TU Delft

Unpaired day-night domain adaptation using CycleGANs combined with CIConv

Bachelor Thesis Computer Science & Engineering

Student: Thomas Streefkerk

Date: 28 January 2022

Supervisor: Attila Lengyel

Responsible professor: Dr. Jan van Gemert

EEMCS, Delft University of Technology, The Netherlands

{t.s.streefkerk}@student.tudelft.nl, {a.lengyel, j.c.vangemert}@tudelft.nl

Abstract

CycleGANs [1] and CIConv [2] are both relatively new approaches to their respective applications. For CycleGANs this application is unpaired image-to-image domain adaptation and for CIConv this application is making images more robust to illumination changes. We investigate whether CycleGANs in combination with CIConv can be used to improve the day-night domain adaptation. The resulting images could then be used during the training of CNNs that can be found in self-driving cars. Attempts were made to get the CycleGANs in combination with CIConv to train in a stable manner. These attempts included a variety of hyperparameter combinations, a number of architecture alterations and training procedure adjustments, and most significantly two different loss functions. Both these loss functions apply a Cycle-Consistency Loss, one applies an additional Adversarial Loss [1] and the other an additional Wasserstein Distance and Gradient Penalty [3]. In this paper we show that CycleGANs with CIConv as the first layer in either the Discriminators or the Generators resulted in unstable training. We conclude that the root of the instability issues lies in the CIConv layer causing exploding gradients resulting in unsuccessful training of the model. Finally, we propose an adjustment to the CIConv layer which shows promise in resolving these issues for the architecture with CIConv in the Generators. However, no extensive testing has been done.

1 Introduction

Nowadays, we are seeing more and more self-driving cars on the roads. These cars rely on cameras and computer vision software to recognize and react to their environment. This software generally contains some sort of Convolutional Neural Network (CNN) [4] to perform the required image processing tasks.

Since CNNs are a type of Neural Network (NN), they require training and therefore training data. To ensure that the self-driving cars become capable of safely navigating the roads, this training data should contain data of several domains. More specifically, sufficient domains to encapsulate the differences in illumination caused by the time of day.

However, the gathering of the data for all these domains has proven a difficult task, especially for nighttime training data. This results in a lack of nighttime training data for the training of CNNs used by self-driving cars. Contrary, training data during the day is readily accessible. This makes a method for adapting the day domain data to night domain data appealing to the self-driving car industry.

One of the main challenges of this potential method of day-night domain adaptation is to learn the mapping between the illumination characteristics of the domains. Recent work on Color Invariant Convolution (CIConv) [2] has introduced a method for making images more robust to illumination changes making it an interesting candidate to assist in the day-night domain adaptation. CIConv works by adding a color invariant edge detector as a trainable layer to a Neural Network hereby transforming an image to a color invariant representation. As a result, images from either domain ran through a trained CIConv layer should be near duplicates.

In this research, we tackle the challenge of domain adaptation by combining CIConv with Generative Adversarial Networks (GANs), more specifically Cycle-Consistent Generative Adversarial Networks (CycleGANs) [1] and Cycle-Consistent Wasserstein Adversarial Networks with Gradient Penalty (CycleWGAN-GP) [3]. This is done in order to convert training data from the day domain to training data from the night domain. This data could then potentially be used to train CNNs on segmentation or other computer vision tasks. A (partially) successful completion of this research could be of great benefit to the GAN field as well as the self-driving car industry.

In order to carry out this research the following research question is posed: *What is the influence of CIConv in combination with CycleGANs on the generation of labeled training data in a domain for which only unlabeled data is available?*

In this paper we will make the following contributions:

- We show that the architecture without CIConv using either the CycleGAN loss function or the CycleWGAN-GP loss function trains in a stable manner and succeeds in the day-night domain adaptation.
- We show that the architectures with original CIConv in either the Discriminators or the Generators using either the CycleGAN loss function or the CycleWGAN-GP loss function results in unstable training due to exploding gradients and is therefore unsuccessful in the day-night domain adaptation.
- We propose an adjusted CIConv layer that, when added to the first layer of the Discriminators or the Generators in a CycleGAN, allows for stable training and shows promising preliminary results. It therefore has the potential to be of benefit during the day-night domain adaptation.
- We provide a code-base containing an adjusted CIConv layer and three CycleGAN architectures, namely: No CIConv, CIConv as the first layer of the Discriminators and CIConv as the first layer in the Generators. For these architectures two loss functions are available, namely the CycleGAN loss function [5] and the CycleWGAN-GP loss function [3].

2 Related Works

Generative Adversarial Networks (GANs) have shown remarkable results in the generation of fake images based on simple noise [6, 7]. A well known example is the generation of fake images of human faces. These faces are (almost) undifferentiable from real images.

GANs train using two opposing models, namely the Generator and the Discriminator. The Generator generates images that the Discriminator classifies as real or fake. The key to the

training of these models and thus the quality is called the Adversarial Loss. This loss ensures that the Generator becomes better at fooling the Discriminator and that the Discriminator becomes better at classifying images as either real or fake. As a result, Adversarial Loss allows for the generation of images that are (nearly) indistinguishable from real images [8]. GANs in itself are unable to perform domain adaptation, therefore we use Cycle-Consistent Generative Adversarial Networks (CycleGAN) and Cycle-Consistent Wasserstein Generative Adversarial Networks with Gradient Penalty (CycleWGAN-GP). These are extensions of the GAN architecture.

The stable training of GANs is known to be a challenging issue and is still an active area of research. This is due to the need for both the Generator and the Discriminator to learn at a similar rate. If this is not the case, typically the Discriminator will overpower the Generator with unsuccessful training as a result.

A frequently encountered training issue is the Vanishing Gradient problem. In this case the Generator fails to learn due to the Discriminator not providing enough information to do so. [9] proposes the Wasserstein loss which is designed to prevent vanishing gradients even when the Discriminator it trained sub-optimally. This loss is then further improved using Gradient Penalty [10, 11], which guarantees GAN convergence.

Another recurrent issue is the Exploding Gradient problem. This problem occurs when the updates to the weights during back-propagation become extremely large. As a result the network is not updating its weights as it should and therefore no improvements are made. [12] proposes the use of gradient clipping in order to prevent the Exploding Gradient problem and [10] proposes the Gradient Penalty as a substitution to gradient clipping.

We use an extension of the GAN architecture, namely CycleGAN. Additionally, we add upon this architecture by implementing a Wasserstein Distance and a Gradient Penalty. Furthermore, the method of Gradient Clipping in order to prevent Exploding Gradients has been implemented.

Domain Adaptation is used as an alternative to the frequently expensive and time-consuming process of gathering real labeled training data. Is is a sought after method for adapting an image from a source domain to a target domain. [8, 13, 14] use Conditional GANs to perform domain adaptation. [15] uses Coupled GANs (CoGANs) and [16] applies StyleGANs. Furthermore, [1, 17] utilize CycleGANs to perform domain adaptation. Of these architectures, we have implemented the CycleGAN architecture due to the others not having a Cycle-Consistency Loss in their loss functions.

Cycle-Consistent Adversarial Networks (CycleGANs) are an extension of the regular GAN architecture and are known to perform unpaired image-to-image translation [1, 17]. This entails translating an image from a source domain, to an image from a different target without a direct pairing between the source and the target domain. CycleGANs are able to perform this translation by introducing an extra type of loss to their loss function, namely Cycle-Consistency Loss. This loss ensures that images converted from one domain to another and then back, differ as little as possible, hereby preserving the structure of the original image.

Wasserstein Distance and Gradient Penalty are a variation of the regular loss function proposed by [9], in which careful maintaining of the equilibrium is no longer needed. The Wasserstein Distance allows for improved training stability due to a decreased reliance on network architecture. Additionally, the Gradient Penalty [10] helps in preventing both the Vanishing Gradient and the Exploding Gradient problem. We use both the Wasserstein Distance and the Gradient Penalty.

Color Invariance is achieved by making certain assumptions within physics-based reflection models allowing images to become invariant towards certain illumination features [18]. It has applications in wide variety of computer vision tasks. Recent works have applied color invariant transformations as a preprocessing step and show improved image segmentation performance [19, 20]. [2] introduces Color Invariant Convolution (CIConv), which uses the Kubelka-Munk theory for material reflections and applies it as a learnable layer in a neural network. We compare CycleGANs with and without the use of CIConv to discover whether CIConv can be of benefit during day-night domain adaptation.

3 Method

Our aim is to determine if the incorporation of Color Invariant Convolution (CIConv) (see section 3.2) in CycleGANs is beneficial for performing day-night domain adaptation. This is done by developing three architectures. One without CIConv (CG), see figure 1a, one with CIConv as the first layer in the Discriminators (CG_{disc}), see figure 1b and one with CIConv as the first layer in the Generators (CG_{gen}), see figure 1c. For these architectures we implemented two loss functions. The first applies the Adversarial Loss and Cycle-Consistency Loss proposed by [1] ($\mathcal{L}_{CycleGAN}$). The second swaps the Adversarial Loss for the Wasserstein Distance and introduces a Gradient Penalty according to [3] ($\mathcal{L}_{CycleWGAN_{GP}}$).

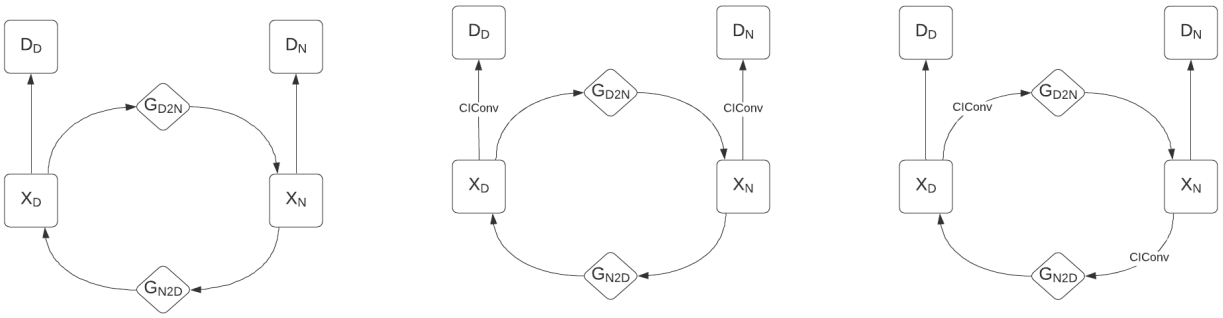
As can be seen in figures 1a, 1b and 1c, images from X_D (day domain) are converted via the Generator G_{D2N} (day-to-night), to images from X_N (night domain). The reverse also occurs, in this case images from X_N are converted via G_{N2D} (night-to-day) to images from X_D. We denote images from X_D that have been converted to images from X_N via a Generator G_{D2N} as $G_{D2N}(X_D)$, the reverse also holds. Consequently, images converted from one domain to the other and then back should resemble each other as closely as possible, hence: $G_{N2D}(G_{D2N}(X_D)) \approx X_D$.

Additionally, for CG, CG_{disc} and CG_{gen}, we have introduced two adversarial Discriminators, namely D_D and D_N. D_D classifies images from X_D as real or fake. Members of X_D could be either original members of X_D or converted members of X_N, so: $G_{N2D}(X_D)$.

Reaching our goal requires combining the developed architectures with a loss function and CIConv. First, the two loss functions and what they consist of are laid out. This is followed by an overview on CIConv.

3.1 Loss Functions

We apply two different loss functions, namely $\mathcal{L}_{CycleGAN}$ and $\mathcal{L}_{CycleWGAN_{GP}}$. These loss functions consist of a com-



(a) CycleGAN architecture without CIConv, referred to as *CG*

(b) CycleGAN architecture with CIConv in the Discriminator, referred to as CG_{disc}

(c) CycleGAN architecture with CIConv in the Generator, referred to as CG_{gen}

Figure 1: CycleGAN architectures without (left) and with (center and right) CIConv. X_D and X_N are the input images of the day and night domain, respectively. D_D and D_N are the Discriminators for the day and night domain, respectively. G_{D2N} and G_{N2D} are the day-to-night and night-to-day domain Generators, respectively.

bination of loss terms. These terms are described in the following section.

3.1.1 Adversarial Loss

We apply an adversarial loss to both the Discriminators and the Generators. For the day-to-night domain conversion ($X_D \rightarrow X_N$), this is done with the following equation adopted from [6]:

$$\mathcal{L}_{GAN}(G_{D2N}, D_N, X_D, X_N) = \mathbb{E}_{\mathrm{n}\sim \mathrm{p_N(n)}}[log(D_N(n))] + \mathbb{E}_{\mathrm{d}\sim \mathrm{p_D(d)}}[1 - log(D_N(n^*))]. \quad (1)$$

In the first part of equation 1 (before the + sign), we see that D_N assesses a real image from X_N. D_N aims to keep this value as high as possible. If successful, real images will be assigned a value close to 1.0.

In the second part of equation 1 (after the + sign), D_N assesses a fake image n^* from X_N generated by G_{D2N}. D_N aims to assess these images as accurately as possible. If successful, by assigning them values close to 0.0. Due to the $[1-]$ term, the Discriminator will again attempt to keep the part of the equation after the + sign as large as possible.

Contrary, G_{D2N} aims to keep the second part of equation 1 as low as possible. If it succeeds, this would mean that G_{D2N} fooled D_N into assessing a fake image as if it were real. Additionally, if G_{D2N} is capable of fooling D_N in some cases, this results in a lower certainty by D_N in the first part of equation 1.

D_N and G_{D2N} are both trying to achieve their own opposing goals in terms of what should happen to the value of equation 1. D_N wants it to be as high as possible and G_{D2N} wants it to be as low as possible. They are therefore competing in a minimax game which can be formulated as: $\min_{G_{D2N}} \max_{D_N} \mathcal{L}_{GAN}(G_{D2N}, D_N, X_D, X_N)$.

Furthermore, an altered version of this minimax game also occurs for the reverse domain adaptation, namely: $X_N \rightarrow X_D$. In this case the minimax game formulation becomes: $\min_{G_{N2D}} \max_{D_D} \mathcal{L}_{GAN}(G_{N2D}, D_D, X_N, X_D)$.

3.1.2 Wasserstein Distance

We apply a Wasserstein Distance as the adversarial part of the loss function for both the Generators and the Discriminators[1]. For the day-night domain conversion ($X_D \rightarrow X_N$) this results in the following equation adopted from [3]:

$$\mathcal{L}_{WGAN}(G_{D2N}, D_N, X_D, X_N) = \mathbb{E}_{\mathrm{n}\sim \mathrm{p_N(n)}}[D_N(n)] - \mathbb{E}_{\mathrm{d}\sim \mathrm{p_D(d)}}[D_N(n^*)]. \quad (2)$$

The first part of equation 2 (before the - sign) represents the mean of the prediction of D_N on real images from X_N. The second part of equation 2 (after the - sign) represents the mean of the prediction of D_N on fake images n^* from X_N generated by G_{D2N}. They are subtracted to arrive at the Wasserstein Distance.

The Wasserstein Distance is a value for how closely the images resemble each other. The Generators aim to obtain a larger score from the Discriminators. The Discriminators aim to provide a larger score for real images and a lower score for fake images. To describe their behavior the following minimax game is formulated: $\min_{G_{D2N}} \max_{D_N} \mathcal{L}_{WGAN}(G_{D2N}, D_N, X_D, X_N)$.

An altered version of this minimax game also holds for the reverse domain adaptation, namely: $X_N \rightarrow X_D$. This can then be described as $\min_{G_{N2D}} \max_{D_D} \mathcal{L}_{WGAN}(G_{N2D}, D_D, X_N, X_D)$.

3.1.3 Gradient Penalty

We apply a Gradient Penalty which penalizes the norm of the gradient of the discriminators. It is defined by the following equation:

$$\mathcal{L}_{\mathrm{GP}}(D_N, X_N) = \lambda_{GP} \mathbb{E}_{\widetilde{x} \sim \mathbb{P}_{\widetilde{x}}}[(||\nabla_{\widetilde{x}} D(\widetilde{x})|| - 1)^2], \quad (3)$$

where λ_{GP} is the Gradient Penalty coefficient, it allows for control over the relative influence of the Gradient Penalty.

[1] Formally, when applying Wasserstein distances we use a Critic instead of a Discriminator. For simplicity, we only speak of Discriminators in this paper.

$\mathbb{P}_{\widetilde{x}}$ is the sampling distribution between pairs of points sampled from the data distribution of real images and generated images [3]. By applying equation 3, we convert the regular WGAN loss function to an improved WGAN-GP loss function.

Apart from the Discriminator D_N and the domain X_N, equation 3 is also applied for the Discriminator D_D and the domain X_D. The result being $\mathcal{L}_{GP}(D_D, X_D)$.

3.1.4 Cycle-Consistency Loss

We apply Cycle-Consistency Loss to ensure that the structure of the image between conversions stays intact. This is represented with the following equation adopted from [1]:

$$\begin{aligned}\mathcal{L}_{\text{cycle}}(G_{D2N}, G_{N2D}) = \mathbb{E}_{\text{d}\sim\text{p}_{\text{data}}(\text{d})}[||G_{N2D}(G_{D2N}(d)) - d||] \\ + \mathbb{E}_{\text{n}\sim\text{p}_{\text{data}}(\text{n})}[||G_{D2N}(G_{N2D}(n)) - n||].\end{aligned} \tag{4}$$

In the first term of equation 4 (before the + sign) the absolute value of the difference between images from X_D converted to X_N and then back, and images from X_D is taken. A similar operation is done in the second part of the equation. Now, with the Generators and the domains flipped.

Cycle-Consistency Loss is used because Adversarial Loss alone is not capable of generating the desired images. Without Cycle-Consistency structural properties of images would go lost between image conversions.

3.1.5 Combined Loss Functions

By combining the losses described in sections 3.1.1, 3.1.2, 3.1.3 and 3.1.4 we arrive at two loss functions, namely $\mathcal{L}_{CycleGAN}$ and $\mathcal{L}_{CycleWGAN_{GP}}$. These losses are further defined in the following sections.

3.1.5.1 CycleGAN Loss Function

From equations 1 and 4 we obtain the following combined loss function for $\mathcal{L}_{CycleGAN}$ adopted from [5]:

$$\begin{aligned}\mathcal{L}_{CycleGAN}(\mathcal{G}, \mathcal{D}, \mathcal{X}) = \mathcal{L}_{GAN}(G_{D2N}, D_N, X_D, X_N) \\ + \mathcal{L}_{GAN}(G_{N2D}, D_D, X_N, X_D) \\ + \lambda\mathcal{L}_{cycle}(G_{D2N}, G_{N2D})\end{aligned} \tag{5}$$

or $\mathcal{L}_{CycleGAN}$ for short. Here, $\mathcal{G}$, $\mathcal{D}$ and $\mathcal{X}$ are the Generators G_{D2N} and G_{N2D}, the Discriminators D_N and D_D, and the domains X_N and X_D, respectively. Equation 5 combines two Adversarial Loss terms, one for each Generator and Discriminator pair, and a Cycle-Consistency Loss term. Additionally, notice the multiplication with λ before the Cycle-Consistency Loss term. This multiplication allows for control over the relative importance between the types of losses.

3.1.5.2 CycleWGAN-GP Loss Function

From equations 2, 3 and 4, we arrive at a the following combined loss function for $\mathcal{L}_{CycleWGAN_{GP}}$ adopted from [3]:

$$\begin{aligned}\mathcal{L}_{CycleWGAN_{GP}}(\mathcal{G}, \mathcal{D}, \mathcal{X}) = \mathcal{L}_{WGAN}(G_{D2N}, D_N, X_D, X_N) \\ + \mathcal{L}_{WGAN}(G_{N2D}, D_D, X_N, X_D) \\ + \lambda_0\mathcal{L}_{cycle}(G_{D2N}, G_{N2D}) \\ + \lambda_1\mathcal{L}_{GP}(D_N, X_N) \\ + \lambda_2\mathcal{L}_{GP}(D_D, X_D)\end{aligned} \tag{6}$$

or $\mathcal{L}_{CycleWGAN_{GP}}$ for short. Here, $\mathcal{G}$, $\mathcal{D}$ and $\mathcal{X}$ are the Generators G_{D2N} and G_{N2D}, the Discriminators D_N and D_D, and the domains X_N and X_D, respectively. Additionally, λ_0 represents the Cycle-Consistency Loss coefficient and both λ_1 and λ_2 represent the Gradient Penalty coefficients.

Equation 6 combines two Wasserstein Distances, one for each Generator and Discriminator pair, a Cycle-Consistency Loss and two Gradient Penalties, one for each Discriminator.

3.2 Color Invariant Convolution (CIConv)

CIConv is a method for making images more robust to illumination changes and has been introduced in recent work [2]. It works by adding a color invariant edge detector as a trainable layer to a CNN.

In this work, we apply the *W* version of the CIConv layer since this was the most successful variant from [2]. This variant is defined in the following equation:

$$W = \sqrt{W_x^2 + W_{\lambda x}^2 + W_{\lambda\lambda x}^2 + W_y^2 + W_{\lambda y}^2 + W_{\lambda\lambda y}^2}, \tag{7}$$

where W_x, $W_{\lambda x}$ and $W_{\lambda\lambda x}$ are defined as follows:

$$W_x = \frac{E_x}{E}, W_{\lambda x} = \frac{E_{\lambda x}}{E}, W_{\lambda\lambda x} = \frac{E_{\lambda\lambda x}}{E} \tag{8}$$

and W_y, $W_{\lambda y}$ and $W_{\lambda\lambda y}$ are similarly defined. E_x, $E_{\lambda x}$ and $E_{\lambda\lambda x}$ are determined by first estimating E, E_λ and $E_{\lambda\lambda}$ based on the Gaussian color model and then convolving these estimations with a Gaussian derivative kernel [2].

Finally, CIConv is defined as:

$$CIConv(x, y) = \frac{log(CI^2(x, y, \sigma = 2^s) + \epsilon) - \mu_S}{\sigma_S}, \tag{9}$$

where CI is the color invariant choice, in our case W from equation 7. μ_S and σ_S are the sample mean and standard deviation over $log(CI^2 + \epsilon)$. ϵ is a small term added to prevent the log term from approaching $-\infty$ when its input approaches zero [2].

A result of using CIConv is that a three-channel image is converted to a one-channel image. When CIConv is used in the Discriminators, this has the consequence of them only classifying images as real or fake based on a one-channel image. Something similar is true when CIConv is used in the Generators. In this case a one-channel image is run through the Generator network and in the final layer it is converted back to a three-channel image.

4 Experiments

Our aim for this project is to determine whether Color Invariant Convolution (CIConv) (see section 3.2) is a useful addition to a CycleGAN performing day-night domain adaptation. To establish this, our first priority is to have the architectures train in a stable manner. To determine if stable training occurred, we ran the training sessions for 10 epochs. This provided us with enough information to conclude whether or not a setup was training stably. The following sections will make use of certain symbols of which the meaning can be found in table 1.

Symbol	Meaning
G_{D2N} or G_D2N	Day-to-Night Generator
G_{N2D} or G_N2D	Night-to-Day Generator
D_D or D_D	Day Discriminator
D_N or D_N	Night Discriminator
X_D	Day domain
X_N	Night domain
CG	CycleGAN without CIConv (figure 1a)
CG_{disc}	CycleGAN with CIConv in the first layer of the Discriminators (figure 1b)
CG_{gen}	CycleGAN with CIConv in the first layer of the Generators (figure 1c)
$\mathcal{L}_{CycleGAN}$	Loss function with adversarial losses and cycle-consistency losses (equation 5)
$\mathcal{L}_{CycleWGAN_{GP}}$	Loss function with Wasserstein distances, gradient penalties and cycle-consistency losses (equation 6)

Table 1: Overview of used symbols and their meaning.

4.1 Network Architectures

We propose three architectures for which the main implementation largely originates from [5]. These architectures consist of *CG*, CG_{disc} and CG_{gen}. We trained all three of these architectures with two different loss functions, namely $\mathcal{L}_{CycleGAN}$ and $\mathcal{L}_{CycleWGAN_{GP}}$.

Generator Layers Specification
Optional CIConv layer, variant W, k 3, initial scale 0.0
7×7 Conv-Norm-ReLu layer, 64 filters, stride 1
3×3 Conv-Norm-ReLu layer, 128 filters, stride 2
3×3 Conv-Norm-ReLu layer, 256 filters, stride 2
9 Residual blocks
3×3 ConvTrans-Norm-ReLu layer, 128 filters, stride 2
3×3 ConvTrans-Norm-ReLu layer, 64 filters, stride 2
7×7 Conv layer, 3 filters, stride 1

Table 2: Specification of the Generator layers. *Conv* represents a Convolutional layer, *ConvTrans* a Transpose Convolutional layer, *Norm* an Instance Normalization step and *ReLu* a ReLu activation function. Notice the optional first CIConv layer.

Tables 2 and 3 show the Generator and Discriminator layers specifications, respectively. They both have an optional first CIConv layer. By varying if we apply this layer, we create the three architectures. Additionally, the Discriminator has an extra Sigmoid activation function in case $\mathcal{L}_{CycleGAN}$ is used.

Furthermore, when $\mathcal{L}_{CycleGAN}$ is used we apply one-sided label smoothing. This entails that when the Discriminator losses on real images are lower than 0.1, we replace these loss values with a random number between 0.0 and 0.1. Additionally, we apply a five percent probabilistic label switching of images going into the Discriminators. Meaning that in five percent of the cases, we swap the labels of the real and fake images going into the Discriminators. Both these methods help steer training to an equilibrium between Generator and Discriminator performance, hereby improving training stability [21].

Further attempted architecture tweaks include gradient clipping of the CIConv layer scale parameter and applying batch normalization instead of instance normalization. However, we found these to show no performance increase and they are therefore not used in the final implementation.

Discriminator Layers Specification
Optional CIConv layer, variant W, k 3, initial scale 0.0
4×4 Conv-LeakyReLu layer, 64 filters, stride 2
4×4 Conv-Norm-LeakyReLu layer, 128 filters, stride 1
4×4 Conv-Norm-LeakyReLu layer, 256 filters, stride 1
4×4 Conv-Norm-LeakyReLu layer, 512 filters, stride 2
4×4 Conv layer, 1 filter, stride 1
If using $\mathcal{L}_{CycleGAN}$: Sigmoid

Table 3: Specification of the Discriminator layers. *Conv* represents a Convolutional layer, *Norm* an Instance Normalization step and *LeakyReLu* a LeakyReLu activation function. Notice the optional first CIConv layer and the final Sigmoid activation function in case $\mathcal{L}_{CycleGAN}$ is applied.

Hyperparameter	$\mathcal{L}_{CycleGAN}$	$\mathcal{L}_{CycleWGAN_{GP}}$
LR Discriminators	1e-5	2e-4
LR Generators	1e-5	2e-4
Batch size	2	2
CC coefficient	10	50
Gradient Penalty	-	10

Table 4: An overview of the hyperparameters used during the experiments for the two loss functions. LR: Learning rate, CC: Cycle-Consistency.

4.2 Training Details

Ensuring that the architectures trained in a stable manner was our first priority. A wide variety of hyperparameter combinations was attempted. For the upcoming experiments we used the hyperparameters in table 4. The hyperparameters under $\mathcal{L}_{CycleGAN}$ follow [1] and under $\mathcal{L}_{CycleWGAN_{GP}}$ follow [3]. The exception is the batch size, this was chosen to be as large as possible but was constrained by the allowed amount of GPU memory allocation. Additionally, we used the CityScapes dataset for the day domain and the Dark Zurich dataset for the night domain.

4.3 Training architectures with $\mathcal{L}_{CycleGAN}$

In this section, we discuss our findings when training *CG*, CG_{disc} and CG_{gen} with $\mathcal{L}_{CycleGAN}$ as the loss function. We compare the architectures based on their output images and on the evolution of their losses during training.

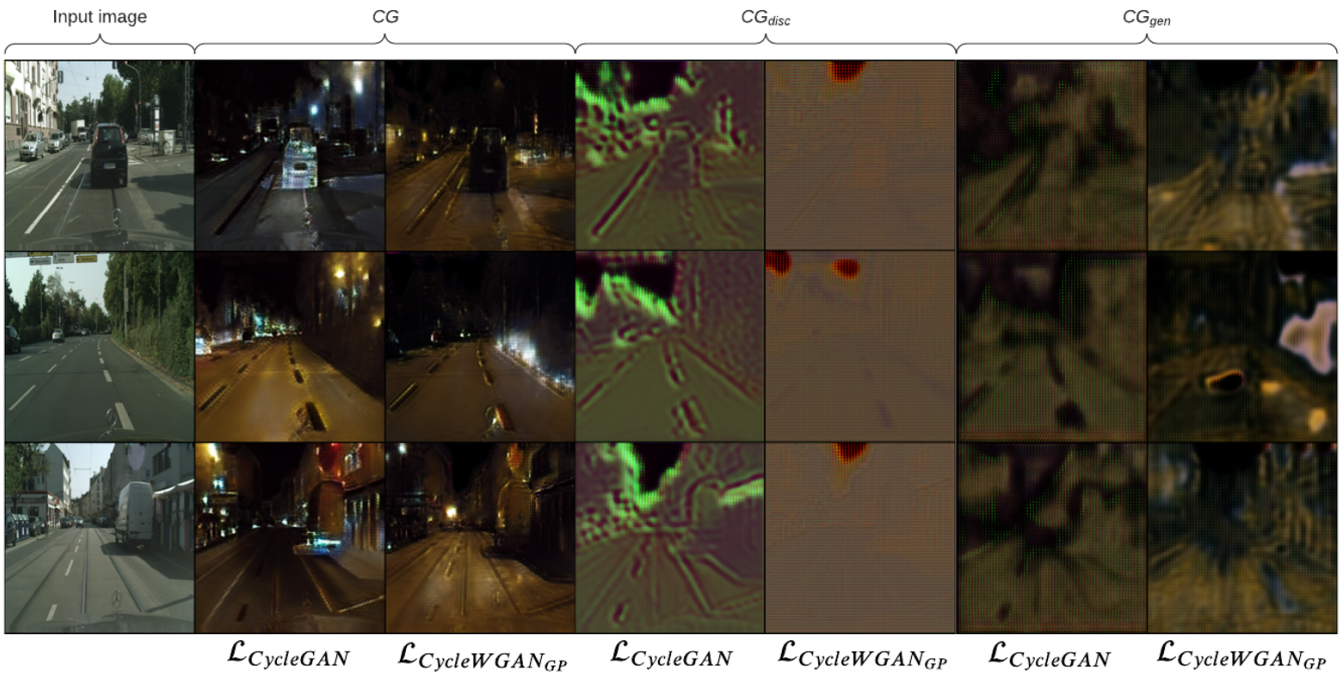


Figure 2: Images from the X_D test-set converted to images from X_N with $\mathcal{L}_{CycleGAN}$ and $\mathcal{L}_{CycleWGAN_{GP}}$ as loss functions after 10 epochs of training. Of the columns under *CG*, CG_{disc} and CG_{gen}, the left columns are images converted using $\mathcal{L}_{CycleGAN}$ as the loss function during training and the right columns are images converted using $\mathcal{L}_{CycleWGAN_{GP}}$. The pore quality of the images converted by CG_{disc} and CG_{gen} compared to images converted by *CG* indicate that CG_{disc} and CG_{gen} are training in an unstable fashion.

4.3.1 Architecture without CIConv

We investigate whether *CG* with $\mathcal{L}_{CycleGAN}$ as its loss function is capable of performing day-night domain adaptation. This experiment will be used as a baseline for following experiments with the same loss function.

The images in figure 2 corresponding to *CG* and $\mathcal{L}_{CycleGAN}$ show some results of the architecture trained for 10 epochs. In this figure, we can see that the day-night domain adaptation is being performed quite well considering the relatively few training iterations.

Furthermore, in figures 3a and 3b we can see that the training follows an expected pattern of Generator and Discriminator losses. The G_{N2D} & G_{D2N} *combined loss* in figure 3a follows a steadily decreasing and then relatively constant curve.

Additionally, the Generator and Discriminator adversarial losses in figure 3b quickly converge to somewhat constant values of approximately 0.4. This is slightly lower than the ideal 0.5, however it still indicates stable training. Moreover, these losses of approximately 0.4 are nicely around the middle of the maximum loss of 1.0 and the minimum loss of 0.0 which is again a good sign that stable training is occurring.

4.3.2 Architectures with CIConv

We investigate whether CG_{disc} and CG_{gen} both with $\mathcal{L}_{CycleGAN}$ as their loss function, are capable of performing day-night domain adaptation. The first priority is to ensure that training is being executed in a stable manner.

The initial indication whether stable training is occurring is the quality of the output images of the architecture. In this case, these output images are the images in figure 2 corresponding to CG_{disc} and CG_{gen} which were trained with $\mathcal{L}_{CycleGAN}$ as their loss function. These images do not indicate any form of convergence towards the sought-after domain adaptation.

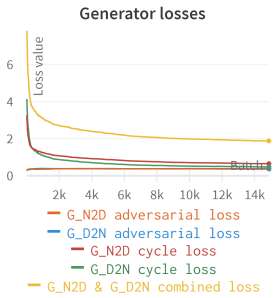
Furthermore, by comparing the stable training in figures 3a and 3b with figures 3c and 3d and figures 3e and 3f, we again find that stable training is not occurring.

Firstly, the G_{N2D} & G_{D2N} *combined loss* in figure 3c and 3e are significantly larger than in figure 3a. Additionally, they converge at a much lower rate.

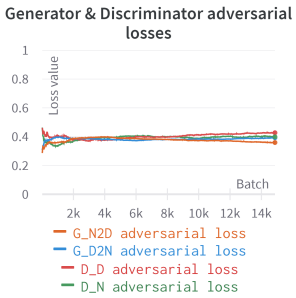
Secondly, the G_{N2D} *adversarial loss* and the G_{D2N} *adversarial loss* in figures 3d and 3f approach a value between 0.9 and 1.0, which represents the maximum loss value for the Generators due to the five percent probabilistic label flipping mentioned in section 4.1. This indicates an extremely high loss for these Generators meaning the Discriminators are effortlessly able to classify the generated images as real and fake.

Thirdly, the G_{N2D} *cycle loss* and the G_{D2N} *cycle loss* in figures 3c and 3e are significantly higher and more chaotic than in figure 3a. This could be caused by the overall pore performance of the Generators, hereby making the domain conversion from one domain to another and then back notably more challenging.

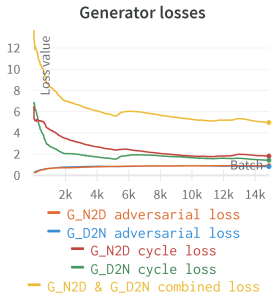
Lastly, the Discriminator adversarial losses in figures 3d and 3f quickly converge to a value of around 0.1, which is practically the minimum due to the one-sided label smoothing



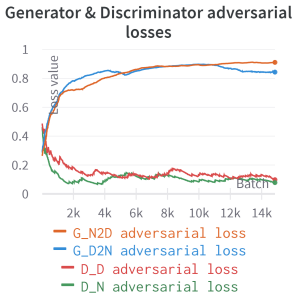
(a) EMA of Generator losses of *CG* with $\mathcal{L}_{CycleGAN}$.



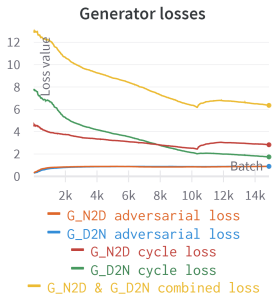
(b) EMA of Generator and Discriminator adversarial losses of *CG* with $\mathcal{L}_{CycleGAN}$.



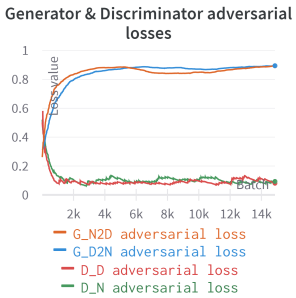
(c) EMA of Generator losses of CG_{disc} with $\mathcal{L}_{CycleGAN}$.



(d) EMA of Generator and Discriminator adversarial losses of CG_{disc} with $\mathcal{L}_{CycleGAN}$.



(e) EMA of Generator losses of CG_{gen} with $\mathcal{L}_{CycleGAN}$.



(f) EMA of Discriminator losses of CG_{gen} with $\mathcal{L}_{CycleGAN}$.

Figure 3: Exponential moving average of Generator and Discriminator losses of *CG*, CG_{disc} and CG_{gen} with $\mathcal{L}_{CycleGAN}$ as the loss function. For *CG*, the initially decreasing and then converging path of the G_{N2D} & G_{D2N} *combined loss* combined with the approximately constant value for the adversarial losses indicate that stable training is occurring. For CG_{disc} and CG_{gen}, convergence of both the Generators and the Discriminators adversarial losses towards practically their maximal and minimal, respectively, combined with a relatively high G_{N2D} & G_{D2N} *combined loss* is a strong indication for unstable training.

mentioned in section 4.1. Hereby, again indicating that the Discriminators have too simple of a job in classifying images as real and fake.

The observations made, especially of the high G_{N2D} and G_{D2N} adversarial losses and the low D_D and D_N adversarial losses strongly indicate unstable training. Section 4.5 will further analyse why CG_{disc} is failing to train in a stable manner.

4.4 Training architectures with $\mathcal{L}_{CycleWGAN_{GP}}$

In this section, we discuss our findings when training *CG*, CG_{disc} and CG_{gen} with $\mathcal{L}_{CycleWGAN_{GP}}$ as the loss function. The architectures are compared based on their output images and the evolution of their losses during training.

4.4.1 Architecture without CIConv

We investigate if *CG* and $\mathcal{L}_{CycleWGAN_{GP}}$ as its loss function is capable of performing the day-night domain adaptation. This experiment will act as a baseline in following experiments with the same loss function. The images in figure 2 corresponding to *CG* and $\mathcal{L}_{CycleWGAN_{GP}}$ show several results after 10 epochs of training. We can see a quite well performing domain adaptation considering the short training time.

Furthermore, the images in figure 2 corresponding to *CG* and $\mathcal{L}_{CycleWGAN_{GP}}$ as the loss function seem to be of higher quality than the same architecture with $\mathcal{L}_{CycleGAN}$ as its loss function. This could suggest that $\mathcal{L}_{CycleWGAN_{GP}}$ is the better of the two $\mathcal{L}_{CycleGAN}$, at least for the current setup.

Figure 4 shows the losses for both the Generators and the Discriminators. Here, we see relatively quick curve convergence for all parameters, most importantly for the G_{N2D} & G_{D2N} *combined loss* in figure 4a. A relatively constant value for this specific loss indicates the desired stable training.

4.4.2 Architectures with CIConv

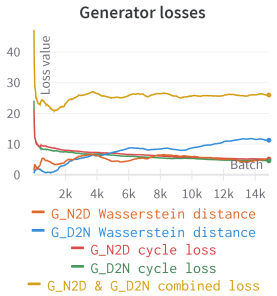
In this section we investigate whether CG_{disc} and CG_{gen}, both with $\mathcal{L}_{CycleWGAN_{GP}}$ as their loss functions, are capable of performing domain adaptation. As in section 4.3.1 our first priority is to have the architectures train in a stable manner.

Once again, our primary indication that stable training is not occurring, is the state of the images in figure 2 corresponding to the architectures trained with $\mathcal{L}_{CycleWGAN_{GP}}$ as their loss function. These images do not suggest a convergence in the direction of the desired domain adaptation. Furthermore, by comparing figures 4a and 4b with figures 4c and 4d and figures 4e and 4f in the following sections, we show stable training does not occur for both architectures with CIConv and $\mathcal{L}_{CycleWGAN_{GP}}$ as their loss function.

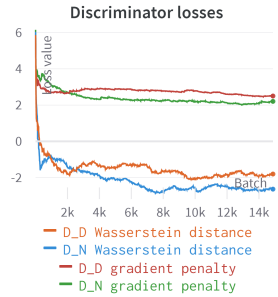
4.4.2.1 CIConv in the Discriminators (CG_{disc})

In figure 4c, we see a relatively low G_{N2D} & G_{D2N} *combined loss* compared to figure 4a. Since the Generators loss is determined via the Discriminators, this suggests underperforming Discriminators.

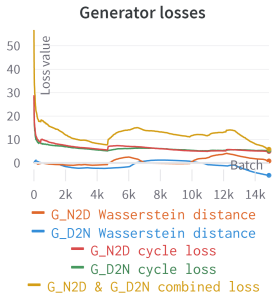
Furthermore, in figure 4d, we see extremely large Gradient Penalty values for both the day and night Discriminators. Notice how the Wasserstein distances are overshadowed by the Gradient Penalty values. To determine the Gradient Penalty, we apply equation 3. This equation uses the Discriminators



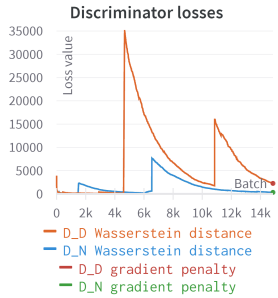
(a) EMA of Generator losses of *CG* with $\mathcal{L}_{CycleWGAN_{GP}}$.



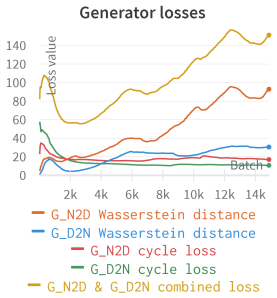
(b) EMA of Discriminator losses of *CG* with $\mathcal{L}_{CycleWGAN_{GP}}$.



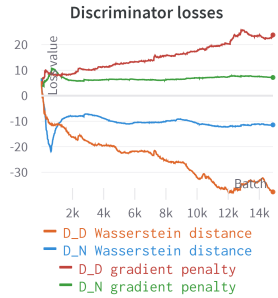
(c) EMA of Generator losses of CG_{disc} with $\mathcal{L}_{CycleWGAN_{GP}}$.



(d) EMA of Discriminator losses of CG_{disc} with $\mathcal{L}_{CycleWGAN_{GP}}$.



(e) EMA of Generator losses of CG_{gen} with $\mathcal{L}_{CycleWGAN_{GP}}$.



(f) EMA of Discriminator losses of CG_{gen} with $\mathcal{L}_{CycleWGAN_{GP}}$.

Figure 4: Exponential moving average of Generator and Discriminator losses of *CG* with $\mathcal{L}_{CycleWGAN_{GP}}$. For *CG*, the quick convergence and relatively constant values for all parameters, especially G_{N2D} & G_{D2N} *combined loss*, indicate stable training. For CG_{disc}, the extremely large Wasserstein distance values are a powerful indication that unstable training is occurring. For CG_{gen}, the relatively large and increasing G_{N2D} & G_{D2N} *combined loss* combined with a far from constant D_D *Wasserstein distance* is a solid sign that unstable training is occurring.

which in this architecture have a CIConv layer as their first layer. This CIConv layer seems to be the cause for the large Gradient Penalty values and thus the unstable training. Further analysis on why the CIConv layer causes unstable training can be found in section 4.5.

4.4.2.2 CIConv in the Generators (CG_{gen})

In figure 4e, we see a strong increase for all parameters compared to figure 4a. As a result the G_{N2D} & G_{D2N} *combined loss* for this architecture is significantly higher than for the same architecture without CIConv. The increased G_{N2D} & G_{D2N} *combined loss* suggest failing Generators.

Additionally, the G_{N2D} *Wasserstein distance* in figure 4e is significantly larger than the G_{N2D} *Wasserstein distance*. These values are determined by using the Discriminators, meaning they have more of a challenge correctly classifying fake images from the night domain than from the day domain. This could be due to the difference in illumination properties between the day and night domain, for example the larger contrast of the day domain.

Furthermore, in figure 4f, we see a decreased value for the D_D *Wasserstein distance* and the D_N *Wasserstein distance* compared to figure 4b. This corresponds to the Discriminators having an undemanding task correctly classifying images as real and fake. Additionally, we again see a smaller value for the D_D *Wasserstein distance* than the D_N *Wasserstein distance*. This is in line with the impression that the Discriminators face less of challenge when classifying fake day images compared to fake night images.

4.5 Analysing Unstable Architectures

In this section, we show that the addition of the CIConv layer in either the Generators or the Discriminators causes exploding gradients in this layer. Additionally, we investigate which term in the CIConv layer causes the exploding gradients problem and finally we show how to mitigate it.

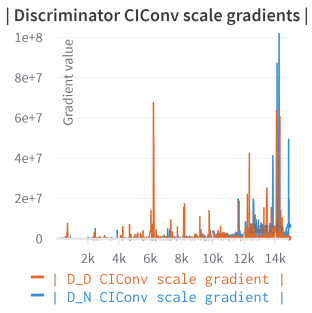
4.5.1 Exploding Gradients

In figures 5a through 5d, we see graphs of the absolute values of the means of the gradients of the CIConv layer scale value. The extremely large values in these graphs indicate exploding gradients.

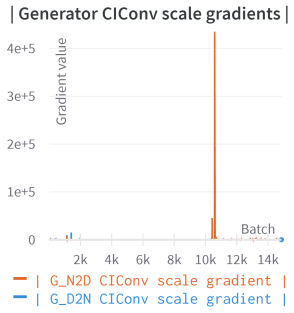
To understand why the gradients in figures 5a through 5d are so large, we partly derive the equation for the gradient of the CIConv layer:

$$\frac{\partial y}{\partial x} = \frac{\frac{1}{W(x)^2+\epsilon}\frac{\partial W(x)^2}{\partial x} - \frac{\partial \sigma_S}{\partial x} log(W(x)^2+\epsilon)}{\sigma_S^2} - \frac{\frac{\partial \mu_S}{\partial x}\sigma_S - \frac{\partial \sigma_S}{\partial x}\mu_S}{\sigma_S^2}, \tag{10}$$

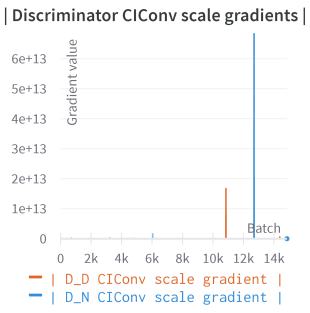
with y being the CIConv layer as in equation 9 with CI as W, see equation 7. We partially derive with respect to x, which represents the input image going into the CIConv layer. Equation 10 is used in following sections to investigate why the CIConv layer has exploding gradients.



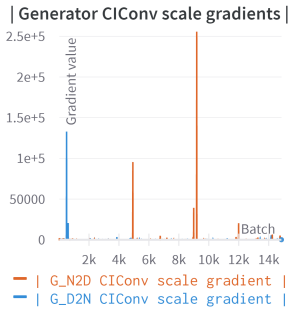
(a) CG_{disc} with $\mathcal{L}_{CycleGAN}$ CIConv scale gradient with log term.



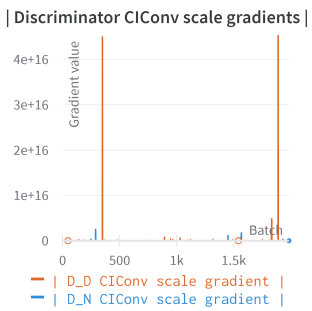
(b) CG_{gen} with $\mathcal{L}_{CycleGAN}$ CIConv scale gradient with log term.



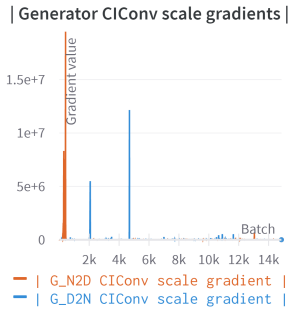
(c) CG_{disc} with $\mathcal{L}_{CycleWGAN_{GP}}$ CIConv scale gradient with log term.



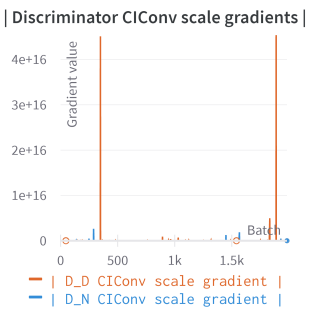
(d) CG_{gen} with $\mathcal{L}_{CycleWGAN_{GP}}$ CIConv scale gradient with log term.



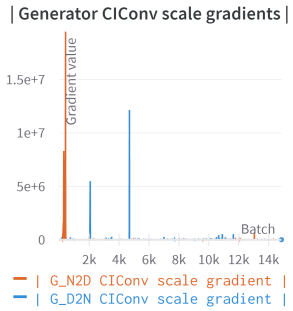
(e) CG_{disc} with $\mathcal{L}_{CycleGAN}$ CIConv scale gradient without log term.



(f) CG_{gen} with $\mathcal{L}_{CycleGAN}$ CIConv scale gradient without log term.



(g) CG_{disc} with $\mathcal{L}_{CycleWGAN_{GP}}$ CIConv scale gradient without log term.



(h) CG_{gen} with $\mathcal{L}_{CycleWGAN_{GP}}$ CIConv scale gradient without log term.

Figure 5: These graphs provide a visualisation of the size of the mean of the CIConv layer scale gradients for the different architectures and the two loss functions, see graph captions for details. The top four graphs show the CIConv layer scale gradients for the CIConv layer with the log term, the bottom four graphs without the log term. The training session at figure 5g only ran for a little more than one epoch before the gradients became too large and caused an error. The difference in magnitude between the top four graphs and the bottom four graphs shows that the log term in equation 9 is not the cause of the exploding gradients.

4.5.2 Remove log term

In equation 10, we see the term $\frac{1}{W(x)^2+\epsilon}$. When $W(x)$ approaches zero, this term becomes large which is a potential cause for the exploding gradients. To further test this theory, we removed the log term in equation 9, resulting in:

$$z = CIConv(x, y) = \frac{CI^2(x, y, \sigma = 2^s) + \epsilon - \mu_S}{\sigma_S}, \quad (11)$$

with the following partial derivative with respect to x as the gradient:

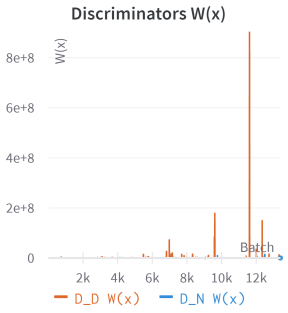
$$\frac{\partial z}{\partial x} = \frac{\frac{\partial W(x)^2}{\partial x}\sigma_S - \frac{\partial \sigma_S}{\partial x}W(x)^2}{\sigma_S^2} - \frac{\frac{\partial \sigma_S}{\partial x}\epsilon}{\sigma_S^2} - \frac{\frac{\partial \mu_S}{\partial x}\sigma_S - \frac{\partial \sigma_S}{\partial x}\mu_S}{\sigma_S^2}. \quad (12)$$

This equation does not contain a $\frac{1}{W(x)^2+\epsilon}$ term. We expect the gradients to be smaller when $W(x)$ approached zero. If this is the case, the log term will be identified as a probable cause for the exploding gradients.

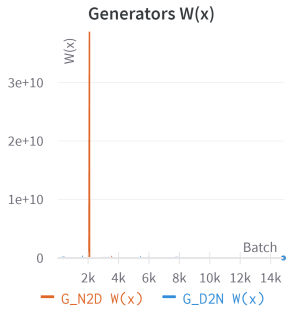
However, we trained the CG_{disc} and CG_{gen} with both $\mathcal{L}_{CycleGAN}$ ad $\mathcal{L}_{CycleWGAN_{GP}}$ and observed a significant increase in the gradient size in all four cases, see figures 5e through 5h. This increase ranged from a factor of 1e2 to 1e9. This shows that the log term in the CIConv layer and thus the $\frac{1}{W(x)^2+\epsilon}$ term in equation 10 is not the cause of the exploding gradients. The opposite seems to be true, where the $\frac{1}{W(x)^2+\epsilon}$ term reduces the size of the gradient. This could suggest that the issue does not lay in $W(x)$ approaching zero but in it becoming considerably large. This theory is explored in the following section.

4.5.3 Clamp W

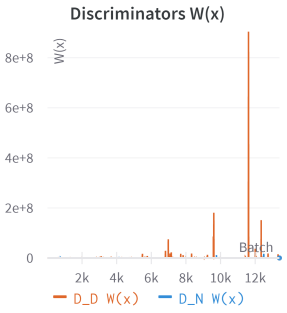
In figures 6a through 6d, we see that the values for $W(x)$ are indeed considerably large. To test whether these large values for $W(x)$ are the cause for the exploding gradients in the CIConv layer, we cap these values before applying the log term. This results in the following equation for CIConv:



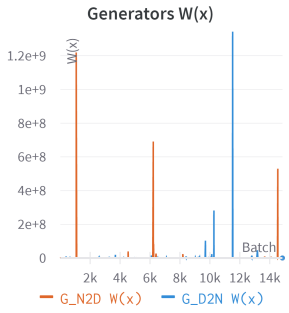
(a) Uncapped $W(x)$ for CG_{disc} with $\mathcal{L}_{CycleGAN}$.



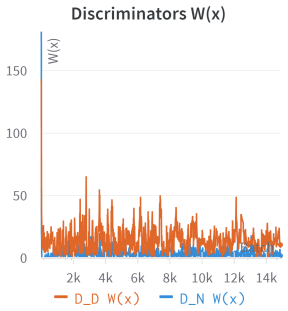
(b) Uncapped $W(x)$ for CG_{gen} with $\mathcal{L}_{CycleGAN}$.



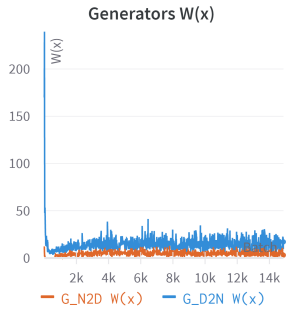
(c) Uncapped $W(x)$ for CG_{disc} with $\mathcal{L}_{CycleWGAN_{GP}}$.



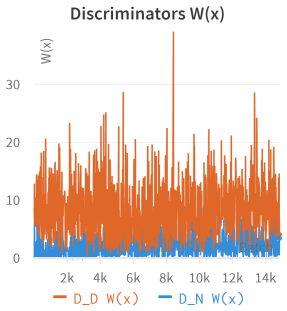
(d) Uncapped $W(x)$ for CG_{gen} with $\mathcal{L}_{CycleWGAN_{GP}}$.



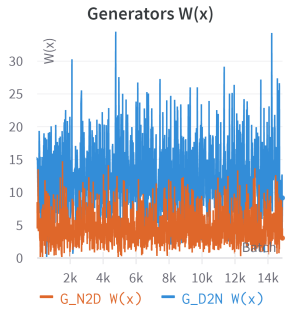
(e) Capped $W(x)$ for CG_{disc} with $\mathcal{L}_{CycleGAN}$.



(f) Capped $W(x)$ for CG_{gen} with $\mathcal{L}_{CycleGAN}$.



(g) Capped $W(x)$ for CG_{disc} with $\mathcal{L}_{CycleWGAN_{GP}}$.



(h) Capped $W(x)$ for CG_{gen} with $\mathcal{L}_{CycleWGAN_{GP}}$.

Figure 6: These graphs provide a visualisation of the size of W for the different architectures and the two loss functions. Each graph shows the value of W for a certain architecture with a certain loss function, see graph captions for details. The top four graphs show the values of an uncapped W, the bottom four graphs of a capped W at 1e4. The training session at figure 6c only ran for 9 epochs before the values became too large and caused an error. The extreme decrease in magnitude of the values of the bottom four graphs compared to the top four graphs shows that an uncapped W has few considerably large values but overall is relatively low. Capping W therefore seems a valid candidate to solve the exploding gradients issue.

$$CIConv(x, y) = \frac{log(C_\alpha(CI^2(x, y, \sigma = 2^s) + \epsilon)) - \mu_S}{\sigma_S}, \tag{13}$$

where C_α is our cap function with α the cap value and where CI is the W variant from equation 7.

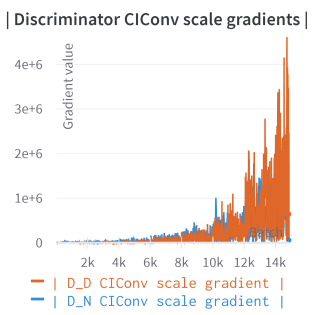
By running the training session again, now with the updated equation for CIConv with $\alpha = 1e4$, we find that the values for $W(x)$ have significantly decreased. This can be seen in figures 6e through 6h. Notice how these values are notably lower than α suggesting that $W(x)$ without capping has a few extremely large values but is overall relatively low. This also follows what we see in figures 6a through 6d.

The corresponding CIConv layer scale gradients can be seen in figure 7. Here, we see that the gradients are significantly lower than in figures 5a through 5d, where $W(x)$ was not capped. Especially CG_{gen} has reduced immensely. For CG_{disc} the gradients have not reduced as much compared to CG_{gen}.

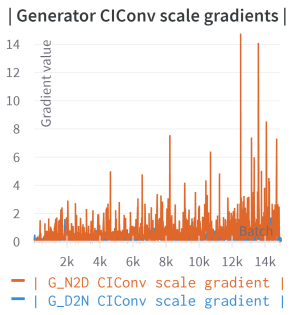
The resulting images of capping $W(x)$ can be seen in figure 8. By comparing these images to the images from figure 2, it becomes apparent that capping $W(x)$ greatly improved the performance of CG_{disc} and especially CG_{gen}. In figure 2, any architecture with CIConv produced images that did not come close to the images generated by CG. In figure 8, images from CG_{disc} structurally resemble the images from CG, the color is still off yet this could be corrected with further testing. Images from CG_{gen} greatly resemble the images from CG.

5 Discussion

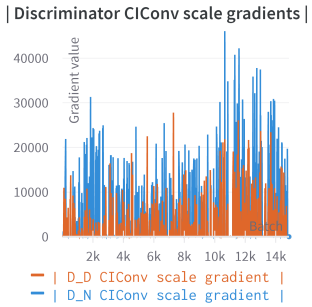
To establish whether CIConv is a useful addition to CycleGANs when it comes to performing day-night domain adaptation, we created three architectures. The first acted as a baseline and did not use CIConv, the second applied CIConv in the first layer of the Discriminators and the third had CIConv as the first layer in the Generators. In an effort to have the architectures train in a stable manner, we made numerous architecture and training procedure adjustments, attempted a number of hyperparameter combinations and most noticeably



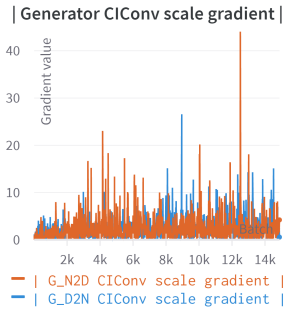
(a) CG_{disc} with $\mathcal{L}_{CycleGAN}$ CIConv scale gradient.



(b) CG_{gen} with $\mathcal{L}_{CycleGAN}$ CIConv scale gradient.



(c) CG_{disc} with $\mathcal{L}_{CycleWGAN_{GP}}$ CIConv scale gradient.



(d) CG_{gen} with $\mathcal{L}_{CycleWGAN_{GP}}$ CIConv scale gradient.

Figure 7: These graphs provide a visualisation of the size of the CIConv layer scale gradient for the different architectures and the two loss functions with W capped at 1e4. See graph captions for the used architecture and the loss function. These graphs no longer show exploding gradients and therefore the capping of W seems a valid method for preventing this issue.

implemented two different loss functions. The first loss function applied an Adversarial Loss and the second a Wasserstein Distance with Gradient Penalty. Additionally, they both applied a Cycle-Consistency Loss. We found that the usage of an unadjusted CIConv layer resulted in unstable training due to it causing Exploding Gradients. It is therefore unsuitable as an addition to the CycleGAN architecture for day-night domain adaptation.

Analysis of the behaviour of the CIConv layer and of the gradient through it led to the discovery that the output of the CIConv layer followed a pattern similar to that of Exploding Gradients. We found this to be the cause for the Exploding Gradients and propose a method for mitigating the issue. This method consists of the capping of W before applying the log term and before normalizing. The usage of this method provided some interesting preliminary results of the day-night domain adaptation with the use of CIConv. These results suggest that CIConv in the first layer of the Generators has the potential to be of use during domain adaptation. Additionally, it seems that CIConv in the first layer of the Discriminators is not a useful addition.

However, the method of capping W has not been extensively tested since only one capping value has been applied so far. Additionally, variations to the method might prove more effective than the simple version it is now. Furthermore, we suspect that the optimal combination of hyperparameters, training procedure and loss function is yet to be discovered. So far, only one combination has been used to show the potential of this method.

Figure 8: Images from the X_D test-set converted to images from X_N with $\mathcal{L}_{CycleGAN}$ and $\mathcal{L}_{CycleWGAN_{GP}}$ as loss functions with W capped at 1e4 after 10 epochs of training. Of the columns under CG_{disc} and CG_{gen}, the left columns are images converted using $\mathcal{L}_{CycleGAN}$ as the loss function during training and the right columns using $\mathcal{L}_{CycleWGAN_{GP}}$. The quality of these images has greatly improved compared to the images where W was not capped (figure 2). Especially CG_{gen} seems to be performing the day-night domain adaptation quite well. CG_{disc} is structurally performing well, however the colors are off.

Finally, we conclude that an altered version of CIConv in combination with CycleGANs has the potential to be of benefit to the day-night domain adaptation and therefore the generation of labeled training data in a domain for which only unlabeled training data is available. However, further research is required to decisively say whether applying the adjusted CIConv layer increases performance compared to not applying any CIConv layer.

6 Responsible Research

When performing research in a field such as neural networks, certain ethical implications need to be discussed. More so if the resulting method has the potential to be used to train CNNs in self-driving cars. This section will discuss the most important ethical implications.

In this paper, we discuss a potential method for converting images from the day domain to the night domain such that they can be used to train the CNNs present in self-driving cars. Because of the potential use in self-driving cars, human lives may be at risk. As a consequence, it is extremely important that this work is not blindly copied and applied in ones

own application. Further tweaking and testing is most likely needed.

The implementation has been made available at [22]. Here, the code can be cloned and used to reproduce old experiments or carry out new experiments.

References

[1] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks,"

[2] A. Lengyel, S. Garg, M. Milford, and J. C. van Gemert, "Zero-shot day-night domain adaptation with a physics prior,"

[3] W. Hu, M. Li, and X. Ju, "Improved CycleGAN for image-to-image translation,"

[4] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation,"

[5] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2020.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks,"

[7] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[8] G. Antipov, M. Baccouche, and J.-L. Dugelay, "Face aging with conditional generative adversarial networks," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 2089–2093. ISSN: 2381-8549.

[9] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN,"

[10] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc.

[11] H. Thanh-Tung, T. Tran, and S. Venkatesh, "Improving generalization and stability of generative adversarial networks,"

[12] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks,"

[13] W. Hong, Z. Wang, M. Yang, and J. Yuan, "Conditional generative adversarial network for structured domain adaptation," pp. 1335–1344.

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976, IEEE.

[15] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks,"

[16] E. Romera, L. Bergasa, K. Yang, J. M. Alvarez, and R. Barea, "Bridging the day and night domain gap for semantic segmentation," pp. 1312–1318.

[17] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "CyCADA: Cycle-consistent adversarial domain adaptation," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 1989–1998, PMLR. ISSN: 2640-3498.

[18] J. M. Geusebroek, R. v. d. Boomgaard, A. W. M. Smeulders, and H. Geerts, "Color invariance," vol. 23.

[19] N. Alshammari, S. Akcay, and T. P. Breckon, "On the impact of illumination-invariant image pre-transformation for contemporary automotive semantic scene understanding," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1027–1032, 2018.

[20] B. A. Maxwell, C. A. Smith, M. Qraitem, R. Messing, S. Whitt, N. Thien, and R. M. Friedhoff, "Real-time physics-based removal of shadows and shading from road surfaces," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1277–1285, 2019.

[21] J. Brownlee, "How to implement GAN hacks in keras to train stable models."

[22] T. Streefkerk, "day2night." https://gitlab.tudelft.nl/attilalengyel/brp-ciconv/-/tree/master/thomas_streefkerk, 2022.

A Appendix

D \ G	1e-6	2e-6	4e-6	1e-5	5e-5	1e-4	2e-4
1e-6	X			X			
2e-6		X		X			
2.5e-6				X			
4e-6			X	X			
7.5e-6				X			
1e-5		X	X	X	X		
5e-5						X	
1e-4						X	
2e-4							X

Table 5: Attempted learning rate combinations with $\mathcal{L}_{CycleGAN}$ as the loss function. The top row represents the learning rates for the Generators, the first column represents the learning rates for the Discriminators. An *X* on an intersection indicates that the learning rate was attempted.