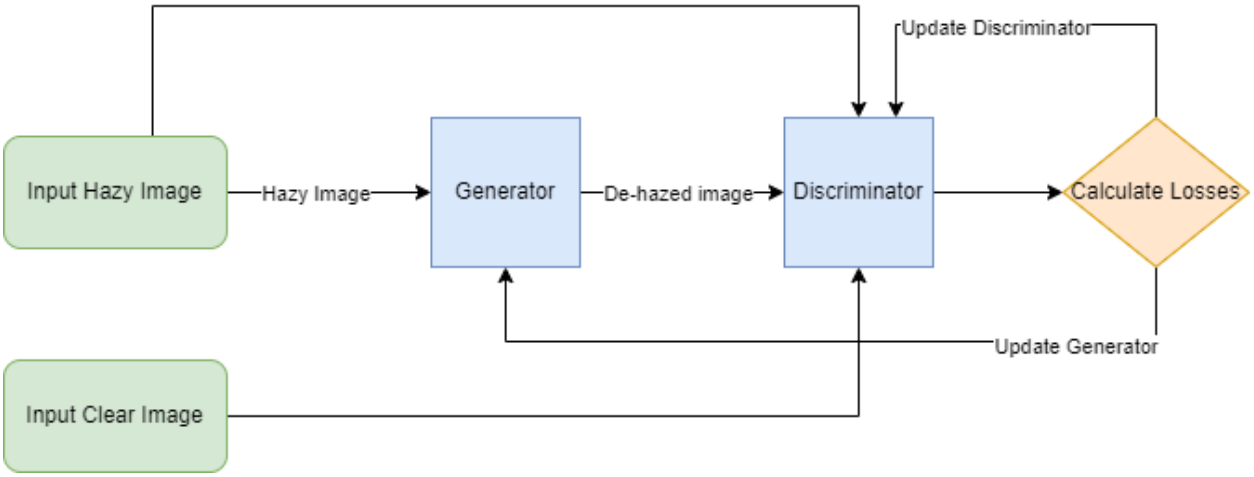

Figure 2: Conditional GAN process flow chart.

3.4. Conditional GAN Generator

The generator follows a U-net structure as shown in figure B1 found in appendix B, which is effective preserving spatial information in image-to-image translation. The foggy image is first passed through 8 downsampling layers to reduce the dimensions of the image while increasing the number of feature channels. Each downsample layer consists of a convolutional layer with a stride of 2 to reduce the dimensions, followed by batch normalization and leaky ReLU activation. This is followed by 7 upsample layers, to increase the dimensions of the image and decrease the number of feature channels. The upsample layers each consist of a transposed convolution layer to increase the image dimensions, followed by batch normalization and ReLU activation. A skip connection is used from the first 7 downsample layers to the 7 upsample layers, with the 8th downsample layer used as a bottleneck. The final layer it goes through is a convolutional layer with tanh activation to produce the dehazed image. The code for this generator can be seen in figure A2, found in appendix A.

3.5. Conditional GAN Discriminator

The discriminator is built using a Patch GAN structure as discussed by Isola et. al (2018), which focuses on distinguishing between real and generated images at the patch level rather than the entire image. The structure is shown in figure B2 found in appendix B. It takes an input and target image and stacks them together. This is followed by three downsampling layers to increase the number of feature channels. Zero padding is applied to maintain the spatial dimensions after downsampling, before going through a convolutional layer, batch normalization, and leaky ReLU activation. Zero padding is applied once more before a final convolutional layer to ensure proper dimensions for the output, which is a matrix of probabilities that a patch of the image is real or fake. Figure A3 found in appendix A shows the code for the discriminator.

3.6. Conditional GAN Loss functions

The discriminator loss is defined as the sum of two components: the real loss and the generated loss. The real loss is computed using binary cross-entropy on the discriminator's output when the original foggy image and clear target image are passed as inputs. This encourages the discriminator to correctly classify real images as real. The generated loss is also computed using binary cross-entropy, but with the dehazed image and the target image used as the input for the discriminator, encouraging it to correctly classify generated images as fake. The total discriminator loss is the sum of the real and generated losses.

The generator loss consists of a GAN loss using binary cross entropy to encourage the generator to produce images indistinguishable from the target, an l1 loss to measure the difference between pixels in the generated and target image, and the total generator loss is the GAN loss plus the l1 loss multiplied by a weight parameter lambda.

```python
def generator_loss(disc_generated_output, gen_output, target):
  """This function describes the loss function used for training the generator"""
  # Compute GAN loss: Encourages the generator to produce images that are indistinguishable from real images
  gan_loss = loss_object(tf.ones_like(disc_generated_output), disc_generated_output)
  # Compute L1 loss: Measures the difference between the generated image and the target image
  l1_loss = tf.reduce_mean(tf.abs(target - gen_output))
  # Total generator loss: Combines GAN loss and L1 loss
  total_gen_loss = gan_loss + (LAMBDA * l1_loss)
  return total_gen_loss, gan_loss, l1_loss

def discriminator_loss(disc_real_output, disc_generated_output):
  """This function describes the loss function used for training the discriminator"""
  # Compute loss for real images: Encourages the discriminator to correctly classify real images as real
  real_loss = loss_object(tf.ones_like(disc_real_output), disc_real_output)
  # Compute loss for generated images: Encourages the discriminator to correctly classify generated images as fake
  generated_loss = loss_object(tf.zeros_like(disc_generated_output), disc_generated_output)
  # Total discriminator loss: Combines the loss for real and generated images
  total_disc_loss = real_loss + generated_loss
```

Figure 3: Conditional GAN generator and discriminator loss functions.

3.7. Cycle GAN Training

The cycle GAN structure depicted in figure 4, consists of two generators and two discriminators, and is trained for 100 epochs. Due to memory constraints training the model, mixed precision was used in training. The impact of this is not clear as the model could not be trained without it for comparison, even through reducing the number of ResNet blocks or buffer size. Using an unpaired clear and foggy image, we first apply random jitter to stop the model from easily learning the images and prevent overfitting. Next, we are creating a fake image, cycled image, and identity mapping for each of the foggy and clear inputs. The real and fake images are then each used in the foggy and clear discriminators to obtain the four discriminator outputs. All of these generator and discriminator outputs are used in the calculations of the loss functions. These are then used to calculate the gradients for each generator and discriminator, and then applied to update their weights. Figure A4 found in appendix A shows the code for each training step of the cycle GAN.

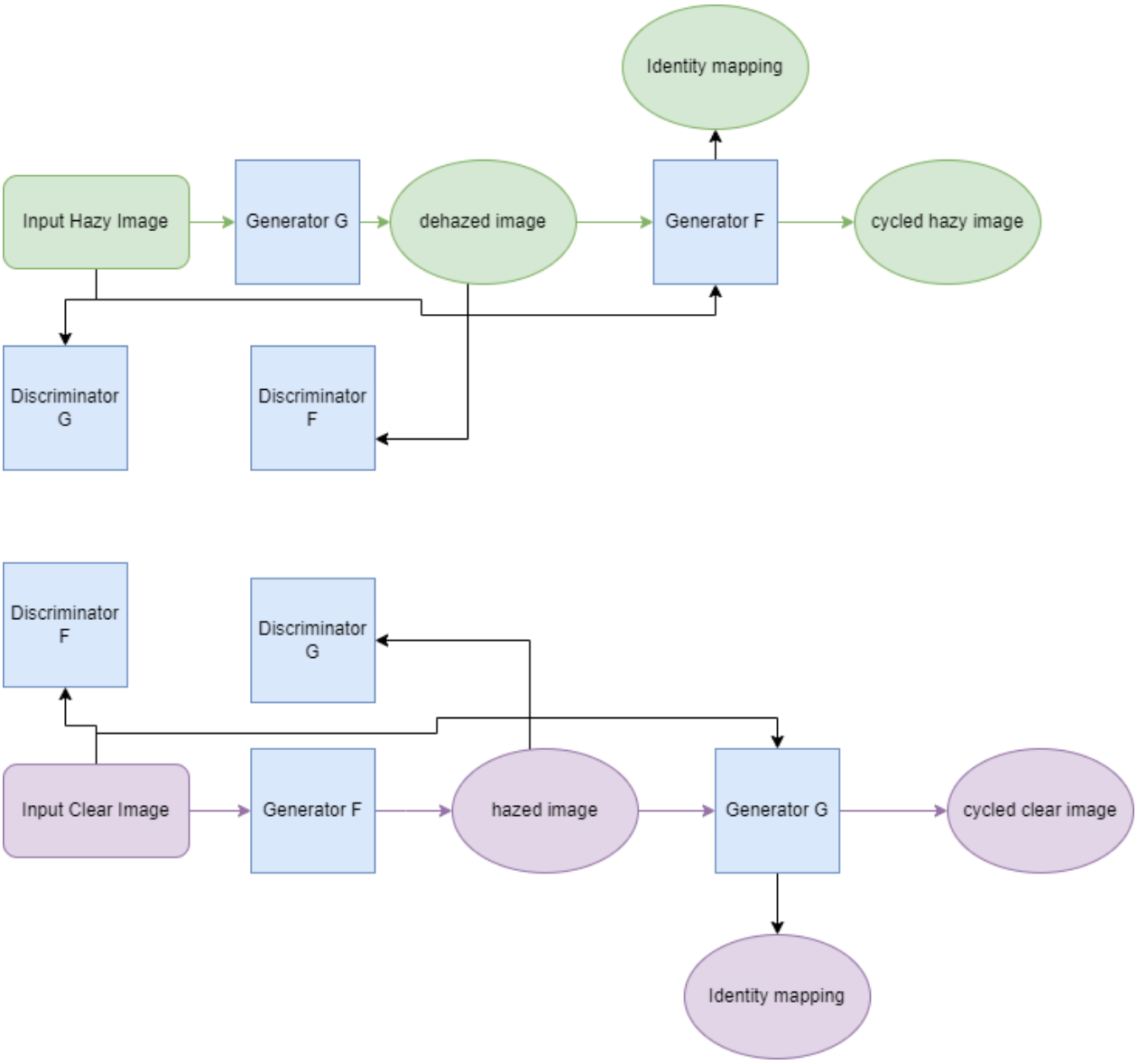

Figure 4: Cycle GAN process flow chart

3.8. Cycle GAN Generator

The Cycle GAN generators are based on a ResNet architecture shown in figure 3B found in appendix B, instead of the U-net used in the conditional GAN. The skip connections in ResNet allow for training deeper networks without the issue of vanishing gradients. The generator starts with an initial convolutional layer that processes the input image to extract low-level features. This is followed by two downsampling layers, each of which consists of a convolutional layer with a stride of 2 to reduce the spatial dimensions while increasing the number of feature channels. Each downsampling layer is followed by batch normalization and ReLU activation to stabilize training and introduce non-linearity. Following this is 12 residual blocks, each designed to maintain the image's content while transforming its style. These blocks help to learn more complex transformations by

maintaining the input features and combining them with learned features. Each residual block has two convolutional layers with batch normalization and ReLU activation. A skip connection is also established from the start to end of the residual block to help avoid vanishing gradients. After the residual blocks, two upsampling layers are applied to increase the spatial dimensions of the image while reducing the number of feature channels. Each upsampling layer consists of a transposed convolutional layer, batch normalization, and ReLU activation. The final layer is a convolutional layer with a 'tanh' activation function to produce the transformed image. The code for the ResNet generator is shown in figure A5 and the code for the Resnet blocks is shown in figure A7 found in appendix A.

3.9. Cycle GAN Discriminator

The Cycle GAN discriminators are based on a Patch GAN architecture, similar to the conditional GAN discriminator described above, which evaluates the realism of the image at the patch level instead of the entire image. The structure is shown in figure B4 found in appendix B. It first applies a series of seven downsampling layers on the input image to progressively reduce the spatial dimensions of the image while increasing the number of feature channels. Each downsampling layer consists of a convolutional layer, batch normalization (except for the first layer), and a Leaky ReLU activation function. After the downsampling, the output is passed through a zero-padding layer to maintain the spatial dimensions before feeding it into a convolutional layer. Batch normalization is applied again, followed by a LeakyReLU activation function to introduce non-linearity. A second zero-padding layer is then used to maintain the dimensions for the final output. The last layer is a convolutional layer, producing a matrix that represents the probability of each patch in the input image being real or fake. Figure A7 found in appendix A shows the code for the discriminators.

3.10. Cycle GAN Loss functions

The cycle GAN uses the same generator and discriminator loss functions as the conditional GAN but has two more loss functions to be added to the generator loss in order to calculate the total generator loss, as shown in figure 5. The first of these is the cycle consistency loss, which is the L1 loss between the original and cycled images. This helps to ensure an image resembles the original after domain transfer. The identity loss is also added, which is the L1 loss between the original and generated image. This loss function helps to preserve the original color and content of the image when it moves to the target domain.

```python
def calc_cycle_loss(real_image, cycled_image):
    """This function calculates cycle consistency loss for the generator, used to ensure an image resembles the original after domain transfer"""
    real_image = tf.cast(real_image, tf.float32)
    cycled_image = tf.cast(cycled_image, tf.float32)
    return LAMBDA * tf.cast(tf.reduce_mean(tf.abs(real_image - cycled_image)), tf.float32)

def identity_loss(real_image, same_image):
    """This function calcyulates identity loss, helps in preserving color and content in target domain"""
    real_image = tf.cast(real_image, tf.float32)
    same_image = tf.cast(same_image, tf.float32)
    return LAMBDA * 0.5 * tf.cast(tf.reduce_mean(tf.abs(real_image - same_image)), tf.float32)
```

Figure 5: Cycle and Identity loss functions used for the cycle GAN.

3.11. Testing the Models

To measure the performance of the conditional GAN and cycle GAN, two metrics are being calculated: Average precision, and frames per second. Average precision provides a summary of the precision recall curve, to measure how well pedestrians are detected by the model. Using the bounding boxes from yolov5 predictions and the bounding boxes provided in the dataset as ground truth, intersection over union (IoU) is calculated to measure the overlap between the boxes. Based on the IoU each yolov5 prediction can be classified as a true positive or false positive, with false negatives being the ground truth boxes that didn't get a match. These are used to calculate the average precision. Next, frames per second are calculated by simply putting every image in the test dataset through the generator and measuring the time it took. For either of the GANs to be used in an autonomous vehicle, they must be capable of real-time performance to process images quickly while the vehicle is moving. This test uses an Nvidia RTX 4080 Super GPU while testing the frames per second of the models, with 20 frames per second used as a benchmark for real time performance. The yolov5 pedestrian detection is excluded from the fps test as the goal is to measure the speed of the conditional and cycle GAN's, not to measure the speed of yolov5.

4. Results

4.1. Synthetic Hazy Images

Following the planned creation of synthetic foggy images outlined in section 3.2, the conditional GAN learned very quickly how to remove the cloud layer from the image. This, however, did not translate to the test dataset, as the model had just learned the simple pattern used to create synthetic fog. The images had a thick white cloud in the center of the image, slowly transitioning to slightly darkened edges and corners around the outside of the image. Similarly, outputs on real data were often much darker with the limited visibility in real hazy images.

To try and solve these issues, some changes were made to the process of creating synthetic foggy images, with the code for these changes shown in figure 6. First, all images at night were removed from both the clear and synthetic fog dataset, and only daytime images were considered. Since the purpose of the model is to improve pedestrian detection, generating a dark or nighttime image would make it much more difficult to detect a pedestrian than generating a brighter, daytime image. The test dataset does not include any nighttime images, they are all either during the day or have such a thick haze that there is no indication of the time.

The next step was to add more randomness and variability to the cloud layer, so that the model couldn't easily learn the pattern of the fog. The ellipse was modified to have a random size and shape within 10% of the image's height and width to add some randomness to the fog while ensuring it covers most of the image, and some randomness was added to the radius of the gaussian blur. Finally, a random blend factor between 0.6 and 0.8 was used for blending the cloud layer to the image to ensure that the images all represented a heavy fog with varying intensity.

```
# Create clouds layer
clouds_layer = Image.new("RGB", original_image.size)
draw = ImageDraw.Draw(clouds_layer)

# Random position and size for the ellipse
width, height = original_image.size
ellipse_x0 = random.randint(int(width * -0.1), int(width * 0.1))
ellipse_y0 = random.randint(int(height * -0.1), int(height * 0.1))
ellipse_x1 = random.randint(int(width * 0.9), int(width * 1.1))
ellipse_y1 = random.randint(int(height * 0.9), int(height * 1.1))

draw.ellipse((ellipse_x0, ellipse_y0, ellipse_x1, ellipse_y1), fill=(255, 255, 255))

# Add random noise for the cloud layer
noise = Image.effect_noise(original_image.size, random.uniform(10, 30))
noise = noise.convert("RGB")
clouds_layer = Image.blend(clouds_layer, noise, alpha=0.2)

# Apply Gaussian blur to clouds layer with random radius
blur_radius = random.uniform(130, 200)
clouds_layer = clouds_layer.filter(ImageFilter.GaussianBlur(radius=blur_radius))

# Random blend factor
blend_factor = random.uniform(0.6, 0.8)

# Blend original image with clouds layer
foggy_image = Image.blend(original_image, clouds_layer, alpha=blend_factor)
```

Figure 6: Creating a cloud layer for synthetic foggy images with randomness introduced for variability.

An example of the synthetic fog creation and de-hazing is shown in figure 7. It shows a high amount of similarity from the ground truth image, with the loss of a few colors as seen in the red of the taillights, the red on the building, or the lighter blue color of the sky.

Figure 7: Synthetic fog generation using a cloud layer and removal with conditional GAN.

4.2. Hazy Images for Cycle GAN

After beginning training on the cycle GAN, we found that the foggy images in the BDD100K dataset were not well suited to this problem. The foggy images provided in this dataset showed an extremely light fog and objects were not at all occluded by the fog as it was only present in the background, making it unsuitable for training to detect pedestrians occluded by heavy fog. This can be seen in figure 8, where all objects are clearly visible in the image with the fog only noticeable in the background.

Figure 8: Foggy Image from BDD100K dataset, showing objects clearly with fog only present in the distance.

Instead, the DAWN dataset (Kenk & Hassaballah, 2020) is used for foggy images, containing driving scenes in heavy fog, haze, and mist. An example of this is shown in figure 9, with a thick haze covering everything in the image. These images better represent the heavy fog and haze that are present in the test dataset, making it much more suitable for training the model. It is also a much better representation of the challenges that will be present in real world driving scenarios, where thick and heavy haze can obstruct everything in view.

This dataset introduced a new challenge that was not previously seen in training either model, the dimensions of the images are inconsistent. The height and width of images in this dataset range from 50 pixels to 1500 pixels. To handle the issue of the inconsistency of image sizes, padding or random cropping was performed to ensure all images matched the 720 by 1280 dimensions of the clear images. Because the goal of GAN training is to learn the mapping between clear and foggy domains, the specific content within each image is less relevant than the overall transformation quality. Therefore, we are not at risk of cropping out information that is important to the models training. These methods were chosen over resizing in order to maintain the aspect ratio and prevent any distortion in the image.

Figure 9: Hazy image from DAWN dataset used for training the cycle GAN.

4.3 Cycle GAN synthetic haze generation and removal

The cycle GAN consists of two generators, one to remove haze from an image and one to add haze to an image. Figure 10 shows haze being generated on a clear image, and then being de-hazed by the second generator. The synthetic haze created looks similar to the training dataset with a grey haze covering objects in the image, and much of the color in the image is either covered by the haze or removed. After being de-hazed, the colors do not return to the image and some of the background objects are lost. However, the objects in the foreground appear more clearly than in the image with synthetic haze.

Figure 10: Synthetic haze generated and removed with cycle GAN

For darker input images of clear weather as seen in figure 11, the discoloration and loss of background details is more severe. Adding the haze to these darkened images almost makes them appear as grayscale, which is an accurate representation of what haze may look like without any bright colors around. However due to this, the cycle GAN is only adding color to the background when removing this haze, and it becomes difficult to make out any object that is not at the front of the image.

Figure 11: Synthetic fog generation using a cloud layer and removal with conditional GAN.

4.4. Conditional GAN De-hazing image outputs

The conditional GAN appears to perform better at defogging the image when the input image is a lighter grey color, as seen in figure 12. As the color of the haze in the image progressively gets darker as shown in figures 13-15, the conditional GAN seems to stop dehazing and predict increasingly dark backgrounds. When the object in the image is clear enough to see as in figures 12-14 the conditional GAN generates a light border around the object, which likely helps with the pedestrian detection component as it makes them stand out from the background. If the person is well blended into a dark background as seen in figure 15, the conditional GANs tendency to generate dark backgrounds for these images is completely covering the pedestrian in darkness. The conditional GAN provided extremely realistic results for images with lots of color visible behind the fog, such as the image in figure 16. These types of images were consistently given clean outputs from the conditional GAN.

4.5. Cycle GAN De-hazing image results

In the initial training, no matter how long the model was trained for each epoch would result in a huge shift between distorted objects and colors in the validation data. To correct this, we used a slower learning rate and increased the lambda value used as a multiplier in the loss functions was required to allow the output on the test images to start to converge to a consistent output. This is because lambda is used in the cycle and identity loss functions, and increasing it causes prioritization to make the generated image more similar to the original. This allowed the model to better converge to a single prediction as training progressed and resulted in more consistent outputs.

Despite these changes, images dehazed by the cycle GAN often did not maintain background colors from the input image, as seen in figures 12-14. This is not effective on lighter color backgrounds such as figure 12, where the blue background appears equally as occlusive as the grey in the input image. Once the input image fog colors become darker, this becomes useful in making the pedestrians stand out from the background. This is seen in figures 13 and 14 where the pedestrians stand out against a slightly off colored background, and in figure 15 with a dark image the cycle GAN helped the pedestrian to better stand out. While it appears to do a good job of making pedestrians directly in frame stand out against the fog, objects in the distance don't appear any clearer with the cycle GAN. Some examples of this can be seen with the pedestrians in the distance in figure 12 or the tree on the left of figure 13. The cycle GAN was not as consistent as the conditional GAN with colorful images and tended to struggle with the colors for distant objects or the background. This is evident in figure 16 where the pedestrians crossing the road are clear and

obvious, but the sign and pedestrian standing underneath of it on the right side of the image are not clear and look to blend in with the background colors.

Figure 12: Dehazing results on test dataset for a colorless photo with light grey haze

Figure 13: Dehazing results on test dataset in a dark thin haze

Figure 14: Dehazing results on test dataset for thick dark haze with some visibility

Figure 15: Dehazing results on test dataset for thick dark haze with very limited visibility

Figure 16: Dehazing results on test dataset for a light haze with lots of visible colors

4.6. Model Accuracy

The first metric being measured is average precision, to determine how accurate pedestrian detection using yolov5 is after the image is dehazed by the models. The average precision for each of the conditional and cycle GANs along with the average precision of the unmodified dataset are shown in table 1. Without processing the images with one of the models created, pedestrian detection with yolov5 has an average precision of 0.53. This is increased significantly by using the conditional GAN to dehaze the image, increasing the average precision to 0.85. The cycle GAN saw a similar level of improvement, resulting in an average precision of 0.83.

	Unmodified Input	Conditional GAN	Cycle GAN
Average Precision	0.53	0.85	0.83

Table 1: Average precision results for both models using yolov5 object detector.

The precision recall curve for the conditional GAN is shown in figure 11. There is a steep drop in precision from 1.0 to around 0.71 in the very low recall range, before recovering back to a high precision of 0.95. The precision then starts on a fairly steep downward trend, before flattening out to a slight downward trend after passing the recall value of 0.2.

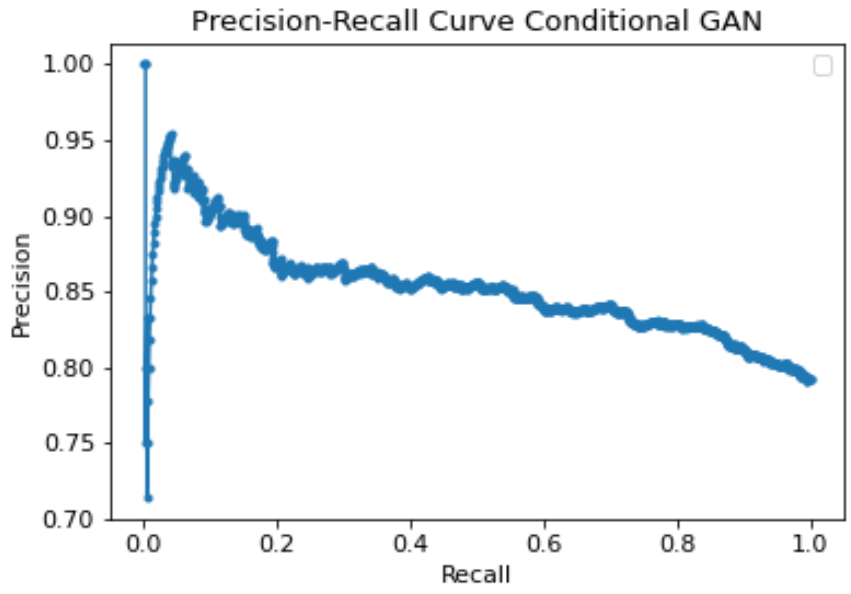

Figure 11: Precision recall curve for the conditional GAN

For the precision recall curve of the cycle GAN shown in figure 12, there are two early steep drops in precision, however they are less significant than the drop seen in the conditional GAN as it only drops around 0.1 precision at a time. After a slight recovery in precision after the second drop, the precision continues on a steady downward trend for recall values from 0.2 to 1.0.

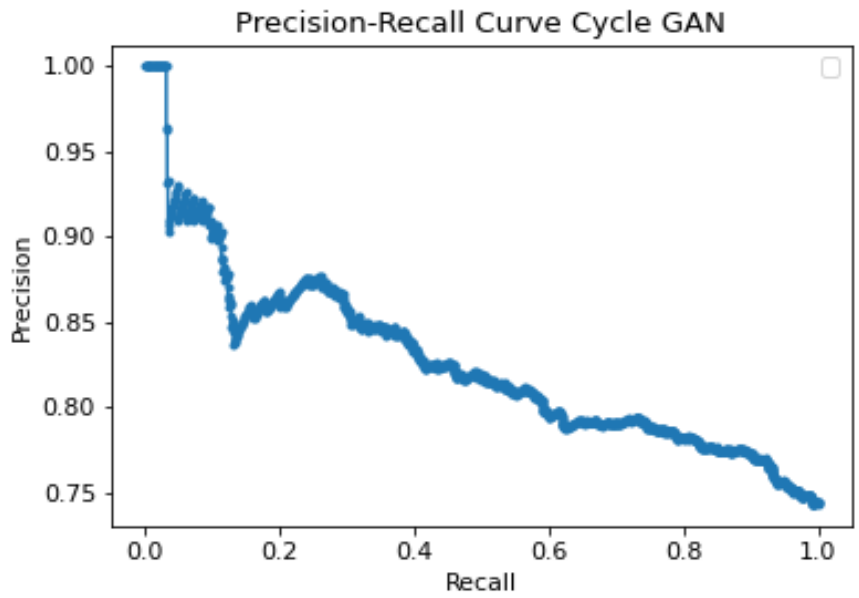

Figure 12: Precision recall curve for the cycle GAN

4.7. Model Speed

To determine if the models were suitable for real-time applications in an autonomous vehicle, the threshold was set to maintaining 20 frames per second while defogging images. Both of the models passed this threshold, as shown in table 2. The conditional GAN was significantly faster than the cycle GAN, processing 51.32 frames per second, compared to the cycle GANs 24.96 frames per second. The slower processing time in the cycle GAN is related to the number of ResNet blocks used in the generator. Reducing the number of blocks from 12 down to 7 would have increased the frames per second to 34, however it also dropped the average precision to 0.80.

	Conditional GAN	Cycle GAN
Frames per Second	51.32	24.96

Table 2: De-hazing frames per second

These results suggest that the Conditional GAN is more suitable for applications requiring faster processing speeds, such as real-time pedestrian detection in autonomous vehicles. However, the Cycle GAN, while slower, still meets a high enough performance level for real-time tasks but may require optimization to balance speed and detection accuracy effectively.

5. Discussion

5.1. Conditional GAN De-hazed Image Quality

The dehazed images shown in figures 12-16 show the high variability of defogging performance based on the color and thickness of the fog in the image. Light shaded fogs and colorful backgrounds tend to lead to more realistic results. Colorless images with light shades of fog show clear foreground with some uncertainty in the background. Images with dark shades of fog produce darkened backgrounds with a light glow around any objects shown close in the image. These high levels of variabilities were not mentioned in other studies using conditional GAN's to dehaze images. Chow et. Al (2023) only used images with synthetic haze while testing their model, which was a much simpler task to train for synthetic haze. Our model easily transformed the synthetic foggy images we created into clear images, as it is much easier for it to learn the characteristics of synthetic fog. Due to this our model faced a much more challenging test dataset, leading to less consistent results. Siddiqua et. al (2023) tested with the DAWN dataset, which we used for training our cycle GAN. While this dataset does have some images with darker fog and haze, none of these were shown or discussed in the paper, with all samples provided being a light colored fog on a colorful background. This highlights a gap found in current research, where the effect of a conditional GAN on dense, dark fog/haze has not been highlighted in similar studies. This is significant when applying the solution to a real world application, as there are large variances in the thickness and color of hazy or foggy conditions in the real world. Dense and dark haze or fog occludes pedestrians significantly more than a light haze, therefore testing the data on a dataset that incorporates all these challenges is required to have any confidence of applying it to a real application.

5.2. Gaps in Conditional GAN Research Addressed

In section 2.4 we identified gaps in current research of conditional GANs in de-hazing images with thick haze or fog for pedestrian detection. Similar studies did not test their model on real world data, only on synthetic images. They also did not measure the speed of their models to determine if they would be suitable for real world applications.

Our results clearly show that the conditional GAN greatly improves the quality of pedestrian detection in real world images, however we do not get clear outputs with the haze removed from the image like we see in figure 5 when using the model on synthetic fog. This highlights the additional challenge of applying this model to real weather conditions but shows that the performance of pedestrian detection is still greatly improved despite the de-hazed image not always being clear. The high frames per second measured with this model clearly showed that it is well suited to real world applications, which was not measured in other related studies.

5.3. Recommendations for improving the quality of images dehazed with conditional GAN

Considering the findings for the conditional GAN discussed in the last section and comparing the test dataset to the synthetic hazy images used for training, the decision to use a white cloud layer is an oversight. Adding some randomness to the darkness of the cloud layer to range from a white to medium grey would have been a better representation of the conditions seen in the test dataset.

With this change there may not be such a large gap in dehazing performance between images based on the differences in the darkness of the haze.

Similarly, applying other methods to make the cloud layer more realistic will help to make it handle more diverse situations better. Using a dataset with camera images paired with Lidar, the haze could be dynamically created to be more dense in front of parts of the image that are far away on Lidar, and less dense on parts of the image that read closer to the Lidar sensor. This would give the synthetic fog a much more realistic look and better represent the way seeing through haze works in a real situation. Future studies could also test the impact of using multiple cloud layers of different sizes and densities, that could be applied to different parts of the image or overlap with each other. While this does not necessarily provide a realistic looking haze, the increased variability between images could improve the models' training.

Another challenge that may have limited the performance of the final model is that the test dataset contained many images without any color. While it is reasonable to expect that any camera on an autonomous vehicle would not be capturing images in black and white, thick haze and fog scatters light and distorts colors, and can sometimes give the impression of a colorless image. Preparing for this in the training dataset by removing color for some images would have better prepared it for these scenarios. Adding this increased level of variability into the model's training could add some complexity to the model. A deeper architecture and higher computational costs may be required to maximize the model's performance as the data evolves into a more thorough and complex dataset.

5.4. Cycle GAN Dehazed image Quality

The cycle GAN had issues with color distortion in the generated images, which Ogunrinde and Bernadin (2021) found to be a common problem across similar studies. While this didn't appear to have much effect on pedestrians or other objects in the front of an image, pedestrians and objects in the background tended to blend in with the background due to the color distortion. While this may not have significant effect on pedestrians close to the vehicle, it greatly reduces the vehicle's ability to identify and track a pedestrian in the distance. Early detection would allow a vehicle to slow down before getting too close to the pedestrian, making it much easier to stop. It would also allow the pedestrian to be tracked by other systems in the autonomous vehicle, such as to make predictions on the likelihood of the pedestrian stepping out in front of the vehicle.

Similar to what we saw in other studies for the conditional GAN, other de-hazing studies such as Sun et. al (2021) and Yang et. al (2023) with the cycle GAN were tested on datasets with less severe haze, often lightly occluding distant objects but with little occlusion for objects in the front of the image. This is a gap that highlights the need for further study on dense hazy images. Our model did make pedestrians stand out against the background in these types of images as shown in figures 13-15, however the images were not dehazed to a similar quality of the other studies mentioned. This was also true for some images with lighter haze where distant objects were lost in the transformation, as seen in figure 12. One of the reasons that our model was not able to de-haze these images with a high level of quality is because of the synthetic haze being generated by the cycle GAN. Figures 10 and 11 show haze being synthetically added to clear images, in which the haze being added is a similar density and color. This is a similar issue to what we experienced with the conditional GAN, where the synthetic hazy images weren't realistic enough to properly train the model to handle real world severe weather. Focusing on improving the synthetic haze generation to

generate more challenging hazy images would in turn improve the way that the cycle GAN is able to remove haze from these images.

5.5. Cycle GAN Limitations and Challenges

At this point it is unclear how significant the effect of using mixed precision was on the image outputs from the cycle GAN. Due to this it's not clear if the cycle GAN tested was performing at its full potential, and it was not possible to test without using mixed precision for comparison due to hardware limitations. Mixed precision allows the use of float16 and float32 data types to improve the speed and reduce memory consumption in an application. This was required as the hardware for training did not have enough memory to only use float32, and allowing the lower precision float16 was necessary. There haven't been any similar studies where the effects between mixed precision and float32 have been compared for dehazing images to establish a baseline.

The memory limit also affected the batch size used in training the model, which needed to be reduced from 1000 to 100 for the model to be able to train without encountering memory errors. To determine the impact of lowering the batch size, the model was trained again with batch sizes of 10 and 50. The average precision stayed at 0.83 when reducing the batch size to 50 and dropped to 0.80 when reducing the batch size to 10. With the results staying consistent when changing the batch size from 100 to 50, it indicates that there is still enough randomness in the order of images to not have an impact on the results. Once the batch size was reduced to 10 the average precision lowered, indicating that this change had a negative impact on training due to the more predictable order of images in training. This tells us that the impact of reducing the batch size to 100 from 1000 is likely minimal, however future studies that are not facing hardware limitations in their training should try to increase this value.

It was much more difficult to tune the cycle GAN for consistent outputs from the test dataset when compared to the conditional GAN. The model is much more complicated and does not use a paired dataset, and we found that it was much more sensitive to any change in hyperparameters. This was seen in our training when the learning rate had to be decreased, and lambda had to be increased in order to achieve consistent results. Making small changes to these parameters had a great influence on the images generated by the model, and due to limitations of both the timeframe of this project and the time required for the cycle GAN to train, it was not possible to try every combination.

5.6. Gaps in Cycle GAN Research Addressed

Despite the cycle GAN not requiring a paired dataset for training, many of the studies highlighted in section 2.7. tested their model on synthetic fog. They did this to have a ground truth image from a paired dataset so that they could calculate metrics such as the log average miss rate over false positive per image. This left the cycle GAN relatively untested on real weather images, except for the small test sample done by Graffieti and Maltoni (2021). This study showed that the cycle GAN faces real challenges with color distortion and keeping information in the background of the image intact. With this considered, the results still showed that despite the challenges this model provided a significant improvement for detecting pedestrians in real hazy images. This study also agreed with Graffieti and Maltoni (2021) in finding that the cycle GAN was capable of dehazing at a fast enough rate for real time applications.

The results of the study disagree with what was found in the study done by Zhang (2023) where the cycle GAN was sited to have extremely low performance compared to all other methods. While we also found higher performance from the conditional GAN, the two were both found to offer a significant boost in average precision with high enough numbers to both be considered as a reasonable approach.

5.7. Recommendations for improving the quality of images dehazed with cycle GAN

Issues of color distortion and loss of features have been discussed in other dehazing studies, and some of those methods could likely improve the output of our model. Sun et. al (2021) developed an additional loss function for information consistency to recover more details and colors from the original image, which could result in more consistent transformations from our cycle GAN. Alternatively, the curl GAN approach discussed by Graffieti and Maltoni (2021) could help to prevent some loss of background information. The curl GAN begins training with a paired dataset like the conditional GAN, and slowly introduces unpaired data. Early stages of training using paired data could help to ensure that objects aren't blended with the background before unpaired data is introduced for training, to allow for more complex training and model refinement. To ensure the model performs to its full potential it would also be advantageous to ensure access to hardware with the capability of training the model without requiring mixed precision.

The issues of color distortion and loss of background information did not only occur on the real hazy images, but also the clear images with synthetic haze generated by the cycle GAN. Even with the haze generation removing some colors from the clear image, the model was not able to correctly learn how to keep or brighten colors when translating from a hazy image to a clear image. Many of the studies discussed in section 2.6 were able to achieve dehazing results much closer to the ground truth on synthetic fog, indicating that improvements can be made to this model. Spending more time working with hyperparameters and tweaking the architecture of discriminators and generators may improve the results of the model on synthetic fog, possibly translating to the real images in the test dataset. Figures 10 and 11 show that the haze generation favored a loss of color with a grey haze being introduced. While this is an accurate representation of many of the hazy images in the test dataset, it indicates that there may not be enough variance in the DAWN dataset used for hazy images in training. Using additional training data with more diverse types of haze and fog will increase the variance in haze generation and should help the model be better equipped to handle the diverse range of hazy conditions seen in the real world.

5.8. Comparing Model Average Precisions

Despite some of the challenges discussed in generating accurate de-hazed images, both models showed a clear increase in average precision when compared to the average precision of pedestrian detection on the original foggy images. This is shown in table 1 where the unprocessed test data had an average precision of 0.53, the conditional GAN had an average precision of 0.85, and the cycle GAN had an average precision of 0.83. While the conditional GAN had a slight edge in terms of precision, it is clear to see from the results that each GAN has challenges with different types of images. Dark images get better results from the cycle GAN, and light images get better results from the conditional GAN. Due to the differences in test data from other studies using conditional and cycle GANs, a direct comparison of average precision would not be meaningful. The authors of the test dataset (Li et. al, 2020) proposed a modified pedestrian detection model to

replace yolo for pedestrian detection in heavy fog conditions. They achieved an average precision of 0.86 on this dataset, which is comparable to the results from our GAN de-hazing. While the conditional and cycle GANs produced slightly lower average precisions, the authors model was trained on a portion of the dataset we used for testing, removing some of the challenges discussed earlier such as the darkness of synthetic fog for the conditional GAN, or the presence of colorless images which were not included in the cycle GAN training dataset. Both of our models still provided similar results to the authors, with the conditional GAN proving slightly more accurate than the cycle GAN.

5.9. Precision Recall Curves

Understanding the details behind the average precision metric is important in determining the suitability of the model for pedestrian detection. False positives in an autonomous vehicle are inconvenient. It could cause the vehicle to stop or swerve to avoid collision when there isn't actually anybody there. Similarly, a false negative could result in a serious accident if the system does not detect the pedestrian and continues driving forward. The threshold can be adjusted based on the precision recall curve to find the correct balance between precision and recall to try and minimize the effects of both false positives and false negatives. Both of the precision recall curves for the conditional and cycle GANs suffer from a sharp drop in precision at a low threshold where the recall is still near zero. This drop is much more significant for the conditional GAN; however, they are likely due to the small sample size of data at that threshold. Since there are only around 1000 test images, there would be few true positives at such a low threshold, and a single false positive would result in a steep drop in precision. It's expected that with a larger data set for testing we would not see such a sharp drop due to increased sample size. Both lines stabilize into a steady trend after passing a recall of 0.2, however the slope of decline in the cycle GAN is roughly twice as steep as the conditional GAN. This shows that at higher thresholds the conditional GAN is striking a better balance between false positive and false negative values, avoiding a steep drop off in performance.

5.10. Comparing Model Speeds

The accuracy of a model is irrelevant for an autonomous vehicle if it can't perform in real time. Autonomous vehicles need to make quick decisions when it comes to stopping for a pedestrian, and any speed slower than real-time would not be suitable for an autonomous vehicle due to the reaction time required to avoid a sudden obstacle like a pedestrian. While both the conditional and cycle GANs met the threshold of 20 frames per second for real time performance, the conditional GAN was over twice as fast at 51.32 frames per second, compared to the cycle GANs 24.96 frames per second. The cycle GAN is a deeper network than the conditional GAN, and to get the best de-hazing results it required 12 residual blocks. Managing to optimize a model with less residual blocks causes a big boost in speed, at 7 blocks the frames per second increased to 34, although the average precision dropped to 0.80. This highlights the challenge of striking a balance between accuracy and speed for real time applications. The smaller network in the conditional GAN provides a clear advantage here, as it requires less processing power for real time performance which is beneficial in an autonomous vehicle where multiple systems will be running together.

5.11 Summary of Research Related to Gaps in Current Literature

This study highlighted the additional challenges of de-hazing real images instead of synthetic images as mostly done in previous studies. Despite these challenges, it showed that both the conditional GAN and cycle GAN are viable options for performing pedestrian detection on hazy images. It also found that both models maintained a high enough frames per second to process images for real world applications in real time. Most previous studies did not directly compare the cycle GAN and conditional GAN to each other to determine which one is best suited to the research problem, however the extreme difference in performance levels measured by Zhang (2023) is called into question by the results of this study. The conditional GAN was found to be superior in accuracy and speed, however both models provided a high enough level of accuracy to justify using either model in future studies depending on the available data. Future studies should also aim to implement the suggestions highlighted in sections 5.3 and 5.7 to improve on the models used in this study, and to determine if the conditional GAN still offers superior performance when both models are operating closer to their full potential.

6. Conclusion

6.1. Key Findings

Viewing the images output from each of the GAN, it's clear to see that neither model was very successful in generating an accurate dehazed image except for in ideal conditions. The conditional GAN struggled for images with a darker grey haze or images lacking color, while the cycle GAN had trouble with color distortion in the images, and objects in the background being blended into the background. While this is significant in discussing the results, the quality of how realistic the generated image looks was not the main objective for this project. The purpose of this study was to determine whether the conditional GAN or cycle GAN was better suited to dehazing images for pedestrian detection.

Both models showed a significant improvement compared to unprocessed images in average precision of pedestrian detection using yolov5. Without processing the average precision 0.53, while the conditional GAN and cycle GAN achieved 0.85 and 0.83 respectively. The precision recall curves indicated the significance in difference between the two models as the cycle GAN was a much more gradual decline when compared to the cycle GANs precision recall curve. This is important for finding balance between the precision and recall for the final model before real world implementation. This indicates that while the cycle GAN also provided a large increase in the performance of pedestrian detection, the conditional GAN offered both a higher average precision and more consistent performance over varying thresholds as seen in the precision recall curve.

The speed of the conditional GAN was far superior to the cycle GAN, processing 51.32 frames per second compared to the Cycle GAN's 24.96 frames per second. While these both met the threshold of 20 frames per second required for real time performance, the light model and faster times displayed by the conditional GAN suggest it is much better suited to real world deployments, where rapid decision making is crucial. Our findings indicate that the conditional GAN is the better model to use for dehazing for pedestrian detection in terms of both precision and speed.

6.2. Impact of Research

This project focused on expanding the research that has been focused on using GANs for dehazing tasks, specifically for use in autonomous vehicles. By comparing conditional GANs and cycle GANs, this study provides insights into their respective strengths and limitations for de-hazing images to improve pedestrian detection. The findings suggest that the conditional GAN, with its superior speed and precision, is a more viable candidate for deployment in real-world applications, such as autonomous vehicles.

This research also highlights gaps in current methodologies, particularly the need to train and test models on more diverse datasets that reflect real world conditions. Existing studies often focus on ideal or synthetic conditions, which do not account for the wide range of fog and haze densities and color variations encountered in the real world. By addressing these gaps, this study highlights the importance of developing more robust dehazing techniques to ensure safer and more reliable pedestrian detection systems in varying weather conditions.

6.3. Limitations of the study

The study was constrained by the available hardware, which necessitated the use of mixed precision training (combining float16 and float32 data types) to manage memory limitations. This might have impacted the performance of the cycle GAN, making it difficult to determine whether the model was performing at its full potential. Future studies could benefit from using hardware that allows for higher precision training to ensure optimal model performance. It also found that the cycle GAN was particularly sensitive to small changes in the hyperparameter settings, which required extensive tuning to achieve stable results. However, due to time constraints, only a limited range of hyperparameters could be explored. While significant improvements to the models performance were achieved by lowering the learning rate and increasing the lambda value, a more exhaustive hyperparameter search could potentially improve the model's performance. Additionally, the complexity of the cycle GAN architecture, with multiple loss functions and training strategies, added another layer of difficulty in achieving consistent results.

For training the conditional GAN, synthetic fog images were generated using a basic white cloud layer, which may not have accurately captured the complexity and variability of real-world fog. This simplified approach to generating training data might have contributed to the model's reduced performance on darker or colorless fog images. Incorporating more realistic data augmentation techniques, such as variable density and color fog simulation, could have provided a more comprehensive training experience for the model.

6.4. Future Work

Future research should focus on enhancing the robustness of GANs for dehazing under diverse and challenging conditions. One potential direction is to incorporate adaptive learning techniques that allow models to dynamically adjust to different fog densities and color variations. Additionally, experimenting with hybrid models that combine the strengths of both conditional GANs and Cycle Gans could potentially mitigate the individual limitations observed in this study, such as color distortion or reduced performance in darker haze. An example of this is the curl GAN, which combined paired and unpaired training data to improve generalization. Finally, expanding the scope

of testing with larger and more varied datasets would provide a more comprehensive understanding of how these models perform in real-world scenarios, further supporting their potential deployment in autonomous vehicle systems.

While GANs show that they can be considered as an approach to enhance pedestrian detection in severe weather in real world scenarios, cameras are not the only equipment available in an autonomous vehicle for this task. A multimodal approach utilizing all available data gathered by the autonomous vehicle may provide the best results and maximize the chances of detecting a pedestrian. Further research can investigate how this approach can be used alongside data from other sources such as thermal imaging or Lidar, to improve the reliability of the system and more accurately detect pedestrians through weather occlusion

6.5. Summary of Final Thoughts

This study explored the application of conditional GANs and cycle Gans for dehazing images to improve pedestrian detection in foggy conditions, providing valuable insights into their relative effectiveness, strengths, and limitations. The results demonstrated that both models significantly enhanced the average precision of pedestrian detection compared to unprocessed foggy images, with the conditional GAN achieving slightly better performance in terms of both accuracy and speed. However, neither model consistently produced high-quality dehazed images under all conditions, revealing specific challenges such as color distortion in the cycle GAN and sensitivity to fog density and color in the conditional GAN.

While the conditional GAN emerged as the more suitable model for real world deployment due to its faster processing speed and better balance between precision and recall, the study also highlighted areas where both models could be improved. The limitations encountered, including dataset constraints, hardware restrictions, and the sensitivity of the models to various training conditions, suggest that there is still considerable scope for further research and refinement.

The findings of this study contribute to the broader understanding of how GAN based models can be utilized for image enhancement in autonomous vehicle systems, particularly under challenging weather conditions. They highlight the importance of selecting the right model architecture based on the specific requirements of the application, balancing accuracy, speed, and robustness to diverse conditions.

In conclusion, while this research has demonstrated the potential of both cycle GANs and conditional GANs for dehazing in pedestrian detection, it also emphasizes the need for continued research. Future efforts should focus on developing more resilient models, refining training methodologies, and expanding datasets to better reflect real world scenarios. By addressing these challenges, we will gain a better understanding of the full potential of GAN based de-hazing for autonomous vehicle safety and performance.

References

Chow, T.-Y., Lee, K.-H., Chan, K.-L. (2023) Detection of Targets in Road Scene Images Enhanced Using Conditional GAN-Based Dehazing Model. Available From: https://doi.org/10.3390/app13095326

Graffieti, G., Maltoni, D., 2021. Artifact-Free Single Image Defogging. Available From: https://doi.org/10.3390/atmos12050577

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). Available From: https://arxiv.org/abs/1406.2661

Gupta, H., Kotlyar, O., Andreasson, H., Lilienthal, A. (2024) "Robust Object Detection in Challenging Weather Conditions," 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Generative Adversarial Networks. Available From: https://ieeexplore-ieee-org.uniessexlib.idm.oclc.org/document/10483822

Kim, J., Lee, G., Lee, J., Yuen, K., Kim, J. (2022). Determinants of personal concern about autonomous vehicles, Cities, Available From: https://doi.org/10.1016/j.cities.2021.103462.

Li, G., Yang, Y., Qu, X. (2020) Deep Learning Approaches on Pedestrian Detection in Hazy Weather, IEEE Transactions on Industrial Electronics. Available From: https://ieeexplore.ieee.org/document/8880634

Mirza, M., Osindero, S. (2014). Conditional generative adversarial nets. Ithaca: Available From: https://login.uniessexlib.idm.oclc.org/login?url=https://www.proquest.com/working-papers/conditional-generative-adversarial-nets/docview/2084631926/se-2

Ogunrinde, I., Bernadin, S., (2021). A Review of the Impacts of Defogging on Deep Learning-Based Object Detectors in Self-Driving Cars, Available From: https://doi.org/10.1109/southeastcon45413.2021.9401941

Pao, W., Howorth, J., Li, L., Agelin-Chaab, M., Roy, L., Knutzen, J., Baltazar-y-Jimenez, A., Muenker, K. (2024). Investigation of Automotive LiDAR Vision in Rain from Material and Optical Perspectives. Available From: https://doi.org/10.3390/s24102997

Parseh, M., Asplund, F. (2022). New needs to consider during accident analysis: Implications of autonomous vehicles with collision reconfiguration systems, Accident Analysis & Prevention, Available From: https://doi.org/10.1016/j.aap.2022.106704.

Razzok, M., Badri, A., Mourabit, I., Ruichek, Y., Sahel, A. (2023) Pedestrian detection under weather conditions using conditional generative adversarial network. Available From: https://ijai.iaescore.com/index.php/IJAI/article/view/22442

Siddiqua, M., Brahim Belhaouari, S., Akhter, N., Zameer, A., Khurshid, J. (2023). MACGAN: An All-in-One Image Restoration Under Adverse Conditions Using Multidomain Attention-Based Conditional GAN. IEEE Access 11, Available From: https://doi.org/10.1109/access.2023.3289591

Sun, Z., Zhang, Y., Bao, F., Shao, K., Liu, X., Zhang, C. (2021). ICycleGAN: Single image dehazing based on iterative dehazing model and CycleGAN, Computer Vision and Image Understanding, Available From: https://doi.org/10.1016/j.cviu.2020.103133.

Tapiro, H., Wyman, A., Borowsky, A., Petzoldt, T., Wang, X., & Hurwitz, D. S. (2022). Automated Vehicle Failure: The First Pedestrian Fatality and Public Perception. Transportation Research Record, 2676(8), 198-208. Available From: https://doi-org.uniessexlib.idm.oclc.org/10.1177/03611981221083297

Teeti, I., Musat, V., Khan, S., Rast, A., Cuzzolin, F., Bradley, A. (2023) Vision in adverse weather: Augmentation using CycleGANs with various object detectors for robust perception in autonomous racing. Available From: https://arxiv.org/pdf/2201.03246

World Health Organization (2023). Global Status On Road Safety 2023. Available From: https://www.who.int/teams/social-determinants-of-health/safety-and-mobility/global-status-report-on-road-safety-2023

Yang, H., Carballo, A., Zhang, Y., Takeda, K. (2023) Framework for Generation and Removal of Multiple Types of Adverse Weather from Driving Scene Images. Available From: https://www.mdpi.com/1424-8220/23/3/1548

Zhang, J. (2023) Traffic sign defogging based on conditional adversarial neural network pix2pixHD, Information and Communication Engineering (NNICE), Available From: https://ieeexplore.ieee.org/abstract/document/10105664

Zhu, J., Park, T., Isola, P., Efros, A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks, *2017 IEEE International Conference on Computer Vision (ICCV)*, Available From: https://ieeexplore-ieee-org.uniessexlib.idm.oclc.org/document/8237506

Bibliography

Tensorflow. (2024). pix2pix: Image-to-image translation with a conditional GAN. Available from: https://www.tensorflow.org/tutorials/generative/pix2pix

- This article was used alongside tensorflow documentation as a starting point for the conditional GAN model

Tensorflow. (2024). CycleGAN. Available from: https://www.tensorflow.org/tutorials/generative/cyclegan

- This article was used alongside tensorflow documentation as a starting point for the cycle GAN model

Appendix

Appendix A: Code

The full code for this project can be found on github: https://github.com/Nicholas-Bandy/eportfolio/tree/main/Dissertation/Code

```python
@tf.function
def train_step(input_image, target, step):
  """This function is a single training step for both the generator and discriminator, updating their weights based on the computed gradients and logging the results for analysis."""
  target, input_image = random_jitter(target, input_image)
  # Record the operations for automatic differentiation
  with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
    # Generate output images from the input images using the generator
    gen_output = generator(input_image, training=True)

    # Get the discriminator's output for real and generated images
    disc_real_output = discriminator([input_image, target], training=True)
    disc_generated_output = discriminator([input_image, gen_output], training=True)

    # Calculate the generator's total loss, GAN loss, and L1 loss
    gen_total_loss, gen_gan_loss, gen_l1_loss = generator_loss(disc_generated_output, gen_output, target)

    # Calculate the discriminator's loss
    disc_loss = discriminator_loss(disc_real_output, disc_generated_output)

    # Check for NaNs or Infs in the loss values
    tf.debugging.check_numerics(gen_total_loss, "Generator total loss has NaNs or Infs")
    tf.debugging.check_numerics(disc_loss, "Discriminator loss has NaNs or Infs")

  # Compute gradients for generator and discriminator
  generator_gradients = gen_tape.gradient(gen_total_loss, generator.trainable_variables)
  discriminator_gradients = disc_tape.gradient(disc_loss, discriminator.trainable_variables)

  # Apply gradients to update model weights
  generator_optimizer.apply_gradients(zip(generator_gradients, generator.trainable_variables))
  discriminator_optimizer.apply_gradients(zip(discriminator_gradients, discriminator.trainable_variables))
```

Figure A1: Conditional GAN training step function

```python
def generator():
    """This function defines the generator architecture. a Unet architecture is used, consisting of 8 downsample layers, 7 upsample layers, and a final conv layer"""
    # Define the input shape for the generator model
    inputs = tf.keras.layers.Input(shape=[720, 1280, 3])

    down_stack = [
      downsample(64, 4, apply_batchnorm=False),  # (batch_size, 640, 360, 64)
      downsample(128, 4),  # (batch_size, 320, 180, 128)
      downsample(256, 4),  # (batch_size, 160, 90, 256)
      downsample(512, 4),  # (batch_size, 80, 45, 512)
      downsample(512, 4),  # (batch_size, 40, 23, 512)
      downsample(512, 4),  # (batch_size, 20, 12, 512)
      downsample(512, 4),  # (batch_size, 10, 6, 512)
      downsample(512, 4),  # (batch_size, 5, 3, 512)
    ]

    up_stack = [
      upsample(512, 4, apply_dropout=True),  # (batch_size, 10, 6, 1024)
      upsample(512, 4, apply_dropout=True),  # (batch_size, 20, 12, 1024)
      upsample(512, 4, apply_dropout=True),  # (batch_size, 40, 23, 1024)
      upsample(512, 4),  # (batch_size, 80, 45, 1024)
      upsample(256, 4),  # (batch_size, 160, 90, 512)
      upsample(128, 4),  # (batch_size, 320, 180, 256)
      upsample(64, 4),  # (batch_size, 640, 360, 128)
    ]

    # Initialize weights for the last layer
    initializer = tf.random_normal_initializer(0., 0.02)

    # Last layer to produce the output imag
    last = tf.keras.layers.Conv2DTranspose(3, 4,strides=2,padding='same',kernel_initializer=initializer, activation='tanh')  # (batch_size, 720, 1280, 3)

    x = inputs #input image

    # Downsampling through the model
    skips = [] #list for skip connections
    for i, down in enumerate(down_stack):
      x = down(x) # Apply each downsampling layer
      #print(f"Downsample layer {i} output shape:", x.shape)  # Print shape of x after each downsample layer
      skips.append(x) # Store the output of each downsampling layer

    skips = reversed(skips[:-1])  # Reverse the list of skip connections (excluding the last one)

    # Upsampling and establishing the skip connections
    for i, (up, skip) in enumerate(zip(up_stack, skips)):
      x = up(x) # Apply each upsampling layer
      if x.shape[1:3] != skip.shape[1:3]:  # Check height and width dimensions
          x = ResizeLayer(target_size=[skip.shape[1], skip.shape[2]])(x) # Resize to match the skip connection
      #print(f"Upsample layer {i} output shape:", x.shape)  # Print shape of x after each upsample layer
      #print(f"Skip connection {i} shape:", skip.shape)     # Print shape of skip connection tensor
      x = tf.keras.layers.Concatenate()([x, skip]) # Concatenate with the skip connection
      #print(f"After concatenation {i} shape:", x.shape)    # Print shape of x after concatenation

    x = last(x) # Apply the last layer to produce the final output

    return tf.keras.Model(inputs=inputs, outputs=x)
```

Figure A2: Conditional GAN generator function

```python
def discriminator():
  """This function is used to classify pairs of images as real or fake, based on how well the input matches the target"""

  # Initialize weights for the model
  initializer = tf.random_normal_initializer(0., 0.02)

  # Input layers for the discriminator
  inp = tf.keras.layers.Input(shape=[720, 1280, 3], name='foggy_image')
  tar = tf.keras.layers.Input(shape=[720, 1280, 3], name='clear_image')

  # Concatenate the foggy and clear images along the channel axis
  x = tf.keras.layers.concatenate([inp, tar])  # (batch_size, 256, 256, channels*2)

  # Apply downsampling layers with batch normalization and LeakyReLU activations
  down1 = downsample(64, 4, False)(x)  # (batch_size, 128, 128, 64)
  down2 = downsample(128, 4)(down1)  # (batch_size, 64, 64, 128)
  down3 = downsample(256, 4)(down2)  # (batch_size, 32, 32, 256)

  # Zero-padding to maintain spatial dimensions for the convolutional layer
  zero_pad1 = tf.keras.layers.ZeroPadding2D()(down3)  # (batch_size, 34, 34, 256)

  # Convolutional layer to increase the depth of feature maps
  conv = tf.keras.layers.Conv2D(512, 4, strides=1,kernel_initializer=initializer,use_bias=False)(zero_pad1)  # (batch_size, 31, 31, 512)

  # Batch normalization to stabilize training
  batchnorm1 = tf.keras.layers.BatchNormalization()(conv)

  # Apply LeakyReLU activation to introduce non-linearity
  leaky_relu = tf.keras.layers.LeakyReLU()(batchnorm1)

  # Further zero-padding before final convolutional layer
  zero_pad2 = tf.keras.layers.ZeroPadding2D()(leaky_relu)  # (batch_size, 33, 33, 512)

  # Final convolutional layer to output a single-channel classification
  last = tf.keras.layers.Conv2D(1, 4, strides=1, kernel_initializer=initializer)(zero_pad2)  # (batch_size, 30, 30, 1)

  return tf.keras.Model(inputs=[inp, tar], outputs=last)
```

Figure A3: Conditional GAN discriminator function

```python
@tf.function
def train_step(real_x, real_y):
    """This function performs one step of training the model, which consists of 2 generators and 2 discriminators"""

    real_x = random_jitter(real_x)
    real_y = random_jitter(real_y)
    with tf.GradientTape(persistent=True) as tape:
       # Generate fake images
        fake_y = generator_g(real_x, training=True)
        fake_x = generator_f(real_y, training=True)

        # Reconstruct images
        cycled_x = generator_f(fake_y, training=True)
        cycled_y = generator_g(fake_x, training=True)

        # Identity mappings
        same_x = generator_f(real_x, training=True)
        same_y = generator_g(real_y, training=True)

        # Discriminator outputs
        disc_real_x = discriminator_x(real_x, training=True)
        disc_fake_x = discriminator_x(fake_x, training=True)
        disc_real_y = discriminator_y(real_y, training=True)
        disc_fake_y = discriminator_y(fake_y, training=True)

        # Losses for generators
        gen_g_loss = generator_loss(disc_fake_y)
        gen_f_loss = generator_loss(disc_fake_x)
        cycle_loss = calc_cycle_loss(real_x, cycled_x) + calc_cycle_loss(real_y, cycled_y)
        identity_loss_x = identity_loss(real_x, same_x)
        identity_loss_y = identity_loss(real_y, same_y)

        total_gen_g_loss = gen_g_loss + cycle_loss + identity_loss_y
        total_gen_f_loss = gen_f_loss + cycle_loss + identity_loss_x

        # Losses for discriminators
        disc_x_loss = discriminator_loss(disc_real_x, disc_fake_x)
        disc_y_loss = discriminator_loss(disc_real_y, disc_fake_y)

    # Calculate gradients for generators and discriminators
    generator_g_gradients = tape.gradient(total_gen_g_loss, generator_g.trainable_variables)
    generator_f_gradients = tape.gradient(total_gen_f_loss, generator_f.trainable_variables)
    discriminator_x_gradients = tape.gradient(disc_x_loss, discriminator_x.trainable_variables)
    discriminator_y_gradients = tape.gradient(disc_y_loss, discriminator_y.trainable_variables)

    # Apply gradients to update weights
    generator_g_optimizer.apply_gradients(zip(generator_g_gradients, generator_g.trainable_variables))
    generator_f_optimizer.apply_gradients(zip(generator_f_gradients, generator_f.trainable_variables))
    discriminator_x_optimizer.apply_gradients(zip(discriminator_x_gradients, discriminator_x.trainable_variables))
    discriminator_y_optimizer.apply_gradients(zip(discriminator_y_gradients, discriminator_y.trainable_variables))
```

Figure A4: Cycle GAN training step function

```python
def resnet_generator():
    """Creates a ResNet-based generator model for the CycleGAN. The generator consists of  an initial convolution layer,
    downsampling layers, multiple residual blocks, upsampling layers, final convolution layer."""

    initializer = tf.random_normal_initializer(0., 0.02)

    # Define the input tensor for the generator
    inputs = tf.keras.layers.Input(shape=[720, 1280, 3])  # Define the input tensor

    x = inputs

    # Initial Convolution Layer
    x = tf.keras.layers.Conv2D(64, 7, padding='same', kernel_initializer=initializer)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.ReLU()(x)

    # Downsampling layers
    x = tf.keras.layers.Conv2D(128, 3, strides=2, padding='same', kernel_initializer=initializer)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.ReLU()(x)

    x = tf.keras.layers.Conv2D(256, 3, strides=2, padding='same', kernel_initializer=initializer)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.ReLU()(x)

    # Residual blocks
    for _ in range(12 ):
        x = resnet_block(x, 256) # Apply 12 residual blocks with 256 filters each

    # Upsampling layers
    x = tf.keras.layers.Conv2DTranspose(128, 3, strides=2, padding='same', kernel_initializer=initializer)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.ReLU()(x)

    x = tf.keras.layers.Conv2DTranspose(64, 3, strides=2, padding='same', kernel_initializer=initializer)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.ReLU()(x)

    # Final Convolution Layer, use 'tanh' activation to produce outputs in the range [-1, 1]
    x = tf.keras.layers.Conv2D(OUTPUT_CHANNELS, 7, padding='same', kernel_initializer=initializer, activation='tanh')(x)

    return tf.keras.Model(inputs=inputs, outputs=x)
```

Figure A5: Cycle GAN resnet generator function

```python
def resnet_block(x, filters, size=3):
    """A residual block for a ResNet-based generator."""
    initializer = tf.random_normal_initializer(0., 0.02)

    # Save the input tensor to add it back after the convolutions
    res = x

    # First convolution layer in the residual block
    x = tf.keras.layers.Conv2D(filters, size, padding='same', kernel_initializer=initializer)(x)

    # Apply Batch Normalization to stabilize and accelerate training
    x = tf.keras.layers.BatchNormalization()(x)

    # Apply ReLU activation to introduce non-linearity
    x = tf.keras.layers.ReLU()(x)

    # Second convolution layer in the residual block
    x = tf.keras.layers.Conv2D(filters, size, padding='same', kernel_initializer=initializer)(x)

    # Apply Batch Normalization again
    x = tf.keras.layers.BatchNormalization()(x)

    # Add the input tensor (residual connection) to the output of the second convolution
    x = tf.keras.layers.Add()([x, res])

    # Apply ReLU activation to introduce non-linearity
    return tf.keras.layers.ReLU()(x)
```

Figure A6: Cycle GAN resnet block function

```python
def discriminator():
    """This function is used to classify pairs of images as real or fake, based on how well the input matches the target"""

    initializer = tf.random_normal_initializer(0., 0.02)  # Initialize weights for the model

    #input layer
    x = tf.keras.layers.Input(shape=[None, None, 3], name='input_image')

      # Apply downsampling layers with batch normalization and LeakyReLU activations
    down1 = downsample(64, 4, False)(x)  # (batch_size, 640, 360, 64)
    down2 = downsample(128, 4)(down1)  # (batch_size, 320, 180, 128)
    down3 = downsample(256, 4)(down2)  # (batch_size, 160, 90, 256)
    down4 = downsample(512, 4)(down3)  # (batch_size, 80, 45, 512)
    down5 = downsample(512, 4)(down4)  # (batch_size, 40, 23, 512)
    down6 = downsample(512, 4)(down5)  # (batch_size, 20, 12, 512)
    down7 = downsample(512, 4)(down6)  # (batch_size, 10, 6, 512)

    # Zero-padding to maintain spatial dimensions for the convolutional layer
    zero_pad1 = tf.keras.layers.ZeroPadding2D()(down7)  # (batch_size, 12, 8, 512)

    # Convolutional layer to increase the depth of feature maps
    conv = tf.keras.layers.Conv2D(512, 4, strides=1,kernel_initializer=initializer,use_bias=False)(zero_pad1)  # (batch_size, 9, 5, 512)

    # Batch normalization to stabilize training
    batchnorm1 = tf.keras.layers.BatchNormalization()(conv)

    # Apply LeakyReLU activation to introduce non-linearity
    leaky_relu = tf.keras.layers.LeakyReLU()(batchnorm1)

    # Further zero-padding before final convolutional layer
    zero_pad2 = tf.keras.layers.ZeroPadding2D()(leaky_relu)  # (batch_size, 11, 7, 512)

    # Final convolutional layer to output a single-channel classification
    last = tf.keras.layers.Conv2D(1, 4, strides=1,kernel_initializer=initializer)(zero_pad2)  # (batch_size, 8, 4, 1)

    return tf.keras.Model(inputs=x, outputs=last)
```

Figure A7: Cycle GAN discriminator function

Appendix B: Network Structures

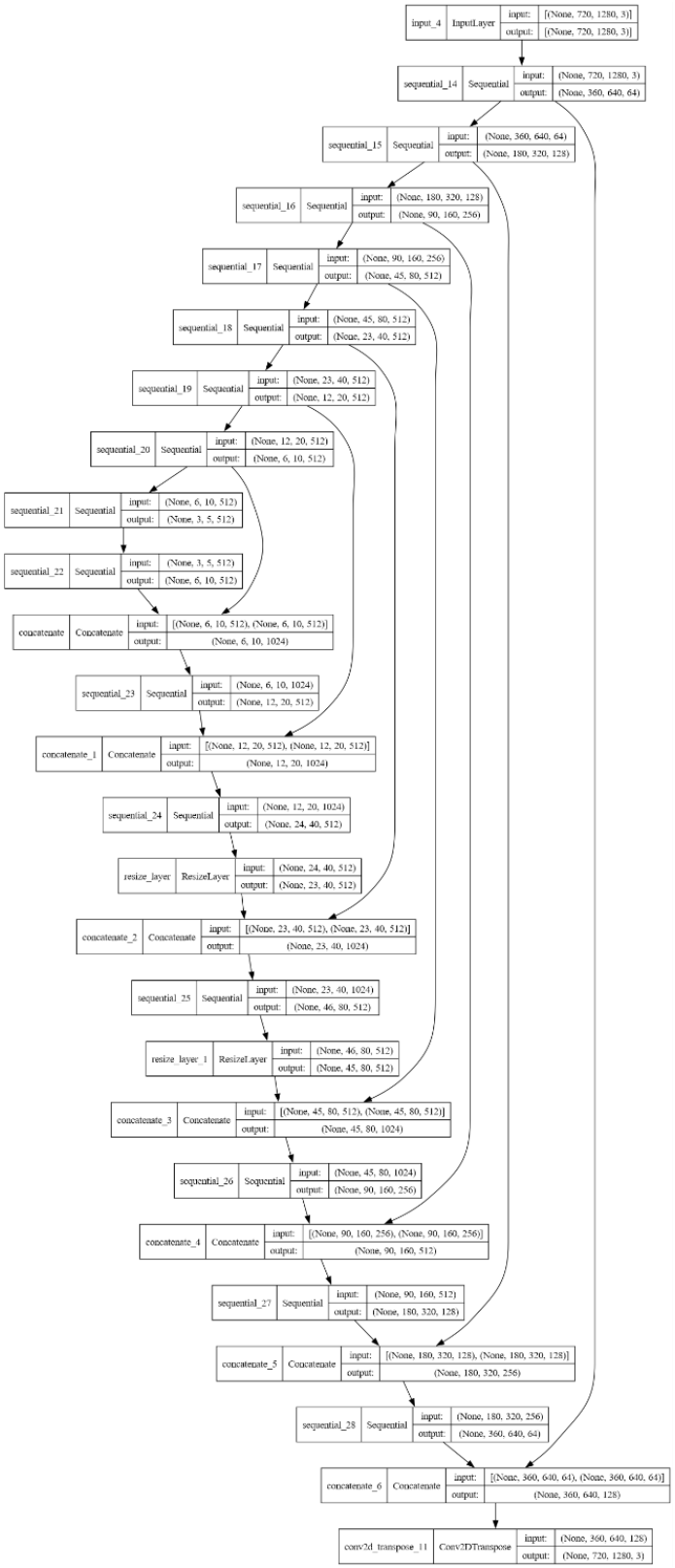

Figure B1: Conditional GAN generator structure

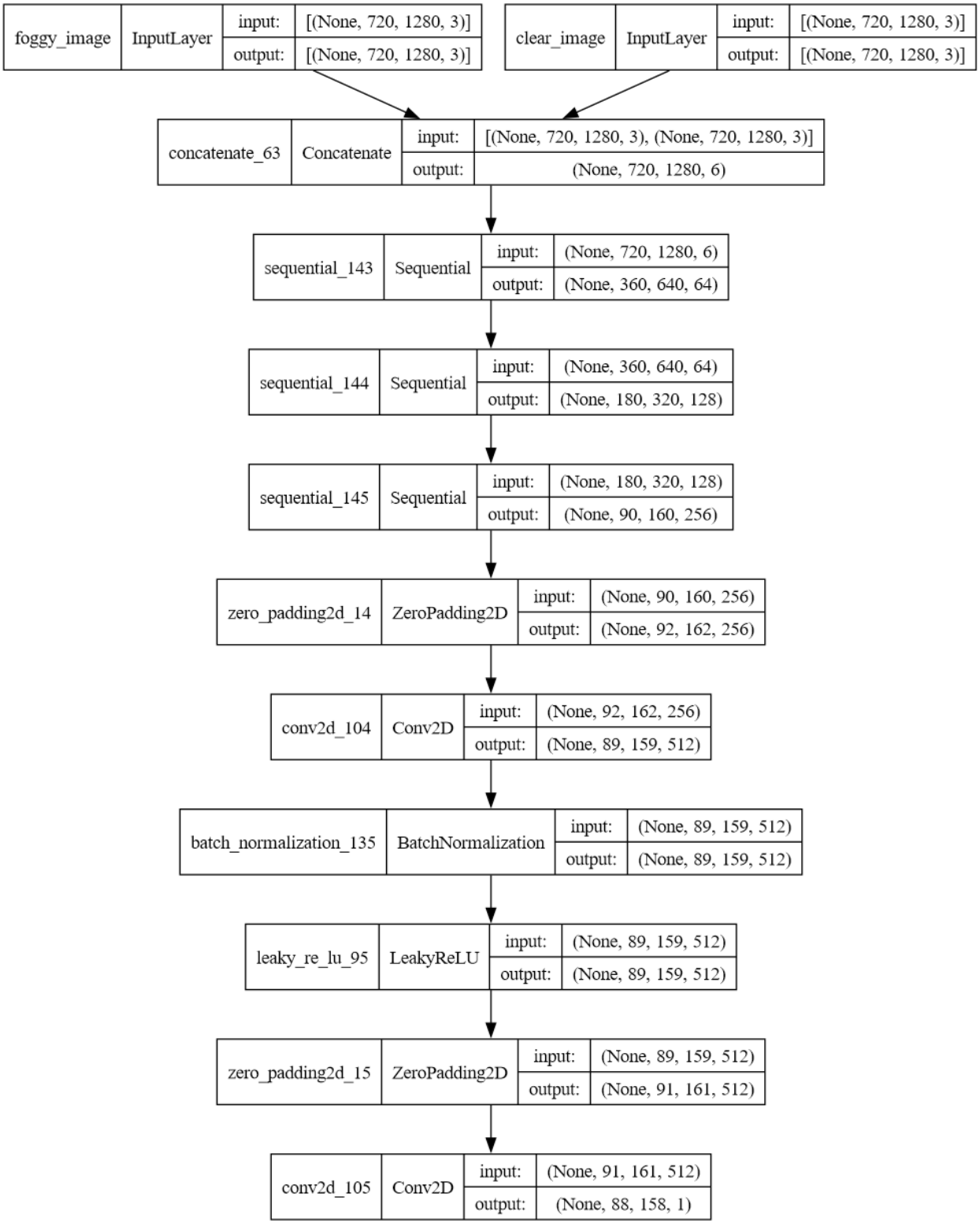

Figure B2: Conditional GAN discriminator structure

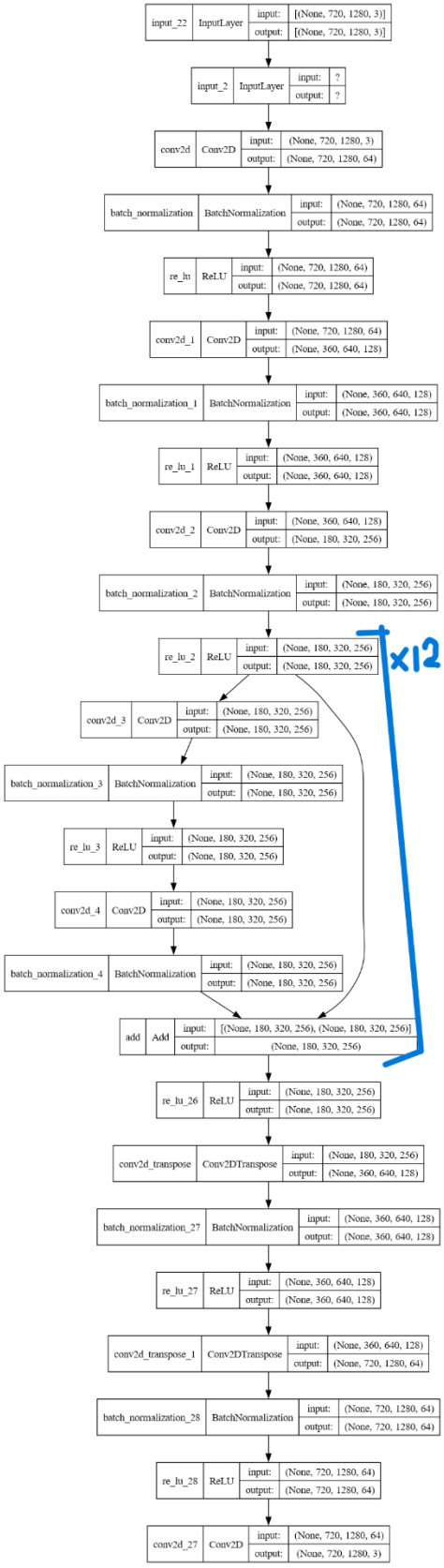

Figure B3: Cycle GAN generator structure. Resnet block highlighted in blue, repeats 12 times

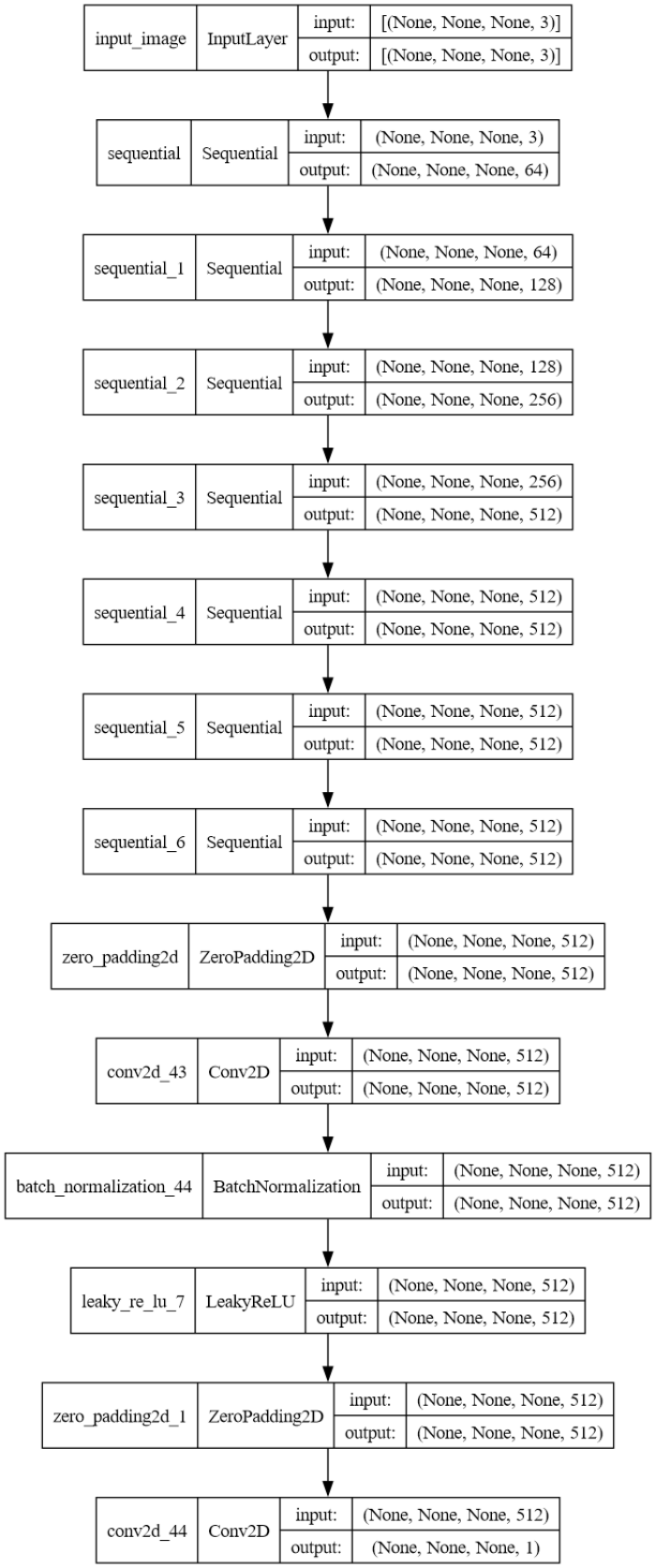

Figure B4: Cycle GAN discriminator structure