Vinicius Ferraço Arruda

Cross-Domain Object Detection Using Unsupervised Image Translation and Neural Style Transfer

Vitória, ES

April 22, 2022

Vinicius Ferraço Arruda

Cross-Domain Object Detection Using Unsupervised Image Translation and Neural Style Transfer

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Supervisor: Prof. Dr. Thiago Oliveira dos Santos

Vitória, ES

April 22, 2022

Vinicius Ferraço Arruda
Cross-Domain Object Detection
Using Unsupervised Image Translation and Neural Style Transfer/ Vinicius Ferraço Arruda. – Vitória, ES, April 22, 2022-
72 p. : il. (algumas color.) ; 30 cm.

Supervisor: Prof. Dr. Thiago Oliveira dos Santos

Dissertação de Mestrado – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Programa de Pós-Graduação em Informática, April 22, 2022.

1. Unsupervised Domain Adaptation. 2. Object Detection. 3. Generative Adversarial Networks. 4. Unpaired Image-to-Image Translation. 5. Neural Style Transfer. I. Arruda, Vinicius Ferraço. II. Oliveira-Santos, Thiago. II. Universidade Federal do Espírito Santo. IV. Cross-Domain Object Detection
Using Unsupervised Image Translation and Neural Style Transfer

CDU 02:141:005.7

Vinicius Ferraço Arruda

Cross-Domain Object Detection Using Unsupervised Image Translation and Neural Style Transfer

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Trabalho aprovado. Vitória, ES, 24 de fevereiro de 2022:

Prof. Dr. Thiago Oliveira dos Santos
Orientador

Prof. Dr. Thomas Walter Rauber
Convidado 1

Prof. Dr. Filipe Mutz
Convidado 2

Vitória, ES
April 22, 2022

Acknowledgements

This study was financed in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES, Brazil) - Finance Code 001; Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Brazil) and Fundação de Amparo à Pesquisa do Espírito Santo (FAPES, Brazil) - grant 84412844. We thank the NVIDIA Corporation for the donation of a Titan Xp GPU used in this research.

"What I cannot create, I do not understand."
(Richard Feynman)

Resumo

A adaptação de domínio de modo não supervisionado para detecção de objetos aborda a adaptação de detectores treinados em um domínio de origem para atuar com precisão em um domínio alvo desconhecido. Em aplicações do mundo real, é desejável que os detectores de objetos trabalhem com precisão, independentemente do domínio de aplicação (por exemplo, condições climáticas). Esses modelos têm a propriedade intrínseca de serem enviesados aos dados de treinamento e são conhecidos por não generalizar bem para dados desconhecidos. A maior disponibilidade de conjuntos de dados pode ser vista nos domínios mais prevalentes (por exemplo, dia ensolarado), mas para certas aplicações pode ser necessário treinar um modelo para atuar em um domínio menos predominante (por exemplo, dia com neblina). Para tal, se faz necessário realizar a coleta de um conjunto de dados de domínios menos predominantes. No entanto, a aquisição de um novo conjunto de dados envolve o laborioso processo de anotação de dados, mas a coleta de grandes quantidades de dados sem anotação pode ser viável. Recentemente, métodos de adaptação de domínio não supervisionados que abordam o alinhamento das características intermediárias mostraram-se promissores, alcançando resultados no estado da arte. No entanto, esses métodos são trabalhosos de implementar e por serem métodos caixa preta são difíceis de interpretar. Embora tenham resultados promissores, ainda há espaço para melhorias para fechar a lacuna de desempenho em relação ao modelo ideal (ao treinar com os dados do domínio alvo). Neste trabalho, propomos um método para gerar um conjunto de dados artificial no domínio alvo para treinar um detector de objetos. Empregamos um tradutor de imagem não supervisionado (CycleGAN) e um método de transferência de estilo neural (baseado em AdaIN) usando apenas dados anotados do domínio de origem e dados não anotados do domínio alvo. Nossas principais contribuições são a proposta de um método menos complexo de adaptação de domínio de modo não supervisionado para detecção de objetos, porém mais eficaz, que também tem uma interpretabilidade aprimorada em relação aos métodos estado da arte. Os resultados em cenários do mundo real para direção autônoma mostram melhorias significativas, superando os métodos estado da arte na maioria dos casos, obtendo resultados ainda mais próximos do modelo ideal.
Palavras-chave: Adaptação de Domínio Não Supervisionado. Detecção de Objetos. Redes Generativas Adversariais. Tradução Não Pareada de Imagem para Imagem. Trasnferência de Estilo Neural.

Abstract

Unsupervised domain adaptation for object detection addresses the adaptation of detectors trained in a source domain to work accurately in an unseen target domain. In real-world applications, object detectors are desired to work accurately regardless of the application domain (e.g., weather condition). These models have the intrinsic property of being biased towards the training data and are known to not generalize well to unseen data. The greatest availability of datasets can be seen in the most prevalent domains (e.g., sunny day), but for certain applications it may be necessary to train a model to deploy in a less prevalent one (e.g., foggy day). In addition, the acquisition of a new dataset involves the laborious process of data annotation, but collecting large amounts of data without annotation might be feasible. Recently, methods for unsupervised domain adaptation approaching the alignment of the intermediate features proven to be promising, achieving state-of-the-art results. However, these methods are laborious to implement and hard to interpret. Although promising, there is still room for improvements to close the performance gap toward the upper-bound (when training with the target data). In this work, we propose a method to generate an artificial dataset in the target domain to train an object detector. We employed an unsupervised image translator (CycleGAN) and a neural style transfer method (AdaIN-based) using only annotated data from the source domain and non-annotated data from the target domain. Our key contributions are the proposal of a less complex yet more effective method that also has an improved interpretability. Results on real-world scenarios for autonomous driving show significant improvements, outperforming state-of-the-art methods in most cases, further closing the gap toward the upper-bound.

Keywords: Unsupervised Domain Adaptation. Object Detection. Generative Adversarial Networks. Unpaired Image-to-Image Translation. Neural Style Transfer.

List of Figures

List of Tables

List of abbreviations and acronyms

MLP	Multilayer Perceptron
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
UDA	Unsupervised Domain Adaptation
FPS	Frames per second
AP	Average Precision
mAP	Mean Average Precision
FC	Fully Connected Layer
RGB	Red Green Blue
SVM	Support Vector Machine
RPN	Region Proposal Layer
NST	Neural Style Transfer
AdaIN	Adaptive Instance Normalization
i.i.d.	Independent and Identically Distributed
p.p.	Percentage Points

Contents

1 Introduction

1.1 Problem description

Object detection is a class of technology for automating object instance search and location retrieval. This class of technology is in constant development, being an interdisciplinary research area, including but not limited to neuroscience, psychology, physics and computer science. The first technologies were developed based on image processing and classical machine learning techniques. However, with the advent of neural networks and deep learning and its promising results, the research focus has changed. Recent works (REN et al., 2015; REDMON et al., 2016; LIN et al., 2017; TAN; PANG; LE, 2020) have shown that object detectors based on deep learning have achieved state-of-the-art performance. This outstanding performance nonetheless comes at the cost of large annotated datasets of the target application domain. Still, one might be interested in deploying such models in a different target domain than the one used in training (source domain). For instance, large annotated datasets may be available for sunny-day driving images, but it is usually desirable to also deploy the model for foggy-day. These models are well known to not generalize well to unseen data, therefore a challenging problem arises. Although annotated data in the target domain may be unfeasible or too expensive to acquire, collecting large amounts of data without annotation of the target domain (e.g., capturing images by driving around at foggy-day) might be easy or cheap. In such a scenario, generalizing to the new domain is known as Unsupervised Domain Adaptation (UDA), where source and target domains are semantically similar (containing the same objects in a similar context), but distinct in appearance. Additionally, annotated data are only available for the source domain while non-annotated data are available for the target domain.

Training object detectors to accurately work across domains is highly desirable for several applications. In particular, in the context of self-driving vehicles, objects such as crosswalks, traffic lights, pedestrians, riders, and cars, must be correctly detected regardless of the light or weather conditions, scenario, and camera settings. However, the amount of drivable scenarios is virtually unlimited, yielding countless domains in which these detectors can be employed, such as day- and night-time, sunny, rainy or snowy weather, urban, highway or country roads, not to mention the different possible camera settings of the dataset acquisition versus the one employed in the target application. Typically, annotated data are more available in the most prevalent domain, e.g., urban, sunny and day-time images, but it is crucial to train detectors to work properly regardless of the domain. Taking into account the scarcity of annotated data available in these different

target domains, the acquisition of a robust model for the detection of objects across domains is a challenge.

1.2 Motivation

Autonomous vehicles have been a promising technology for a decade. The accelerated pace of recent years was only possible due to the advent of deep learning, which is an important component for building a fully autonomous vehicle. Due to the great diversity of application domains, it is expensive and laborious to acquire a significant amount of data from each domain to train a robust object detector. This fact is due to the manual and arduous process of annotating each image, which involves drawing a tight bounding-box around each object instance of interest. It is important to emphasize the need to annotate each object of interest, otherwise the model may have high false negative rates. Therefore, the problem of adapting a model based on deep learning to an unseen target data becomes a pertinent problem.

This problem of adapting such a model to unseen data is an active research area in the literature. The current state-of-the-art methods approach the problem very similarly to each other, by integrating domain adaptation components to an object detector (e.g., Faster R-CNN (REN et al., 2015)). Chen et al. (CHEN et al., 2018) combined two domain adaptation components in the Faster R-CNN to minimize the domain shift at image- and instance-level, respectively. Claiming that such domain-shift minimization may hurt the performance for large shifts, Saito et al. (SAITO et al., 2019) integrated Faster R-CNN with a weak global alignment, which partially minimizes the domain-shift at image-level, and a strong local alignment.The UDA problem for object detection is far from solved. Currently, the upper-bound (when training with the target data) remains much higher than current state-of-the-art methods (ARRUDA et al., 2019; CHEN et al., 2018; SAITO et al., 2019; SHEN et al., 2019; LI et al., 2020; ZHU et al., 2019), indicating a lot of room for improvement. In this work, we aim at further closing this performance gap.

1.3 Objectives

The objective of this work is to investigate and develop a method to adapt an object detector trained in a source domain to a target application domain. The adaptation of the model must be performed in an unsupervised way, without the need of annotations for the target images. Furthermore, it must be possible to interpret the intermediate steps of the adaptation process.

In this context, we propose a simpler yet efficient two-stage approach: i) training an unsupervised image-to-image translation model to generate an artificial dataset that

resembles the target domain (fake-data), and ii) training an object detector with the newly acquired data.

More specifically, the objectives of this work are the following:

- investigation of the use of an unsupervised image-to-image translation model and a neural style transfer model to artificially generate data;
- investigation of the efficacy of the method on multi-class datasets;
- investigation of the benefits of using two translation models together, comprising an artificial dataset from two distinct generation processes;
- evaluation of the method with different arrangements of the annotated source training dataset and those artificially generated for the second-stage;
- conduction of an extensive comparison of the results with state-of-the-art methods, leading to a more detailed discussion;
- investigation of a diverse set of experiments with several well-known datasets (Cityscapes (CORDTS et al., 2016), KITTI (GEIGER et al., 2013), SIM10k (JOHNSON-ROBERSON et al., 2017) and Foggy Cityscapes (SAKARIDIS; DAI; GOOL, 2018)) that includes learning from synthetic data, driving in adverse weather conditions, and cross-camera adaptation.

1.4 Contributions

The main contribution of our work is showing that using a method to artificially generate images in combination with a standard object detector is sufficient to surpass the accuracy of state-of-the-art methods which, in contrast, are laborious to implement. In addition, our method has as a novelty the possibility of visualizing the artificially generated data in the target domain, enabling the inspection of the quality of the target data that will be used in the training of the detector. This represents a significant advantage in terms of interpretability when compared to the state of the art.

As an outcome of this investigation, the contributions of this work resulted in two publications. One publication in an international conference, IJCNN, and another in a notable journal, ESWA:

- ARRUDA, V. F. et al. Cross-Domain Car Detection Using Unsupervised Image-to-Image Translation: From Day to Night. In: *International Joint Conference on Neural Networks (IJCNN)*, 2019. (ARRUDA et al., 2019)

- ARRUDA, V. F. et al. Cross-Domain Object Detection Using Unsupervised Image Translation. *Expert Systems with Applications*, 2021. (ARRUDA et al., 2021)

1.5 Dissertation Structure

The remainder of the work is organized as follows. The next chapter presents the theoretical background that supports this work. Chapter 3 describes the proposed method. The experimental method and the obtained results are presented in Chapter 4. Finally, the conclusion of the work is presented in Chapter 5.

2 Theoretical Background

In this chapter, fundamental concepts and related works is presented for a better understanding of the rest of the text. First, we introduce the basic concepts of Artificial Neural Networks, from simple architectures to complex ones. Next, the problem of object detection is addressed introducing a widely adopted state-of-the-art object detector. Then, in sequence, the tasks of Unsupervised Image-to-Image Translation and Neural Style Transfer are described, presenting the models employed in this work. Finally, in the last section, the problem of Unsupervised Domain Adaptation for Object Detection is introduced along with a review of related works.

2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) is a computational model inspired by the biological neuron that allows learning by example from real-world data to address a complex problem, usually classification and regression problems.

2.1.1 Biological and Artificial Neuron

The biological neuron is responsible for managing all the activities of the human body, being located in the brain, which counts billions of neurons, where each element is divided into three basic parts: dendrites, cell body and axon (HAYKIN, 1998), as shown in Figure 1. The dendrites, receive information from other neurons and transmit it to the

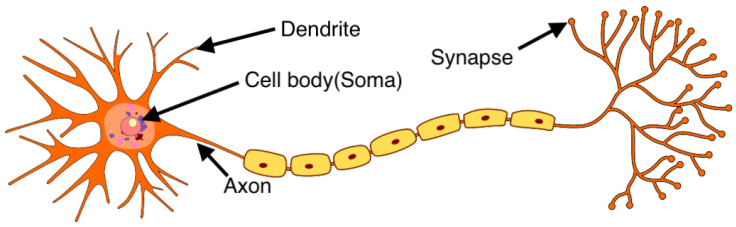

Figure 1 – Illustration of a biological neuron (Source: Wikimedia Commons).

nucleus, which is located within the cell body, then the axons receive information from the cell body and transport it through the synapses.

Analogous to the biological neuron, the perceptron is extremely important in the understanding of artificial neural networks (ROSENBLATT, 1958), since artificial neural networks use the perceptron (Figure 2) as its basic component. Its structure has inputs, which are equivalent to dendrites, an adder that is equivalent to the nucleus and concentrates all the information, and has an output that transmits a signal for a given

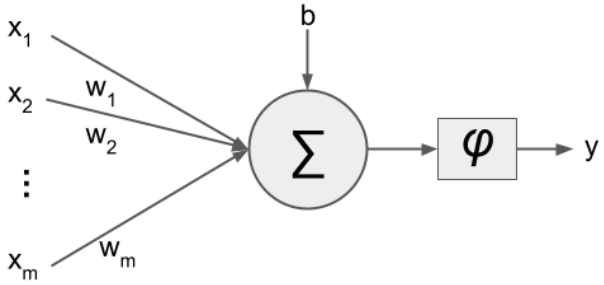

Figure 2 – Illustration of a perceptron (Source: the author).

purpose or for another perceptron. Mathematically describing Figure 2, $X = [x_1, x_2, ..., x_m]$ is the set of information coming to the perceptron, which are weighted by the synaptic weights $W = [w_1, w_2, ..., w_m]$ with a bias b, being all signals unified by the additive function. This whole procedure is called presynaptic activity, and can be summarized in the following equation:

$$I = \sum_{i=1}^{m} w_i x_i + b \tag{2.1}$$

The postsynaptic activity is given by the activation function $y = \varphi(I)$, where φ is usually a non-linear function, such as the listed in Table 1. The procedure of tuning the

Name	Function
Sigmoid	$\frac{1}{1+e^{-x}}$
ReLU	$max(0, x)$
tanh	$\frac{e^z - e^{-z}}{e^z + e^{-z}}$
Leaky ReLU	$max(0.1x, x)$

Table 1 – Examples of non-linear activation functions.

parameters W and b is called *training*. Several techniques for training ANNs exist, but due to its efficacy, algorithms based on the gradient descent technique are usually employed.

2.1.2 Network Architectures

ANNs have as basic element the artificial neuron or perceptron, which can be combined with other perceptrons, giving rise to several types of network aimed at different applications. Thus, some existing structures are presented, which serve as a background for this work, such as single-layer networks, multiple-layer networks, convolutional neural networks and generative adversarial networks.

2.1.2.1 Single-layer Networks

ANNs, in general, are arranged in layers, each containing an arbitrary number of neurons. The single-layer network is the simplest architecture, where the input layer is directly connected with the output layer, as described in Figure 3.

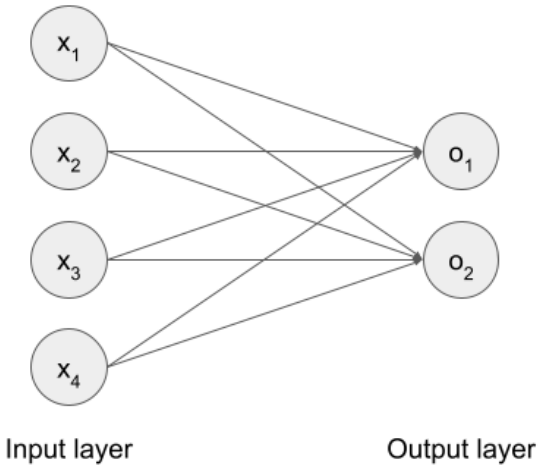

Figure 3 – Illustration of a single layer network with four input neurons and two output neurons. There is only one set of weights, which are connecting the input to the output layer (Source: the author).

2.1.2.2 Multi-layer Networks

Multi-layer Networks are characterized by having an input layer, intermediate layers or hidden layers and an output layer. The hidden layers start to mediate the information between the input and output of the network, making it more capable to extract information of high complexity (HAYKIN, 1998). Figure 4 illustrates an example of this type of architecture with a network with two hidden layers, four input variables and two output variables

2.1.2.3 Convolutional Neural Networks

Convolutional Neural Network (CNN) is a class of ANN bio-inspired in the brain human visual cortex, starting from basic primitives like lines, curves and shapes. This type of network gained the name of Convolutional Neural Networks for their most important layer, the convolutional layer, and have become the state-of-the-art type of network for computer vision problems.

Figure 5 presents a standard model of a CNN, composed in its first part by a sequence of layers that was empirically observed to be responsible to extract successive levels of abstractions (OLAH; MORDVINTSEV; SCHUBERT, 2017), known as a feature maps, building more abstract concepts from the input data. The second part is composed of fully-connected layers (FC or Dense) that are responsible for selecting the attributes that most relate to the desired output. CNNs are usually employed in problems which the input data are images, or data where it is possible to take advantage of their surroundings to extract information. This property of extracting information from its neighborhood comes from convolutional layers, where each convolution filter scans the entire image and specializes in a particular feature.

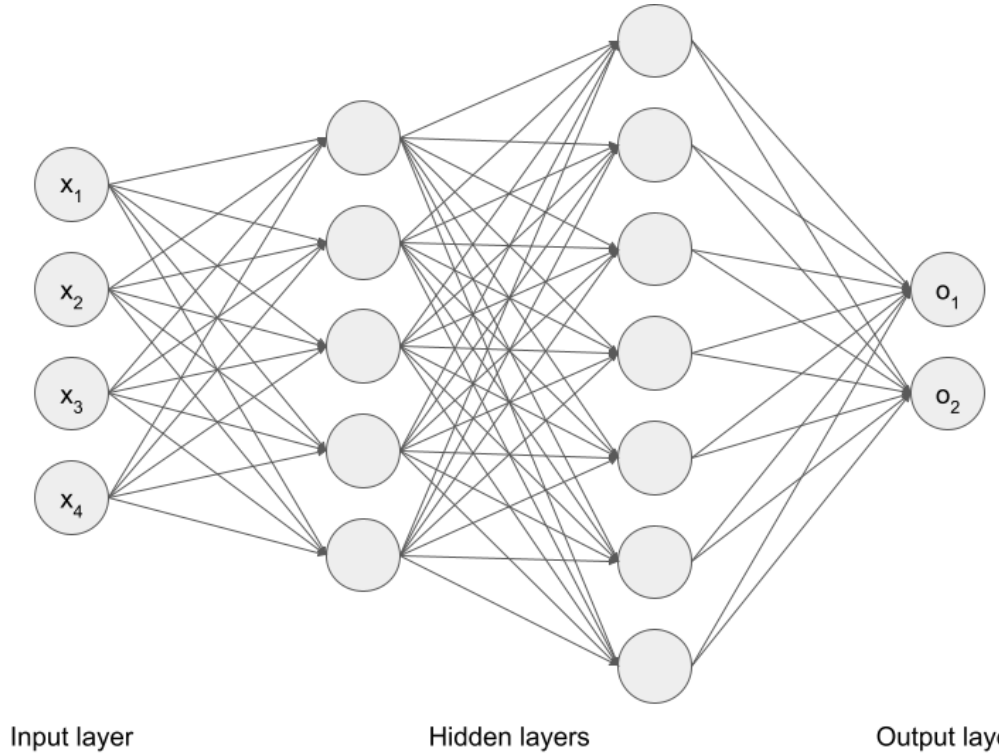

Figure 4 – Illustration of a multi layer network with four input neurons and two output neurons. There are two hidden layers. The first with five neurons and the second with seven neurons. There are three sets of weights, totaling 69 weights (Source: the author).

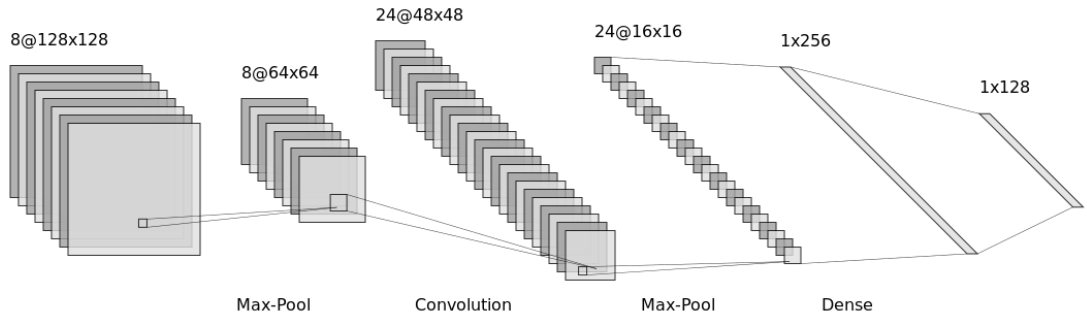

Figure 5 – Illustration of a CNN network. The first four layers are of type CNN, followed by two fully-connected layers. The illustration shows the feature dimensions and the operations applied between them (Source: the author).

The convolution layer is primarily responsible for extracting spatial features from the presented data, applying the convolution operation on the data from convolutional filters. In a convolution layer, although originally defined in only one dimension, for the spatial case, the operation acts on four dimensions: filter width (x'), filter height (y'), filter channels (c) and number of filters (f).

$$F_{\text{out}}(f, x, y) = \sum_{c=1}^{C} \sum_{x'=1}^{X} \sum_{y'=1}^{Y} F_{\text{in}}(c, x - x', y - y') W(c, x', y', f) \tag{2.2}$$

The output feature (F_{out}) of dimensions ($R_x \times R_y$) is produced by the convolution of filters

(W), of dimensions ($x' \times y'$), with the input feature (F_{in}), of dimensions ($X \times Y$).

Figure 6 shows an example with an image of dimension 5×5 and 1 channel of depth being convoluted by a filter of dimension 3×3, resulting a $3 \times 3 \times 1$ feature map. The filter's channel of depth should match the image's channel of depth. For instance, if a RGB image is fed as input, the filter must have three channels of depth, one to convolve with each channel of the image.

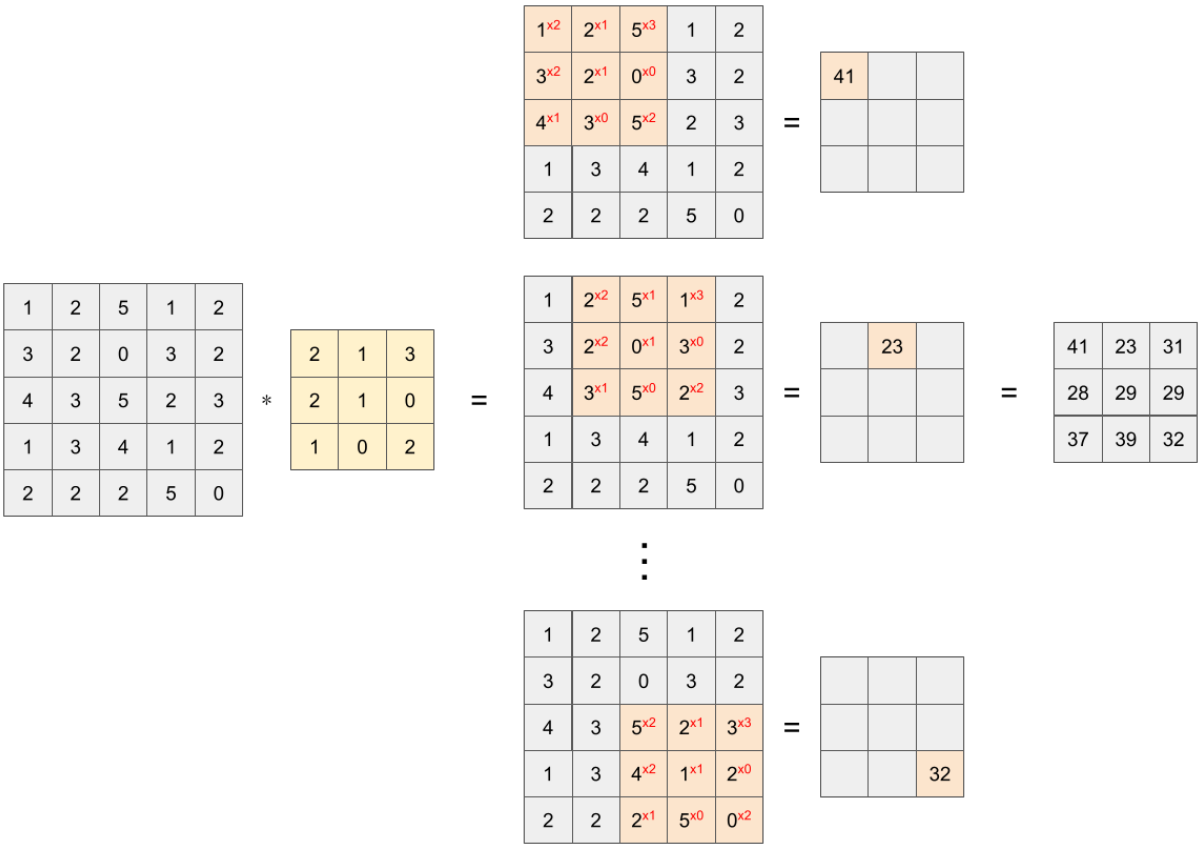

Figure 6 – Illustration of a convolution operation. The input image (gray) is of dimensions 5×5. The filter (yellow) is of dimensions 3×3. The resulting feature map is of dimensions 3×3 (Source: the author).

2.1.2.4 Generative Adversarial Networks

Currently, there is great interest in studying generative models due to their ability to learn the distribution of datasets and to be able to measure or sample these distributions, even though they have a large number of dimensions. This characteristic of generative models allows them to be applied in different areas, such as image generation (KARRAS et al., 2020), photo-realistic image resolution (LEDIG et al., 2017) and text to image synthesis (REED et al., 2016).

A Generative Adversarial Network (GAN) is composed of two differentiable functions. Usually an artificial neural network is used for both functions, named discriminator

(D) and generator (G) networks. The discriminator network is trained using supervised learning techniques and is responsible for identifying whether a data presented to it belongs to the dataset (real) or not (fake). On the other hand, the goal of the generator network is to create samples with structural characteristics that resembles the distribution of real data in order to deceive the discriminator network. The relationship between the generator and the discriminator networks during training is shown in Figure 7. Each network has

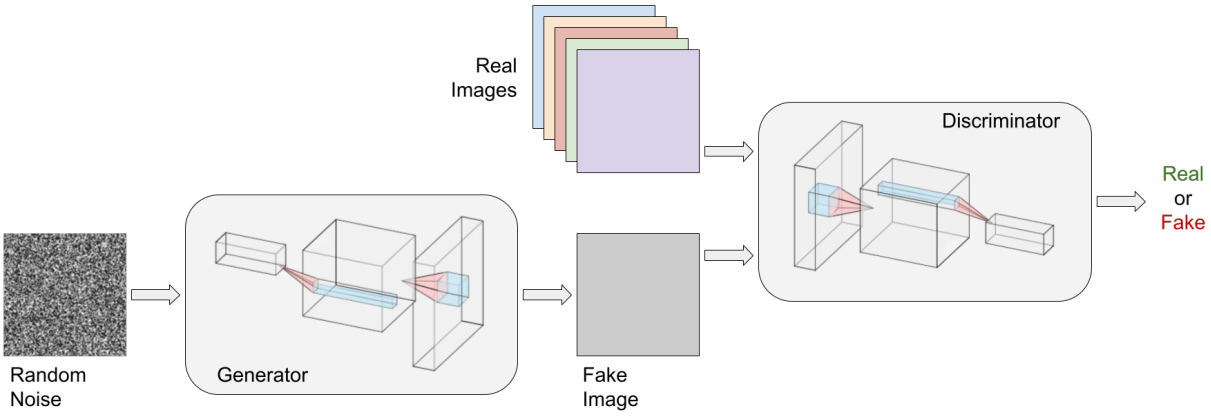

Figure 7 – Illustration of a GAN. The generator generates a fake-image from random noise. A discriminator receives as input real and fake images. The goal of the generator is to generate a fake-image that resembles the real ones. The goal of the discriminator is to correctly classify the input images to real or fake. (Source: the author).

a set of parameters to be optimized according to the loss functions $J^D\left(\theta^D, \theta^G\right)$, for the discriminator network, and $J^G\left(\theta^D, \theta^G\right)$, for the generator network. However, each network updates only their respective parameters. The loss function of the discriminator network can be given by[1] distribution p. The expected value can be seen as the sum of the value of every sample in a given distribution multiplied by its probability, i.e., a kind of average.:

$$J^D\left(\theta^D, \theta^G\right) = -\frac{1}{2}E_{x\sim p_{data}} \log D(x) - \frac{1}{2}E_{z\sim p_z} \log(1 - D(G(z))) \tag{2.3}$$

where G is the generator network, D is the discriminator network, θ^G is the set of parameters for the generator network, θ^D is the set of parameters for the discriminator network, x is a randomly chosen image from the dataset *data*, z is a variable from a random noise distribution p_z.

The more the generator network is able to produce data that deceive the discriminator network, the more success it will be making. Therefore, the loss function of the generator network can be given as follows:

$$J^G = -J^D \tag{2.4}$$

This scenario can be described as a zero-sum game where two players depend on each other's parameters, but can only control their own (GOODFELLOW, 2016). In other

[1] $E_{x\sim p}$: x is the expected value of a random variable, subjected to the

words, D and G play the following two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(G, D) = E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p_z} [\log(1 - D(G(z)))] \tag{2.5}$$

In the GAN training process, the networks are normally trained with gradient descent techniques to find the smallest value in the loss function. Usually, the weights of the networks are updated separately, where the weights of the discriminator network are updated first using samples from the real dataset and samples from the dataset generated by the generator network. Then, the weights of the generator network are updated using the loss functions discussed above.

Although GAN has achieved impressive results with a variety of architectures (PAN et al., 2019), there are some difficulties regarding their training process (THANH-TUNG; TRAN; VENKATESH, 2019). One of the biggest problems with the training of GANs is non-convergence. As the optimization problem involves two players, the optimal scenario would be when one player improves his parameters, he also improves the equilibrium with the other player. But in practice, when one player improves, he undoes the other players progress causing oscillations in GAN.

A well-known type of non-convergence is called mode collapse (SRIVASTAVA et al., 2017) and refers to the situation in which the generator network starts to map different values of the noise variable z to the same output value. This situation causes the generator network to always generate the same image for a subset of z values (partial) or for any z value (total). While the total mode collapse is rare, the partial is frequent.

2.2 Object Detection

Object detection corresponds to one of the main and most classic problems investigated within the field of Computer Vision, not only for its complexity but also for its relevance for performing other tasks related to vision: from segmentation, pose estimation, tracking to image captioning, several algorithms are based on object detection as an essential preliminary step for their execution (LIU et al., 2020).

Given an image $I(x, y)$, the task of detecting objects consists in pointing out, with confidence $c \in [0, 1]$, the location (u, v) and the spatial dimensions (w, h) of instances of objects belonging to a set $L = \{l_1, ..., l_N\}$ of N previously known categories (e.g., person, car, bicycle) or the nonexistence of such instances. The output of a detector corresponds, therefore, to the set $D = \{d_1, d_2, ..., d_i, ...\}$, with $d_i = [u, v, w, h, c, l_j]$ being the detection referring to the i-th object located in the image $I(x, y)$ with category l_j. Note that this task differs from mere classification, which is limited to assigning a single category to the entire image based on its main content. Figure 8 illustrates the difference between such tasks. Although distinct, the detection task can be understood as an extension of the

Figure 8 – Comparison between image classification (left) and object detection (right) tasks. While the first consists of assigning a single category to the image based on its main content, the second aims to point out the location, spatial dimensions and category of different objects contained in the image. (Source: the author).

classification problem, as it is possible to apply the same classifier over different regions of an image, in order to identify the location of the objects of interest. Indeed, this is how most of the object detection algorithms proposed in the literature solve this problem.

Among the first detectors based on CNN, are the region based detectors. It originated with the R-CNN detector (GIRSHICK et al., 2014), which is illustrated in Figure 9. As well as more classical detection methods, R-CNN performs object detection based on the classification of different regions of the image. In order to select such regions, the selective search method is applied to the image (UIJLINGS et al., 2013). This method is based on the hierarchical grouping of visually similar regions in terms of colors, textures, shapes and sizes, such grouping being carried out through a super segmentation algorithm (FELZENSZWALB; HUTTENLOCHER, 2004). The sub images for each region obtained by the selective search method are then rescaled and entered into a CNN. This, in turn, is responsible for extracting relevant visual characteristics, which are finally presented to a SVM classifier and a regressor. The first has the function of pointing out the class of the object supposedly contained in the analyzed region. The second plays the role of adjusting the dimensions of the proposed region, aiming to correct possible flaws in the selective search algorithm. As illustrated, the R-CNN method presents as main bottleneck the application of the same CNN over different regions of the analyzed image, which prevents it from being used in real-time applications. Furthermore, the selective search algorithm used by R-CNN does not have any learning capability, which limits the quality of the proposed regions. In order to overcome such bottlenecks, other region based detection methods were developed. Directly inspired by the R-CNN, the Fast R-CNN method (GIRSHICK, 2015) proposes the application of the entire image in the same CNN only once, in order to generate a map of convolutional features. This map corresponds to an image with multiple channels obtained after the convolution of several spatial filters on the input image of

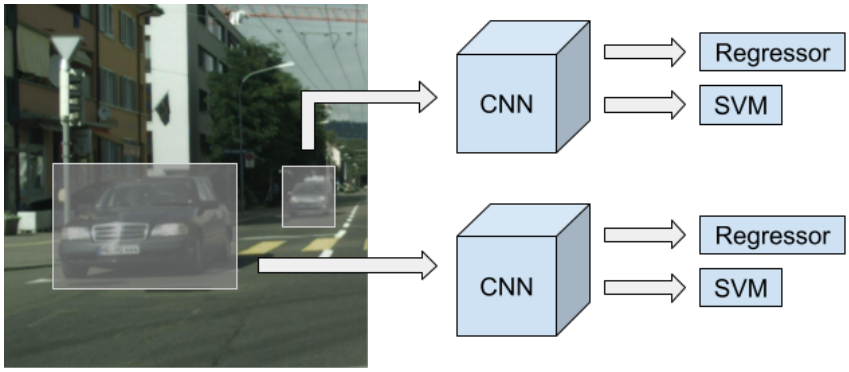

Figure 9 – Illustration of the R-CNN detector (GIRSHICK et al., 2014). After applying the method of selective search (UIJLINGS et al., 2013), the regions found are rescaled and presented to a CNN. It extracts relevant characteristics from each sub-image which are presented to an SVM classifier and a regressor. The first is responsible for indicating the class of the possible object contained in the region, while the second corrects the dimensions of the region proposed by the selective search algorithm.

the network. The regions proposed by the selective search method are projected on this map. Based on these regions, sub-maps are extracted, which are rescaled and presented to a classifier and a regressor. These are used in a similar way to the process performed by the R-CNN method. Figure 10 illustrates the behavior of such a detector. Both the

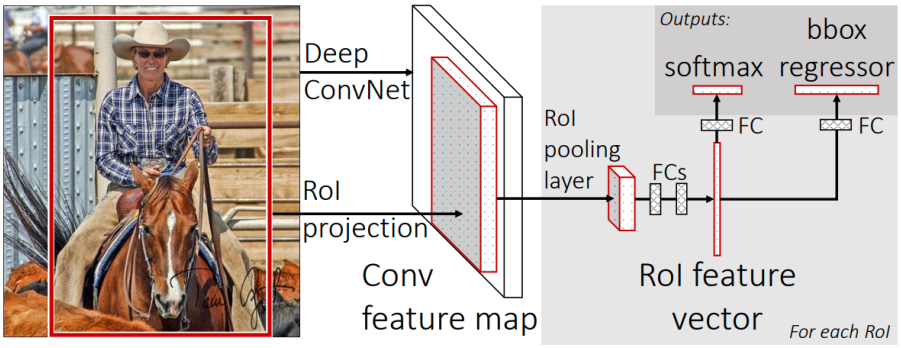

Figure 10 – Illustration of the Fast R-CNN detector (GIRSHICK, 2015). A CNN is used to extract the convolutional features. Then, the region of interest (RoI) are obtained via a selective search method, projecting the regions found over the image. Similarly to R-CNN, each projection is passed through a classifier and a regressor, which are respectively responsible to predict the object class and to adjust the region proposals found by the selective search algorithm. FC stands for fully-connected layer.

R-CNN detector and its successor Fast R-CNN are based on the application of the selective search method to obtain the regions of interest to be analyzed. As previously stated, this method does not have learning capability, which limits the quality of the proposed regions and, consequently, the quality of the final detections. In order to overcome this bottleneck, a second R-CNN extension, the Faster R-CNN algorithm (REN et al., 2015), replaces the selective search method by applying a CNN designed exclusively for the

proposals of regions of interest . Figure 11 illustrates the behavior of such a detector. As in Fast R-CNN, initially a single CNN is applied over the entire input image, in order to obtain a convolutional feature map. Then, this map is presented to a CNN called RPN (Region Proposal Network), which aims to propose the regions in which there is greater probability of finding objects of interest. Once proposed, such regions are projected onto the convolutional feature map. Finally, as in the region based methods discussed above, the projections obtained are presented to a classifier and to a regressor. Due to their

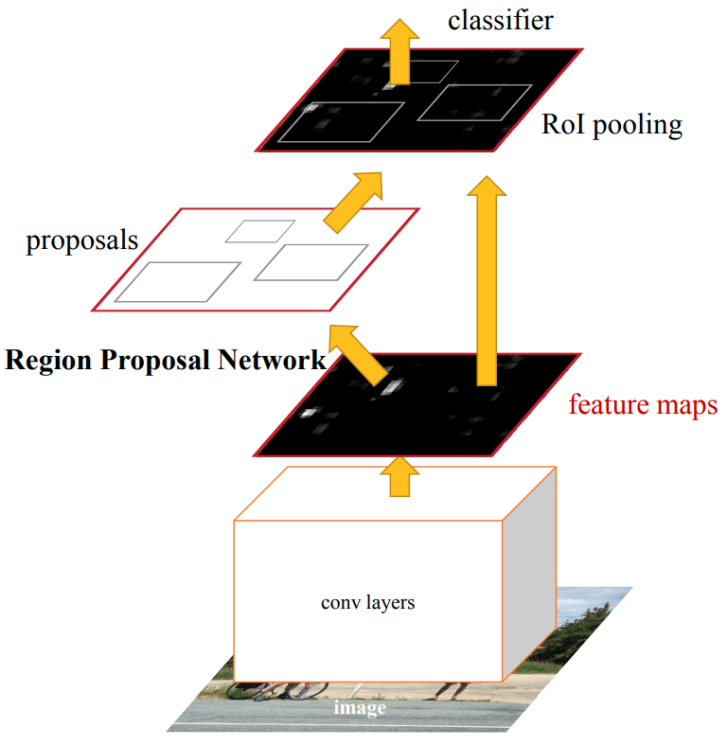

Figure 11 – Illustration of the Faster R-CNN detector (REN et al., 2015). First, a convolutional feature map is extracter from the input image using a CNN. Then, this map serves as input to a RPN, which proposes regions of the image where there is higher probability of finding objects. The region proposals are projected over the feature maps. Finally, the projections are passed through a classifier and a regressor, similarly as in R-CNN and Fast R-CNN.

promising results, such detectors influenced the development of new methods also based on convolutional networks. This is the case of the R-FCN (DAI et al., 2016) and Mask R-CNN (HE et al., 2017) detectors, also of the region based type, and the so-called single shot methods, such as the SSD detector families (LIU et al., 2016), YOLO (REDMON et al., 2016) and EfficientDet (TAN; PANG; LE, 2020), capable of operating in real time. Thus, the use of convolutional networks as feature extractors began a new era in the field of object detection research.

2.3 Unsupervised Image to Image Translation

Image-to-image translation is a class of vision and graphics problems which aims to obtain a model capable to map an input image to a desired output image. Investigating the GAN framework based on conditional priors, Isola et al. (2017) extended GANs to conditionally translate images in a novel framework named as pix2pix. In order to learn the mapping between an input image (x) and an output image (y), this technique requires a set of aligned image pairs in the training process. In essence, the pix2pix framework adds an encoder network to the generator, mapping a given image x to its representative code in a latent space. While the discriminator remains unchanged, a loss L_1 is employed to the generator, objectifying to output images near to the desired output y.

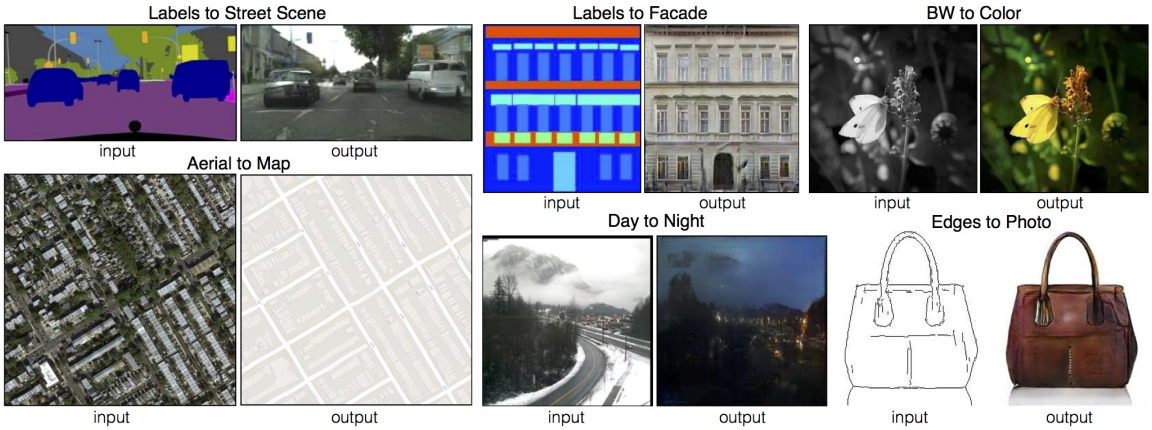

Figure 12 – Sample results with the pix2pix framework where in each case the same architecture and objective was used, changing only the training data. (ISOLA et al., 2017).

Although the technique has shown impressive results, as depicted in Figure 12, the requirement of paired images narrows its applicability to only supervised applications. Such constraint hinders its use in most of real-life scenarios that do not have paired training data available, e.g., two images with the same objects in the same positions but in different weather conditions.

In contrast, Zhu et al. (2017) presented an unsupervised approach, called CycleGAN, which is capable of learning to translate an image from a source domain X to a target domain Y in the absence of paired examples.

The goal of the CycleGAN (illustrated in Figure 13) is to learn a two-way mapping functions G and F, so that the distribution of images generated by G is indistinguishable from the distribution Y using an adversarially trained discriminator D_Y, and vice-versa for $F : Y \rightarrow X$ with D_X. In addition, for each image $x \in X$, a cycle image translation should be able to bring x back to the original image, i.e., $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$. Named by the authors as a cycle consistency constraint, the backward cycle translation should also be satisfied: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$. To enforce this transitivity mapping between the

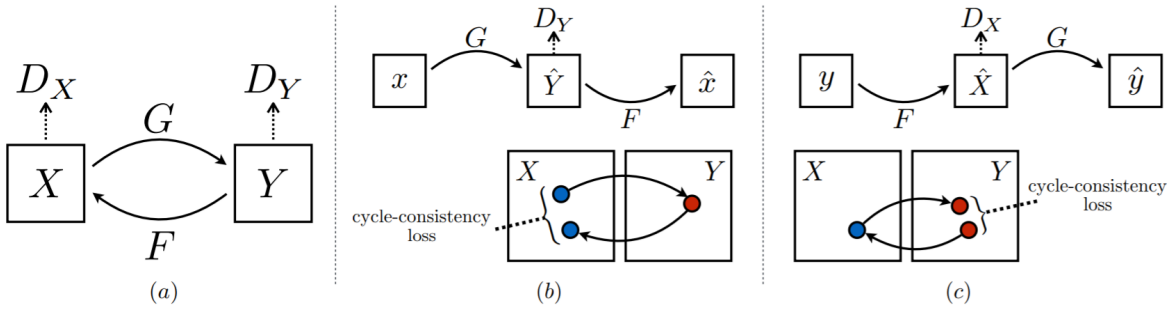

Figure 13 – (a) Overview of the CycleGAN. (b) The mapping function $G : X \rightarrow Y$ and the associated adversarial discriminator D_Y. (c) The backward mapping function $F : Y \rightarrow X$ with D_X. The cycle-consistency loss is minimized in both G and F (ZHU et al., 2017).

domains, a ℓ_1-norm is employed in order to minimize $|F(G(x)) - x|$ and $|G(F(y)) - y|$, for $x \in X$ and $y \in Y$. Figure 14 shows some results obtained from applying CycleGAN to perform unpaired image translation between two distinct domains.

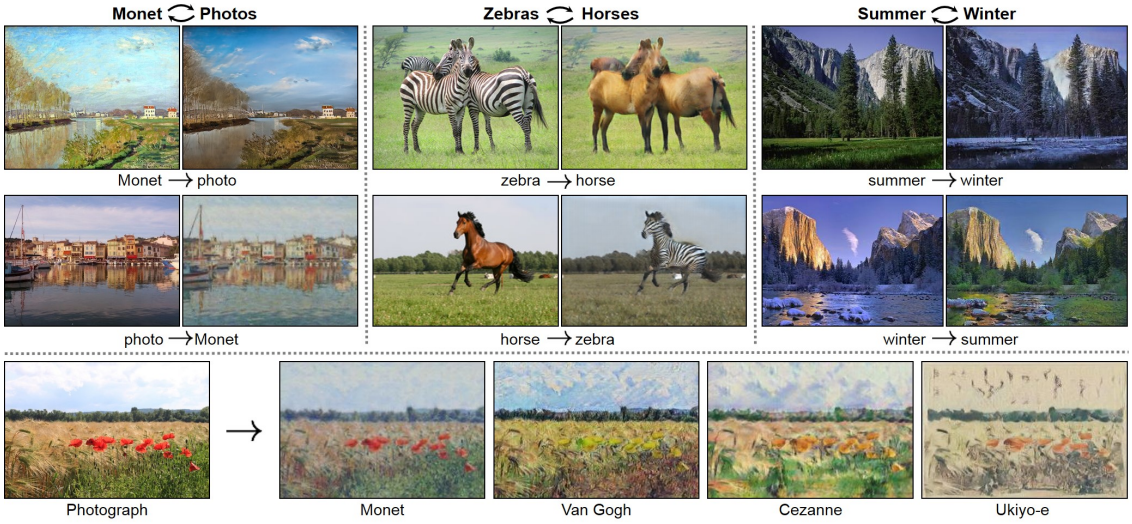

Figure 14 – Some sample results obtained with the CycleGAN framework performing unpaired image translation between two distinct domains (ZHU et al., 2017).

2.4 Neural Style Transfer

Style transfer is a class of vision problem that aims to combine a particular style of painting to a certain content image (JING et al., 2019). Style transfer can be traced back to the research of image texture synthesis. At that time, the most important idea used was that texture can be described by the statistical model of the local features of the image. In addition to image recognition and image classification, CNNs can also be used for style transfer, thus being called Neural Style Transfer (NST).

Among the NST works, a simple framework is to employ the encoder-decoder

approach using a Adaptative Instance Normalization (AdaIN) (HUANG; BELONGIE, 2017), referred from now on as AST-AdaIN. The model works by encoding using a CNN-based network (a VGG network (SIMONYAN; ZISSERMAN, 2014)), both content and style images, and mapping the mean and variance of the content feature to match those of the style image. The adjusted feature is then decoded back to the image space, but, is resembling the style of the style image. This approach does not require paired images and it only tries to match low-level features of the style images, such as textures and colors. Figure 15 illustrates the AST-AdaIN model. Formally, an encoder $E(\cdot)$ receives the content

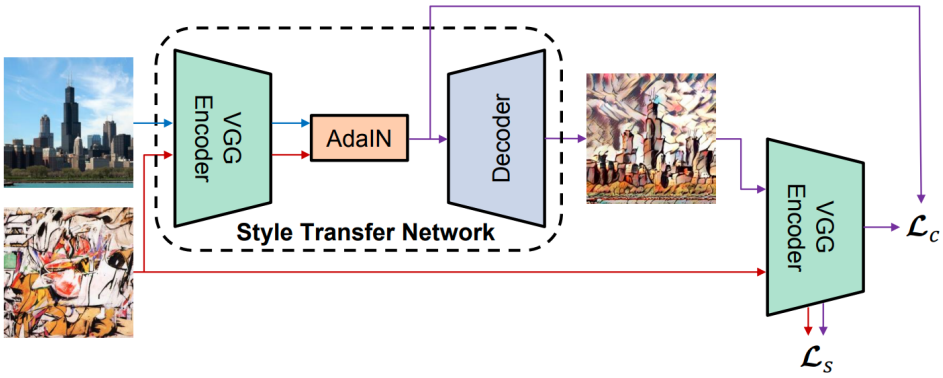

Figure 15 – Illustration of the AST-AdaIN model. Features of the content and style images are obtained by encoding them using a VGG network. The mean and variance of the content feature are adjusted to match those of the style image. The adjusted feature is then decoded back to the image space. (Source: (HUANG; BELONGIE, 2017))

and style images, generating its respective features maps, $E(c) = h_c$ and $E(s) = h_s$, where c and s are the content and style images, respectively, with their respective representations in the feature space h_c and h_s. The Adaptative Instance Normalization layer aligns the mean and variance of the content feature maps to those of the style feature maps, producing the styled feature maps $\hat{h}_s$:

$$\hat{h}_s = \text{AdaIN}(h_c, h_s) = \sigma(h_s)\left(\frac{h_c - \mu(h_c)}{\sigma(h_c)}\right) + \mu(h_s) \tag{2.6}$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ are the functions to extract the mean and variance, respectively.

A decoder $D(\cdot)$ is trained to decode features in the feature space to the image space, thus, enabling to retrieve the styled image, $D(\hat{h}_s)$. The training of the encoder-decoder leverages a content and a style loss. The content loss aims to minimize the difference between the aligned feature maps with the encoded feature maps of the styled feature maps, as following:

$$\mathcal{L}_c = \left\| E(D(\hat{h}_s)) - \hat{h}_s \right\|_2 \tag{2.7}$$

The style loss aims to minimize the difference between the mean and variance features

along the encoder layers, as following:

$$\mathcal{L}_s = \sum_{i=1}^{L} \left\| \mu(\phi_i(D(\hat{h}_s))) - \mu(\phi_i(s)) \right\|_2 + \sum_{i=1}^{L} \left\| \sigma(\phi_i(D(\hat{h}_s))) - \sigma(\phi_i(s)) \right\|_2 \tag{2.8}$$

where each ϕ_i denotes a layer of the network used to compute the loss.

After the training of the AST-AdaIN, the model is ready to transfer the image styles, by feeding a content and a style images into $E(\cdot)$, producing the h_c and h_s, then, $D(\text{AdaIN}(h_c, h_s))$, producing a new and corresponding styled image.

In practice, a style weight can be used to control how much of the style should be transferred:

$$output = D((1-\alpha)h_c + \alpha \text{AdaIN}(h_c, h_s)) \tag{2.9}$$

where the input image is recovered when $\alpha = 0$ and the most stylized image is synthesized when $\alpha = 1$. The Figure 16 shows how the α value influences the synthesized image.

Figure 16 – Illustration of the effect of different choices for α. (Source: (HUANG; BELONGIE, 2017))

2.5 Unsupervised Domain Adaptation for Object Detection

Let X be the input space (or image space) and let Y be the output space (or label space). The objective is to learn a hypothesis $h : X \to Y$ able to attach a label from Y to an example from X. This model is learned from a set of learning samples $S = \{(x_i^s, y_i^s) \in (X \times Y)\}_{i=1}^m$.

In traditional supervised learning, without domain adaptation, the assumption is that the examples $(x_i^s, y_i^s) \in S$ are independent and identically drawn from a distribution (i.i.d.) D_S of support $X \times Y$. Then, the objective is to learn h from S such that the error for labelling new examples coming from the distribution D_S is as minimal as possible.

But in unsupervised domain adaptation learning, two different but related distributions are considered, D_S and D_T, where labeled samples $(x_i^s, y_i^s) \in S$ are drawn from D_S, and unlabeled samples $(x_j^t) \in T$ are drawn from D_T. Then, the domain adaptation task consists of the transfer of knowledge from the source domain D_S to the target one D_T. Thus, the objective is to learn h, from samples coming from the two domains, such that the error is as minimal as possible in the target domain D_T.

2.5.1 Related Works

Addressing domain adaptation for object detection, Chen et al. (CHEN et al., 2018) proposed the Domain Adaptative Faster R-CNN (DA-Faster), that tackles the domain shift at two feature levels. The proposal is built on top of the Faster R-CNN model by adding two domain adaptation components, aiming to minimize the domain discrepancy at image- and instance-level representation, respectively. Both are small networks adversarially trained to classify the domain of the feature. In short, the components enforce the domains of the features to be indistinguishable. However, as argued by Saito et al. (SAITO et al., 2019), a domain-shift minimization at image-level may hurt the performance for domains with large discrepancies by also aligning background objects. Trying to address this shortcoming, Saito et al. (SAITO et al., 2019) proposed the Strong-Weak Distribution Alignment (Strong-Weak DA). The model employs two domain classifier networks, one at local-level, in the middle of the feature extractor, and other at image-level, right after the feature extractor. Both domain classifiers are also trained in an adversarial manner. The local-level network tries to strongly minimize the domain discrepancies only at low-level representations, such as color or texture. Conversely, the image-level network weights more on hard-to-classify examples than the easy ones, hence a weak domain alignment. However, some domains may still require a strong domain-shift minimization at image-level. In addition, the alignment of the features is not guaranteed to preserve the essential information of the target domain. Since annotations are not available at training time, the optimization procedure may find a shortcut to fool the domain classifier (GEIRHOS et al., 2020).

With a similar approach to our method, Shan, Lu e Chew (2019) proposed a model integrating a domain adaptation framework strongly based on CycleGAN and an object detector very similar to Faster R-CNN. This model is trained in an end-to-end manner, leading to a large consumption of GPU resources, requiring a GPU capable to simultaneously host a CycleGAN and a Faster R-CNN during training.

Using different strategies, Shen et al. (2019) proposed a gradient detach based stacked complementary losses method that uses a combination of several losses in different network layers along with gradient detach training. Zhu et al. (2019) proposed a two-fold method, first by finding pertinent regions that cover the objects of interest using k-means clustering and then aligning the target regions to the source distributions in an adversarial manner. Li et al. (2020) constructed a spatial pyramid representation with multi-scale feature maps, capturing context information of objects at different scales to further be attended softly to extract features for adversarial learning. Although promising, in contrast to our method, these methods are more laborious to implement, having several additional parameters to be trained with different losses.

Despite of the efforts made by the state-of-the art methods, there is still a large gap

between them and the upper-bounds (when training with target data). While aiming at reducing this performance gap, our approach has advantages and limitations. For instance, the state-of-the-art methods have the advantage of being single-stage methods, requiring the training of a single model. In contrast, our method is a two-stage approach, requiring the training of an image translation model and the generation of a novel dataset in the target domain before training the object detector. However, as a benefit, the adopted image translation models translate the image from the source to the target domain at pixel-level (raw image). With this approach, it becomes harder for the optimization procedure not to consider all the essential information. In addition, there is no need to align the features of the object detector, employed in the second stage, since the input data are already in the target domain. Also, the one-stage methods require more GPU resources as they have more parameters to train and additional losses. Conversely, our two-stage approach employs two simple models, performing one training at a time, requiring less GPU resources. Moreover, as the second stage is a simpler model (a standard Faster R-CNN), the inference is performed with higher FPS than the aforementioned methods. Although the more sophisticated methods might intuitively seem to perform better than the simpler two-stage methods (like the one proposed), our experiments suggest that this is not true for most of the evaluated scenarios.

3 Proposed Method

The proposed method, illustrated in Figure 17, is two-fold. Firstly, an image translation model is trained with unpaired images of source and target domains, to artificially generate images in the target domain. The objective of the model is to translate only the appearance between the domains, retaining unaltered the location, pose and co-occurrences of the objects. Therefore, the fake-images annotations (classes and bounding boxes) can be inherited from their respective versions in the source domain, comprising together the fake-target dataset. The second step is training an object detector with the artificially-generated dataset to detect objects in a domain which an annotated dataset was not previously available. Finally, the trained object detector can be used to infer objects in the real images of the target domain.

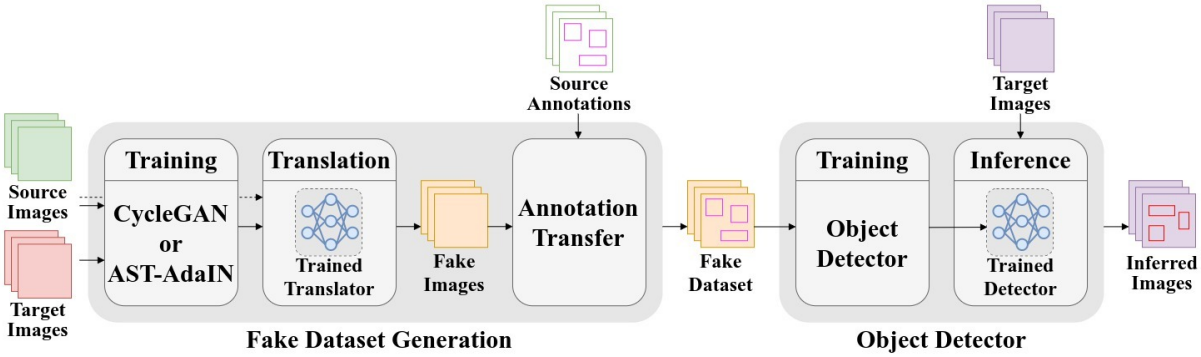

Figure 17 – Overview of the proposed method. Firstly, an unsupervised image translation model is trained with unpaired source and target images. Then, the source image set is translated to its fake-target version. The annotations of source images are directly transferred to the fake-target images, composing the fake-target dataset. Finally, an object detector is trained resulting in an object detector trained on a domain without previous annotations.

As the image translation and the detection are decoupled procedures, the computational performance of the inference (time spent to infer each image) depends only on the chosen object detection model, Faster R-CNN in this work.

3.1 Fake Dataset Generation

The fake dataset generation produces fake-images which resemble the target domain along with their respective annotations, providing training data for the object detector. This approach requires a set of annotated real images from the source domain and non-annotated real images from the target domain. The process is given in two steps: firstly, a model for unpaired image translation is trained with the annotated source images and

non-annotated target images. Secondly, the trained model is used to produce the fake-target images by translating the source images used in the training step. Their respective annotations are automatically inherited from the source domain. It is worthy to mention that since our goal is to obtain a fake-target dataset, we do not require that the translation model generalizes to other unseen images during the training phase. Also, once we acquired the fake-target dataset, the translation model is no longer necessary and can be discarded. In this work, we employed two models of image translation: an unpaired image-to-image translator and a neural style transfer model.

3.1.1 CycleGAN

For the unpaired image-to-image translation, we use the CycleGAN model. The CycleGAN is trained without supervision by processing two unpaired sets of images from the source and target domains. The model comprises two siblings generators, G_T and G_S, where G_T translates images from the source to the target domain and G_S translates back from the target to the source domain completing the cycle. Simultaneously, two siblings discriminators, D_T and D_S, are trained to classify whether a given image is coming from a real or a generated set, where D_T is responsible to distinguish between the real target images and those produced by G_T. Likewise, D_S and G_S are trained in the same manner but in the source domain. G_T and G_S are trained to produce confident images aiming to fool D_T and D_S, respectively. To encourage the coherence of the generated images, a cycle-consistency constraint (ZHU et al., 2017) is added to the loss in order to enforce to recover the source/target image when is translated back from the target/source domain. The cycle-consistency loss is defined as the ℓ_1-norms $|G_S(G_T(s)) - s|$ and $|G_T(G_S(t)) - t|$, where s and t are real source and target image samples from the training set, respectively.

With the CycleGAN trained, the generator G_T is used to translate each source image used in the training set to a corresponding fake-target image. Assuming the coherence between the translated image and its original, the annotations of the corresponding original images can be directly reused by the fake-image. Finally, the fake-target images and their respectively inherited annotations comprise the fake-target dataset.

3.1.2 AST-AdaIN

We used AST-AdaIN for the neural style transfer. As CycleGAN, the AST-AdaIN model is also trained with unpaired images from both source and target domains. An encoder $E(\cdot)$ receives the unpaired images from source and target domains, generating their respective features maps: $E(s) = h_s$ and $E(t) = h_t$, where s and t are the source and target images, respectively, with their respective representations in the feature space h_s and h_t. Following Equation 2.6, with h_s and h_t as the content and style feature maps, respectively, the AdaIn layer aligns the mean and variance of h_s to match those of h_t,

producing the fake-target feature maps $\hat{h}_t$.

A decoder $D(\cdot)$ is trained to learn the mapping from the feature space to the image space, thus enabling the generation of a fake-target image, $D(\hat{h}_t)$. The training of the encoder-decoder relies on the content and a style losses described by Equation 2.7 and Equation 2.8, respectively.

After the training of the AST-AdaIN, the model is ready to produce the fake-target images. Each image from the initial training dataset belonging to the source domain is fed into $E(\cdot)$, producing the h_s. To produce the h_t, we input into $E(\cdot)$ a randomly chosen image from the target domain dataset. Then, the AdaIN is employed and the resulting features are decoded using Equation 2.9, producing a new and corresponding fake-target image.

Likewise CycleGAN, image coherence between the source and target images was assumed, thus the fake-target images can inherit their corresponding source images annotations, comprising the fake-target dataset.

3.2 Object Detector

The detection stage uses a general purpose object detector to locate objects in the images. The training is performed with the previously generated fake-target dataset comprising a set of images and their respective annotations (bounding box along with its class). Among several existing models (GIRSHICK, 2015; REN et al., 2015; REDMON et al., 2016; LIN et al., 2017), this work adopts Faster R-CNN, which besides being one of the state-of-the-art deep detectors, enables fair comparison with (CHEN et al., 2018) and (SAITO et al., 2019).

Finally, with the trained object detection model, scenes in the target domain can be addressed. For each real target image, the detector predicts the bounding boxes of the objects and their respective class with a confidence score.

4 Experimental Methodology and Results

The general purpose of the experiments is to evaluate (and compare with the state of the art) the performance and robustness of our method in three challenging domain-shift scenarios proposed in previous works (CHEN et al., 2018; SAITO et al., 2019): (i) learning from synthetic data, in which annotation data is only available for synthetic rendered images (source domain) and the model has to detect objects in real-world images (target domain); (ii) driving in adverse weather conditions, in which the source domain comprises sunny day images and the target domain comprises foggy images; and, finally, (iii) cross-camera adaptation, in which the images of the source and target domains have a different aspect ratio, contrast, white balance, field-of-view, etc.. Since our focus is on autonomous driving, we used popular traffic-related datasets for evaluation of computer vision methods for self-driving cars. The following sections detail, respectively, the datasets, the performance metric, the setup for training the models, and the computational platform on which the experiments were carried out. Finally, we describe the experiments and present the results alongside with a discussion.

4.1 Datasets

Three datasets were used in the experiments. For the first scenario (learning from synthetic data), synthetic rendered images come from the Sim10k dataset (JOHNSON-ROBERSON et al., 2017) and real images come from Cityscapes (CORDTS et al., 2016) (Sim10k $\rightarrow$ Cityscapes). For the second scenario (driving in adverse weather), source and target images are from Cityscapes and Foggy Cityscapes (SAKARIDIS; DAI; GOOL, 2018), respectively (Cityscape $\rightarrow$ Foggy Cityscape). Cross-camera adaptation includes two subscenarios: (i) Cityscapes' images as source and KITTI's (GEIGER et al., 2013) as target (Cityscapes $\rightarrow$ KITTI) and (ii) vice-versa (KITTI $\rightarrow$ Cityscapes). Details of the datasets are provided in the following subsections.

The Cityscapes (CORDTS et al., 2016) dataset comprises urban 2048×1024 images for driving scenarios. The images were acquired using a car-mounted video camera while driving through several European cities with diverse scenarios. The original annotations are instance segmentation of the objects, thus it was necessary to adapt the annotations to bounding boxes coordinates as in (CHEN et al., 2018; SAITO et al., 2019) by taking the tightest rectangles of the instance masks. The dataset is originally divided in 2,975 images for training and 500 for validation. For our purposes, the training partition is used to train the adaptation models and to train the Faster R-CNN, whereas the validation images are used to test the detector. The dataset comprises a total of eight object categories: person,

Figure 18 – Samples of each dataset. The datasets samples are presented row-by-row in the following order (top to bottom): Cityscapes, Foggy Cityscapes, Sim10k and KITTI. The original aspect ratio of the images was preserved.

rider, car, truck, bus, train, motorcycle and bicycle. The top row of Figure 18 shows some samples from the training set.

Foggy Cityscapes (SAKARIDIS; DAI; GOOL, 2018) is an artificial dataset rendered with computer graphics techniques to add fog in the Cityscapes' images. The annotations, object classes and dataset split are the same as in the original Cityscapes dataset. To illustrate, the second row of Figure 18 shows the same samples from the training set of the Cityscapes in the foggy version.

Sim10k (JOHNSON-ROBERSON et al., 2017) is a synthetic dataset rendered by the game Grand Theft Auto V. The annotations are only available in the training partition. The dataset comprises 10,000 1914×1052 images with 58,701 bounding-box annotations of cars. Samples are shown in the third row of Figure 18.

KITTI (GEIGER et al., 2013) comprises 1250×375 images from several traffic objects, and it is originally divided into training and test sets, and only the training set is annotated. Since annotations are required for both training and evaluating the models, only the training images (7,481 samples) were used. Following previous work (CHEN et al., 2018), only annotations regarding cars were considered. Samples are shown in the last row of Figure 18.

4.2 Performance Metric

The performance of the object detector is measured by the mean Average Precision (mAP). Clearly, mAP has become the most popular measure for evaluating object detectors in the academic literature (GIRSHICK, 2015; REN et al., 2015; REDMON et al., 2016; TAN; PANG; LE, 2020), as well as in the main related competitions: PASCAL Visual Object Classes Challenge (PASCAL VOC) (EVERINGHAM; WINN, 2007; EVERINGHAM; WINN, 2011; EVERINGHAM et al., 2015) and Common Objects in Context Challenge (COCO) (LIN et al., 2014). The mAP summarizes the precision and recall curves into a single value, therefore taking into consideration both properties simultaneously. Unlike the traditional measures such as precision, recall, and F1 measures, mAP has the advantage of being independent of a particular threshold for the confidence score, which enables a fairer comparison.

The Average Precision (AP) of each object class is defined as the area under the precision-recall curve. This curve is built by calculating the precision and recall values of the accumulated true positive or false positive detections. For this, detections are ordered by their confidence scores, and precision and recall are calculated for each accumulated detection, i.e., the precision and recall of each ordered detection are calculated considering the current and preceding detections. Given the precision-recall curve, for the 2007 version (VOC'07) (EVERINGHAM; WINN, 2007), interpolated precision values are measured for 11 evenly divided recall levels, while for the 2012 version (VOC'12) (EVERINGHAM; WINN, 2011), interpolated precision values are measured for all recall levels. For this, for each recall level r, it is taken the maximum precision whose recall value is greater or equal than $r + 1$. Then, AP is calculated as the total area under the interpolated precision-recall curve. Finally, the mAP is calculated as the mean of the AP of all classes. As mAP is the mean of the APs across all classes, if only one class is provided, the mAP will be the same as the AP of the class.

While the works of the state-of-the-art methods (CHEN et al., 2018; SAITO et al., 2019) use the VOC'07 definition of the mAP, in this work we used the definition proposed in the PASCAL VOC Challenge 2012 (VOC'12) (EVERINGHAM; WINN, 2011), because it improves the precision and ability to measure differences between methods with low AP.

To keep the results comparable, we also computed the results of the state-of-the-art methods with the VOC'12 metric. It is important to note that this update of the metric does not worsen the comparison, but makes it more accurate.

4.3 Training Setup

In this section, the details and hyperparameters used for training the translation and detection models are presented.

4.3.1 Faster R-CNN

The Faster R-CNN feature extractor was initialized with ResNet-101 (HE et al., 2016) pre-trained on the ImageNet dataset (DENG et al., 2009). Anchor scales and ratios were defined considering the application working range as $\{4, 8, 16, 32\}$ and $\{0.5, 1, 2\}$, respectively. For experiments with only one dataset, the learning rate was kept fixed to 0.001 for the first 50k iterations and linearly decayed to zero over the next 20k iterations, resulting in 70k iterations. For the experiments with two or more datasets, the same learning rate was kept fixed for the first 70k iterations and linearly decayed to zero over the next 30k iterations, resulting in 100k iterations. The training was carried out with one image per batch, each one having its smaller side resized to 500 pixels keeping the aspect ratio. Also, data-augmentation was performed by flipping the images horizontally with probability of 50%. A public source code[1] was used for carrying out the experiments.

4.3.2 CycleGAN

The architecture used was the same as in the original paper (ZHU et al., 2017), except by the 16×16 PatchGAN (ZHU et al., 2017; ISOLA et al., 2017) discriminator that has a smaller field of view (and fewer layers) than the original 70×70 PatchGAN discriminator. The model was trained with 100 epochs using a fixed learning rate of 0.0002 followed by 100 epochs with learning rate linearly decaying to zero, totaling 200 epochs. Only one image was used per batch, each one loaded and resized to 572×572 pixels (the aspect ratio may be deformed) followed by a random crop of 512×512 pixels. Data augmentation was employed by flipping the images horizontally with probability of 50%. The default values were used on the other hyperparameters. The adopted source code was developed by the CycleGAN's authors, and is publicly available[2].

4.3.3 AST-AdaIN

For the Arbitrary Style Transfer with AdaIN, we used the same architecture as in the original paper (HUANG; BELONGIE, 2017). During training, the input images are resized to 512×512 pixels followed by random cropping to 256×256 pixels. At test time, the images' dimensions are not modified since the model is fully convolutional. The training is conducted for 160k iterations and the images are processed in batches of 8

[1] <https://github.com/endernewton/tf-faster-rcnn>
[2] <https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix>

images. All other hyperparameters were set to the default values. For transferring the style from a source to a target image, we adopted $\alpha = 1$ (Equation 2.9) to generate fully stylized images. The source code employed in the experimentation is publicly available[3].

4.3.4 State-of-the-art Models

For comparison purpose, the state-of-the-art models DA-Faster (CHEN et al., 2018) and Strong-Weak-DA (SAITO et al., 2019) were used, training with the same hyperparameters described in their respective works. Among the four datasets experimented in this work, all four were adopted in the DA-Faster work and three in the Strong-Weak-DA work. Considering that all experiments conducted in their work employed the same set of hyperparameters, it was decided to keep their proposed values with the assumption that these were the best hyperparameters found by the authors. As our method uses ResNet-101 pre-trained on the ImageNet dataset, the backbone of the state-of-the-art models were changed from pre-trained VGG-16 to ResNet-101. The code used is publicly available for both models[4,5].

4.4 Experimental Platform

The experiments were carried out with an Intel Xeon E5606 2.13 GHz x 8 with 32 GB of RAM, and 1 Titan Xp GPU with 12 GB of memory. The machine was running Linux Ubuntu 16.04 with NVIDIA CUDA 9.0 and cuDNN 7.0 (CHETLUR et al., 2014) installed. The training and inference steps were done using the TensorFlow (ABADI et al., 2016) and PyTorch (PASZKE et al., 2019) frameworks, depending on the model. The training sessions took, on average, 3 weeks for the CycleGAN model, 20 hours for the AST-AdaIN model, and 12 hours for the Faster R-CNN model. For the state-of-the-art models (DA-Faster and Strong-Weak-DA), 24 training hours were taken, on average. In the used setup, CycleGAN translates images at an approximate rate of 6 frames-per-second (fps), AST-AdaIN transfers the style of the images at a rate of 11 fps, whereas Faster R-CNN performs detections at ≈ 7 fps. For comparison, due to the extra parameters, the state-of-the-art methods run at approximately 5 fps. In contrast, our simpler approach performs detections 1.4× faster.

4.5 Experiments and Results

The evaluation of the object detectors was carried out on the test partition of the target domain dataset. For some tasks, the detection system might need to work well in

[3] <https://github.com/naoto0804/pytorch-AdaIN>
[4] DA-Faster: <https://github.com/tiancity-NJU/da-faster-rcnn-PyTorch>
[5] Strong-Weak-DA: <https://github.com/VisionLearningGroup/DA_Detection>

Training Setting	Real Source	Fake CycleGAN	Fake AST-AdaIN
OURS-C		✓	
OURS-A			✓
OURS-S+C	✓	✓	
OURS-S+A	✓		✓
OURS-C+A		✓	✓
OURS-S+C+A	✓	✓	✓

Table 2 – Description of the training settings used in the experiments. The training settings are assembled from the annotated data that are available: real source training set (S), CycleGAN fake-data (C), and AST-AdaIN fake-data (A), comprising up to six different training settings.

both source and target domains, e.g., sunny and foggy driving scenarios in the same day, therefore we also evaluated the test sets of both domains when applicable.

For our approach, different training settings were evaluated according to the data available. These settings combine fake data, generated by the image translation process, and the source-domain training data, assembling up to three datasets into each setting. The training settings receive an acronym in the format: OURS-{S, C, A}, and are described in Table 2.

For comparison, we trained: the (i) source- and (ii) target-only as an empirical measure of the lower- and upper-bound, respectively, and the state-of-the-art models (iii) DA-Faster, and (iv) Strong-Weak DA. For source- and target-only, Faster R-CNN was trained directly with the source and target training sets, respectively, following the setup described in Section 4.3.1. The source-only was trained only in the source domain training set. The target-only, when evaluating only in the target domain, was trained only in the target domain, and when evaluating simultaneously on both test sets of source and target domains, the training was performed on the source and target domains training sets altogether.

4.5.1 Learning from synthetic data

This experiment intends to measure the ability of our method in adapting to real-world data while having access only to the annotations of the synthetic data. For this, the source and target domains datasets were Sim10k and Cityscapes, respectively. Following previous works (CHEN et al., 2018; SAITO et al., 2019), we considered only the car class in this experiment as it is the only class present in both datasets simultaneously. All the available training data was used (10k annotated images from Sim10k and 2,975 non-annotated images from Cityscapes) for training the adaptation models. The evaluation of the detector was only performed on the target domain since the system is not expected

Method	Real S	Real T	Fake C	Fake A	car AP
Source-only	✓				34.7
DA-Faster (CHEN et al., 2018)	✓	✘			27.3
Strong-Weak DA (SAITO et al., 2019)	✓	✘			31.7
OURS-C			✓		42.4
OURS-A				✓	43.7
OURS-C+A			✓	✓	**44.6**
Target-only		✓			58.9

Table 3 – Sim10k→Cityscapes: Results from training with annotated Sim10k (S) and non-annotated Cityscapes (T), evaluating on Cityscapes. The check-mark denotes which data was used during training, and if crossed, only non-annotated data was used. Fake data was acquired by translating Sim10k to Cityscapes using CycleGAN (C) and AST-AdaIN (A).

to run on synthetic scenarios like Sim10k.

In this work, due to the unavailability of the source-code, it was not possible to directly compare with the method proposed by Shan et al. (SHAN; LU; CHEW, 2019) and Zhu et al. (ZHU et al., 2019). Also, other works have recently appeared (SHEN et al., 2019; LI et al., 2020) but, despite the availability of the source-code, the requirement of editing them to match our settings may lead to a biased result. Although the experiments reported here are also present in their work, their method was built with the VGG-16 as the backbone and the results were reported using the VOC'07 metric. Exceptionally in this section, we added the results experimenting with our method matching their settings, using the VGG-16 as the backbone, and evaluating with the VOC'07 metric.

4.5.1.1 Results

The detection results are shown in Table 3. As it can be seen, there is a clear performance gap of 24.2 p.p. (58.9 vs. 34.7) between the empirical lower- and upper-bounds, indicating a need for methods that handle this domain adaptation problem. Our method outperformed source-only and both state-of-the-art models. Although training with the fake-data generated by both translation methods (OURS-C+A) attained the best result with 44.6% in AP (an increment of 9.9 p.p. over the highest baseline result), the simpler training settings (OURS-C and OURS-A) also surpassed the other models (for results using also the source domain to train, please see our supplementary material).

The AST-AdaIN method achieved a higher AP (+1.3 p.p.) when compared to the CycleGAN (43.7 vs. 42.4), which shows that it better translates synthetic to real images. This result is in line with the intuition that this scenario requires a more accurate alignment at the low-level image features, such as color and texture. Although Strong-

Figure 19 – Samples of translated images with their respective bounding boxes. The real source training images are shown in the first row and their respective fake-target versions are shown in the second and third row for the CycleGAN and AST-AdaIN models, respectively.

Weak DA promises strong low-level alignment, it may not preserve the essential (semantic) information. This suggests that pixel-level translation methods should be preferred over Strong-Weak DA. The lower performance of the DA-Faster model w.r.t. the source-only has already been reported in (SAITO et al., 2019).

Samples of translated images with their respective bounding boxes can be seen in Figure 19 (more samples available in the Appendix A). As illustrated, the translated models are capable of preserving the global scene structure, as well as the geometry of objects and their co-occurrence, i.e., the geometric relation between the elements from the real to the respective fake image. Despite some unwanted artifacts, as the gray-striped skies observed in the second row, the artificial images proved to be useful for training the detector.

4.5.1.2 Results with VGG-16 and VOC'07

The detection results are shown in Table 4. As it can be seen, our method outperformed source-only and the other models. Although training with the fake-data generated by both translation methods (OURS-C+A) attained the best result with 46.4% in AP, the simpler training settings using the AST-AdaIN method (OURS-A) showed to be sufficient to surpass the other models. This result reinforces the argument that low-level features are more important than high-level ones when adapting from synthetic to real images. Lastly, in this scenario, our method outperformed all state-of-the-art and other known methods for unsupervised domain adaptation for object detection.

Method	Real S	Real T	Fake C	Fake A	car AP
Source-only	✓				34.6
PDA+FDA (SHAN; LU; CHEW, 2019)	✓	✗			39.6
SCL (SHEN et al., 2019)	✓	✗			42.6
SCDA (ZHU et al., 2019)	✓	✗			43.0
SAPNet (LI et al., 2020)	✓	✗			44.9
OURS-C			✓		41.9
OURS-A				✓	45.0
OURS-C+A			✓	✓	**46.4**
Target-only		✓			61.2

Table 4 – Sim10k→Cityscapes: Results from training with annotated Sim10k (S) and non-annotated Cityscapes (T) using the VGG-16 as backbone, evaluating on Cityscapes with the VOC'07 metric. The check-mark denotes which data was used during training, and if crossed, only non-annotated data was used. Fake data was acquired by translating Sim10k to Cityscapes using CycleGAN (C) and AST-AdaIN (A).

Figure 20 – Samples of translated images with their respective bounding boxes. The real source training images are shown in the first row and their respective fake-target versions are shown in the second and third row for the CycleGAN and AST-AdaIN models, respectively.

4.5.2 Driving in adverse weather

In this experiment, the Cityscapes and Foggy Cityscapes datasets are used to evaluate the methods in a similar driving scenario but with a distinct weather condition. Both datasets have the same size, aspect ratio, train/test split, and also have paired instances, only differing by the presence or absence of fog. Although they have paired images, the proposed method does not use this information in order to resemble the real-world scenario where this pairing is virtually impossible.

Method	Real		Fake		mAP	
	S	T	C	A	T	S+T
Source-only	✓				27.1	36.3
DA-Faster (CHEN et al., 2018)	✓	✘			33.1	37.7
Strong-Weak DA (SAITO et al., 2019)	✓	✘			32.8	39.2
OURS-C			✓		39.4	-
OURS-A				✓	30.8	-
OURS-C+A			✓	✓	**39.5**	-
OURS-S+C	✓		✓		-	44.6
OURS-S+A	✓			✓	-	40.8
OURS-S+C+A	✓		✓	✓	-	**44.9**
Target-only		✓			44.8	-
	✓	✓			-	46.3

Table 5 – Cityscapes→FoggyCityscapes: Results from training with annotated Cityscapes (S) and non-annotated Foggy Cityscapes (T), evaluating solely on Foggy Cityscapes (T) or on both (S+T). The check-mark denotes which data was used during training, and if crossed, only non-annotated data was used. Fake data was acquired by translating Cityscapes using CycleGAN (C) and AST-AdaIN (A).

4.5.2.1 Results

The results for Cityscapes to Foggy Cityscapes scenario are shown in Table 5. As it can be seen, our method outperformed, again, source-only and both state-of-the-art models. The training setting OURS-C+A surpassed the compared models when evaluated on the target domain, achieving 39.5% in mAP. Evaluating on source and target domains (S+T) simultaneously, representing a scenario in which a single model should handle different weather conditions, our method also outperformed source-only and both the state-of-the-art models. When using real source images together with fake images from both translation models (OURS-S+C+A), our method achieved a performance comparable to the target-only, i.e., below by 1.4 p.p. (44.9 vs. 46.3). Since the image content (e.g., layout, objects) in this adaptation scenario is the same, a model that strongly translates the domains is expected to perform better, and indeed the CycleGAN-based models achieved better results for generating fake data compared with the AST-AdaIN ones. As it can be seen in Figure 20, AST-AdaIN struggled to generate realistic foggy images. For the AP of each class, remaining evaluations and more samples of translated images, please refer to Appendix A.

4.5.3 Cross-camera adaptation

An object detection system is expected to work well as generally as possible regardless of the hardware setup. In the context of autonomous driving, even datasets

Method	Real		Fake		car AP (C→ K)				car AP (K→ C)			
					Protocol 1		Protocol 2		Protocol 1		Protocol 2	
	S	T	C	A	T	S+T	T	S+T	T	S+T	T	S+T
Source-only	✓				74.3	71.9	74.1	68.2	29.1	80.0	28.6	65.2
DA-Faster (CHEN et al., 2018)	✓	✗			71.3	68.8	70.5	64.5	31.8	81.2	30.6	66.5
Strong-Weak DA (SAITO et al., 2019)	✓	✗			64.3	63.2	70.1	64.9	36.1	**82.1**	35.0	**68.8**
OURS-C			✓		75.0	-	74.8	-	34.9	-	36.0	-
OURS-A				✓	72.3	-	71.4	-	**38.7**	-	**36.8**	-
OURS-C+A			✓	✓	**75.9**	-	**76.5**	-	34.5	-	35.6	-
OURS-S+C	✓		✓		-	73.3	-	69.4	-	81.3	-	68.5
OURS-S+A	✓			✓	-	72.9	-	68.8	-	80.6	-	68.0
OURS-S+C+A	✓		✓	✓	-	**73.7**	-	**70.1**	-	80.8	-	68.3
Target-only		✓			88.2	-	84.9	-	58.9	-	58.9	-
	✓	✓			-	86.1	-	78.3	-	86.1	-	78.3

Table 6 – Cityscapes→KITTI (C→K): Results from training with annotated Cityscapes (S) and non-annotated KITTI (T), evaluating solely on KITTI (T) or on both (S+T). KITTI→Cityscapes (K→C): Results from training with annotated KITTI (S) and non-annotated Cityscapes (T), evaluating solely on Cityscapes (T) or on both (S+T). The check-mark denotes which data was used during training, and if crossed, only non-annotated data was used. Fake data was acquired by translating the source training set using CycleGAN (C) and AST-AdaIN (A).

collected on similar weather conditions and time of the day may exhibit large domain shifts due to the different image quality and resolution caused by distinct camera settings. Motivated by this fact, this experiment intends to measure how well can a method accurately detect objects on a target dataset whose images have been taken by a different camera setting than the source dataset.

The experiment was carried out using the Cityscapes and KITTI datasets. The datasets were employed in both directions, resulting in two experiments: (i) Cityscapes as source domain and KITTI as target domain; and (ii) KITTI as source domain and Cityscapes as target domain. Taking into account that KITTI has no test set available, two protocols were considered. The first, following the protocol used in (CHEN et al., 2018), uses KITTI original training partition as the test set for the experiment Cityscapes→KITTI. However, this protocol does not seem to be fair, given that it uses the training data (without the annotations) as test set. Therefore, for the second protocol, we randomly split the KITTI training set into 70% and 30% partitions for the training and test set, resulting in 5237 and 2244 images, respectively.

4.5.3.1 Results (Cityscapes→KITTI)

As shown in Table 6, our method outperformed source-only and both state-of-the-art models in both protocols. Overall, although the translation produced unwanted artifacts (Figure 21, more samples available in the supplementary material) such as the stain over the asphalt, the CycleGAN yielded better results than AST-AdaIN. This may be because KITTI has richer colors and textures in contrast to Cityscapes (see Figure 18), requiring a

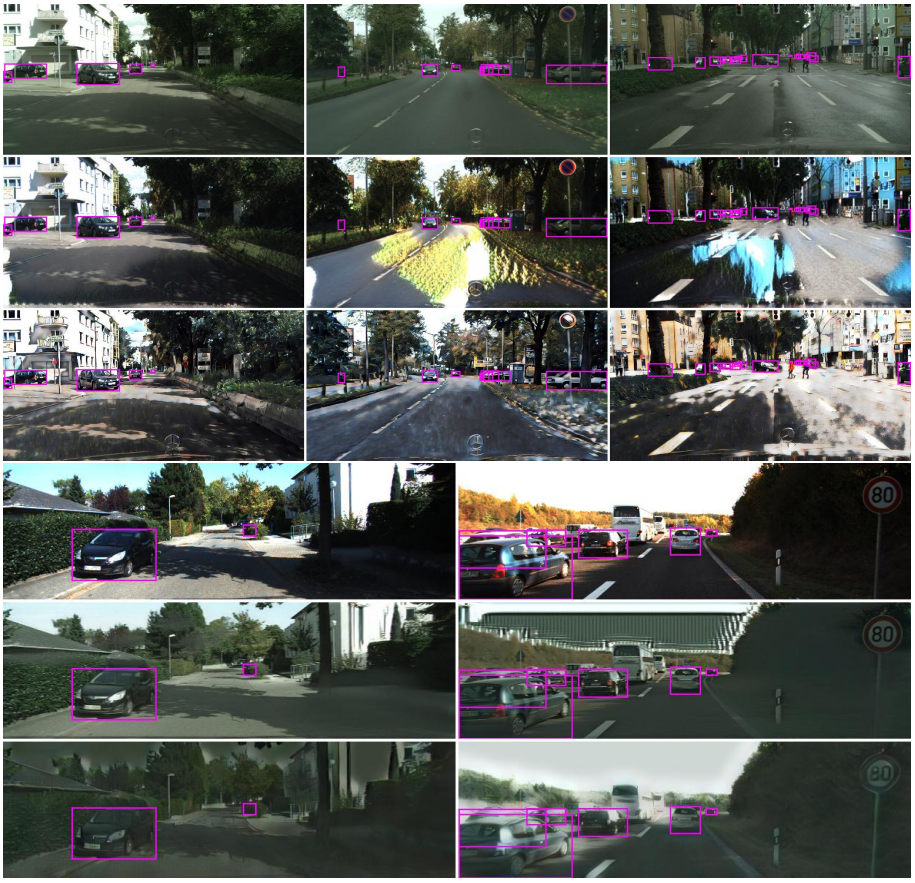

Figure 21 – Samples of translated images with their respective bounding boxes. Top: the translation from Cityscapes to KITTI, and vice-versa for the bottom one. The real source training images are shown in the first row and their respective fake-target versions are shown in the second and third row for the CycleGAN and AST-AdaIN models, respectively.

more complex model to translate the image domain. Nevertheless, although the results of OURS-A were below source-only (suggesting that the AST-AdaIN model did not transfer the style properly), when jointly trained with fake data from CycleGAN (OURS-C+A), it added significant information to the training, improving the AP from 75.0% to 75.9% and 74.8% to 76.5% when testing on target with protocol 1 and 2, respectively.

4.5.3.2 Results (KITTI→Cityscapes)

In the opposite direction of the previous scenario, our method outperformed source-only and both state-of-the-art models when testing on target domain. In contrast to previous scenario (Cityscapes→KITTI), the setting OURS-A produced (Table 6) the best results on both protocols when evaluating on the target domain, being suitable when targeting a less complex dataset, such as Cityscapes where images resembles more opaque colors when compared to KITTI.

Among the compared models, Strong-Weak-DA achieved the highest mAP, with 2.6 and 1.8 p.p. below our best results on protocol 1 and 2 (36.1 vs. 38.7 and 35.0 vs.

36.8), respectively. However, when evaluating on both domains, our best results achieved 81.3 and 68.5 compared to 82.1 and 68.8 from Strong-Weak-DA for protocols 1 and 2, respectively; a difference of only 0.8 and 0.3 p.p.. Although our method did not achieve the highest mAP in this last experiment, the small difference still makes our method comparable.

Regarding the protocols, protocol 1 reflects the results of the state-of-the-art methods, however, protocol 2 depicts a fairer setup by partitioning the data into train and test properly. Still, our results on both protocols remain consistent throughout the training settings. The remaining mAP results are available in the Appendix A.

4.6 Limitations

The effectiveness of the proposed method seems to be related to the quality of the translated images, i.e., how much the translated images resemble the target domain. Therefore, the limitations of the methods used for the image translation also apply to our approach. For instance, it is well-known that GAN-based models present unstable and hard-to-converge training, which may result in mode collapse, requiring re-training. Moreover, training these models is time-consuming, mostly because of their substantial amount of parameters, as in CycleGAN. In contrast, style-transfer methods not based on GANs, such as AST-AdaIN, are stable and less time-consuming. However, they are limited to translating only colors and textures without concerning about the semantics of the scene, being effective only for small domain-shifts. Moreover, they do not allow verifying the quality of the adaption as in two stage methods.

In addition, the proposed method may be limited to object sizes seen by the detector during training. If objects in the source domain have different sizes in the target domain, the effectiveness of the method will depend on the capability of the chosen object detector to work on object sizes not seen during training.

In this work, we did not include all results for the methods proposed in the works (SHAN; LU; CHEW, 2019; SHEN et al., 2019; LI et al., 2020; ZHU et al., 2019) due to the unavailability of the source-code or because it requires editing it to match our settings. Instead, we adapted our proposed method to match their settings (Section 4.5.1).

5 Conclusion

This work addressed the detection of objects across distinct domains by training a detector on annotated data from a source domain and non-annotated data from a target domain, which poses an Unsupervised Domain Adaptation problem. The proposed method tackles this challenging problem using a simpler approach compared to the state of the art, yet it achieves comparable or, in most cases, higher performance in several scenarios. The method uses unsupervised image-to-image translation and style-transfer models to generate annotated data in the target domain that are later used to train a deep object detector. The proposed two-stage approach has the benefit of giving access to the translated training images, from the source to the target domain at pixel-level. Therefore, it allows a visual inspection of the images before training the detector. Moreover, as the two stages are decoupled, our approach does not require additional GPU memory compared to the original object detector. In contrast, the state-of-the-art methods are single-stage approaches that aimed at minimizing the domain discrepancy at the cost of adding more complexity into their models and less interpretability.

The evaluation was performed on well-known relevant datasets for object detection in autonomous driving that resembles a diverse and realistic set of scenarios, including learning from synthetic data, driving in adverse weather, and cross-camera adaptation. In addition, we carried out an extensive comparison with the state of the art. Following, we provide a summary of the results aggregating our main achievements. Our method outperformed the state-of-the-art models by up to 9.9 p.p. in mAP. When learning from synthetic data, we outperformed the source-only and both state-of-the-art models on all training settings (for this experiment: OURS-C, OURS-A and OURS-C+A) from 7.7 to 9.9 p.p. in mAP improvement. For driving in adverse weather, we achieved 1.6 to 6.4 p.p. superior in five out of six different training settings. On the cross-camera adaptation, considering the two protocols, our method outperformed the source-only and both state-of-the-art models in 10 out of 12 training settings for the Cityscape to KITTI scenario, and 4 out of 12 training settings for the KITTI to Cityscapes scenario. An increase ranging from 0.6 to 2.6 p.p. in mAP.

The results show that our method is simple yet effective, outperforming the current state-of-the-art models in all three scenarios, reducing the gap toward the upper-bound. Moreover, the results demonstrate that our method can benefit from the fake-data generation models even when the qualitative results seem inaccurate and show some unwanted artifacts.

There is still room for improvements. For instance, in future studies, one should

investigate alternative methods for image translation and style transfer, especially those that could improve the semantic consistency of the translation. This will likely boost the performance of the object detector. In addition, future works should collect and experiment with new datasets, preferably in other domains and scenarios less related to the driving environment.

Bibliography

ABADI, M. et al. Tensorflow: Large-Scale Machine Learning On Heterogeneous Distributed Systems. In: *arXiv preprint arXiv:1603.04467*. [S.l.: s.n.], 2016. Citado na página 51.

ARRUDA, V. F. et al. Cross-domain object detection using unsupervised image translation. *Expert Systems with Applications*, Elsevier, p. 116334, 2021. Citado na página 26.

ARRUDA, V. F. et al. Cross-Domain Car Detection Using Unsupervised Image-to-Image Translation: From Day to Night. In: *International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2019. Citado 2 vezes nas páginas 24 and 25.

CHEN, Y. et al. Domain Adaptive Faster R-CNN For Object Detection In The Wild. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2018. Citado 14 vezes nas páginas 24, 41, 45, 47, 48, 49, 51, 52, 53, 56, 57, 69, 70, and 71.

CHETLUR, S. et al. cuDNN: Efficient Primitives For Deep Learning. In: *arXiv preprint arXiv:1410.0759*. [S.l.: s.n.], 2014. Citado na página 51.

CORDTS, M. et al. The Cityscapes Dataset For Semantic Urban Scene Understanding. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. Citado 2 vezes nas páginas 25 and 47.

DAI, J. et al. R-fcn: Object detection via region-based fully convolutional networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2016. p. 379–387. Citado na página 36.

DENG, J. et al. Imagenet: A Large-Scale Hierarchical Image Database. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2009. Citado na página 50.

EVERINGHAM, M. et al. The pascal visual object classes challenge: A retrospective. In: *International Journal of Computer Vision (IJCV)*. [S.l.]: Springer, 2015. v. 111, n. 1, p. 98–136. Citado na página 49.

EVERINGHAM, M.; WINN, J. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Development Kit. In: *University of Leeds, Tech. Rep.* [S.l.: s.n.], 2007. Citado na página 49.

EVERINGHAM, M.; WINN, J. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Development Kit. In: *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep.* [S.l.: s.n.], 2011. Citado na página 49.

FELZENSZWALB, P. F.; HUTTENLOCHER, D. P. Efficient graph-based image segmentation. *International journal of computer vision*, Springer, v. 59, n. 2, p. 167–181, 2004. Citado na página 34.

GEIGER, A. et al. Vision Meets Robotics: The KITTI Dataset. In: *The International Journal of Robotics Research (IJRR)*. [S.l.]: Sage Publications Sage UK: London, England, 2013. v. 32, n. 11, p. 1231–1237. Citado 3 vezes nas páginas 25, 47, and 48.

GEIRHOS, R. et al. Shortcut Learning in Deep Neural Networks. In: *Nature Machine Intelligence*. [S.l.]: Nature Publishing Group, 2020. v. 2, n. 11, p. 665–673. Citado na página 41.

GIRSHICK, R. Fast R-CNN. In: *International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2015. Citado 5 vezes nas páginas 14, 34, 35, 45, and 49.

GIRSHICK, R. et al. Rich Feature Hierarchies For Accurate Object Detection And Semantic Segmentation. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2014. Citado 3 vezes nas páginas 13, 34, and 35.

GOODFELLOW, I. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. Citado na página 32.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. [S.l.]: Pearson, 1998. Citado 2 vezes nas páginas 27 and 29.

HE, K. et al. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2961–2969. Citado na página 36.

HE, K. et al. Deep Residual Learning For Image Recognition. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. Citado na página 50.

HUANG, X.; BELONGIE, S. Arbitrary Style Transfer In Real-Time With Adaptive Instance Normalization. In: *International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2017. Citado 4 vezes nas páginas 14, 39, 40, and 50.

ISOLA, P. et al. Image-to-Image Translation With Conditional Adversarial Networks. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2017. Citado 3 vezes nas páginas 14, 37, and 50.

JING, Y. et al. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, IEEE, v. 26, n. 11, p. 3365–3385, 2019. Citado na página 38.

JOHNSON-ROBERSON, M. et al. Driving In The Matrix: Can Virtual Worlds Replace Human-Generated Annotations For Real World Tasks? In: *International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2017. Citado 3 vezes nas páginas 25, 47, and 48.

KARRAS, T. et al. Analyzing and improving the image quality of stylegan. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. [S.l.: s.n.], 2020. p. 8110–8119. Citado na página 31.

LEDIG, C. et al. Photo-realistic single image super-resolution using a generative adversarial network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 4681–4690. Citado na página 31.

LI, C. et al. Spatial attention pyramid network for unsupervised domain adaptation. In: *European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2020. Citado 5 vezes nas páginas 24, 41, 53, 55, and 59.

LIN, T.-Y. et al. Focal Loss For Dense Object Detection. In: *International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 23 and 45.

LIN, T.-Y. et al. Microsoft COCO: Common Objects in Context. In: *European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2014. Citado na página 49.

LIU, L. et al. Deep learning for generic object detection: A survey. *International Journal of Computer Vision (IJCV)*, Springer, 2020. Citado na página 33.

LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 21–37. Citado na página 36.

OLAH, C.; MORDVINTSEV, A.; SCHUBERT, L. Feature visualization. *Distill*, 2017. Https://distill.pub/2017/feature-visualization. Citado na página 29.

PAN, Z. et al. Recent progress on generative adversarial networks (gans): A survey. *IEEE Access*, 2019. Citado na página 33.

PASZKE, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2019. Citado na página 51.

REDMON, J. et al. You Only Look Once: Unified, Real-Time Object Detection. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. Citado 4 vezes nas páginas 23, 36, 45, and 49.

REED, S. et al. Generative adversarial text to image synthesis. In: PMLR. *International conference on machine learning*. [S.l.], 2016. p. 1060–1069. Citado na página 31.

REN, S. et al. Faster R-CNN: Towards Real-Time Object Detection With Region Proposal Networks. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2015. Citado 7 vezes nas páginas 14, 23, 24, 35, 36, 45, and 49.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 27.

SAITO, K. et al. Strong-Weak Distribution Alignment For Adaptive Object Detection. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2019. Citado 14 vezes nas páginas 24, 41, 45, 47, 49, 51, 52, 53, 54, 56, 57, 69, 70, and 71.

SAKARIDIS, C.; DAI, D.; GOOL, L. V. Semantic Foggy Scene Understanding With Synthetic Data. In: *International Journal of Computer Vision (IJCV)*. [S.l.]: Springer, 2018. v. 126, n. 9, p. 973–992. Citado 3 vezes nas páginas 25, 47, and 48.

SHAN, Y.; LU, W. F.; CHEW, C. M. Pixel and Feature Level Based Domain Adaptation for Object Detection in Autonomous Driving. In: *Neurocomputing*. [S.l.]: Elsevier, 2019. v. 367, p. 31–38. Citado 4 vezes nas páginas 41, 53, 55, and 59.

SHEN, Z. et al. Scl: Towards accurate domain adaptive object detection via gradient detach based stacked complementary losses. In: *arXiv preprint arXiv:1911.02559*. [S.l.: s.n.], 2019. Citado 5 vezes nas páginas 24, 41, 53, 55, and 59.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na página 39.

SRIVASTAVA, A. et al. Veegan: Reducing mode collapse in gans using implicit variational

learning. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. [S.l.: s.n.], 2017. Citado na página 33.

TAN, M.; PANG, R.; LE, Q. V. Efficientdet: Scalable and efficient object detection. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2020. Citado 3 vezes nas páginas 23, 36, and 49.

THANH-TUNG, H.; TRAN, T.; VENKATESH, S. Improving generalization and stability of generative adversarial networks. *arXiv preprint arXiv:1902.03984*, 2019. Citado na página 33.

UIJLINGS, J. R. et al. Selective search for object recognition. *International Journal of Computer Vision (IJCV)*, Springer, 2013. Citado 3 vezes nas páginas 13, 34, and 35.

ZHU, J.-Y. et al. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In: *International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2017. Citado 5 vezes nas páginas 14, 37, 38, 44, and 50.

ZHU, X. et al. Adapting object detectors via selective cross-domain alignment. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2019. Citado 5 vezes nas páginas 24, 41, 53, 55, and 59.

Appendix

APPENDIX A – Supplementary Material

A.1 Results

A.1.1 Learning from synthetic data

In Table 7 and Figure 22 are shown additional results and translation samples from Sim10k to Cityscapes. In general, the addition of source data in the training process did not increase the mAP significantly.

Method	Real S	Real T	Fake C	Fake A	car AP
Source-only	✓				34.7
DA-Faster (CHEN et al., 2018)	✓	✗			27.3
Strong-Weak DA (SAITO et al., 2019)	✓	✗			31.7
OURS-C			✓		42.4
OURS-A				✓	43.7
OURS-C+A			✓	✓	**44.6**
OURS-S+C	✓		✓		42.6
OURS-S+A	✓			✓	44.5
OURS-S+C+A	✓		✓	✓	**44.6**
Target-only		✓			58.9

Table 7 – Sim10k→Cityscapes: Results from training with annotated Sim10k (S) and nonannotated Cityscapes (T), evaluating on Cityscapes. The check-mark denotes which data was used during training, and if crossed, only non-annotated data was used. Fake data was acquired by translating Sim10k using CycleGAN (C) and AST-AdaIN (A).

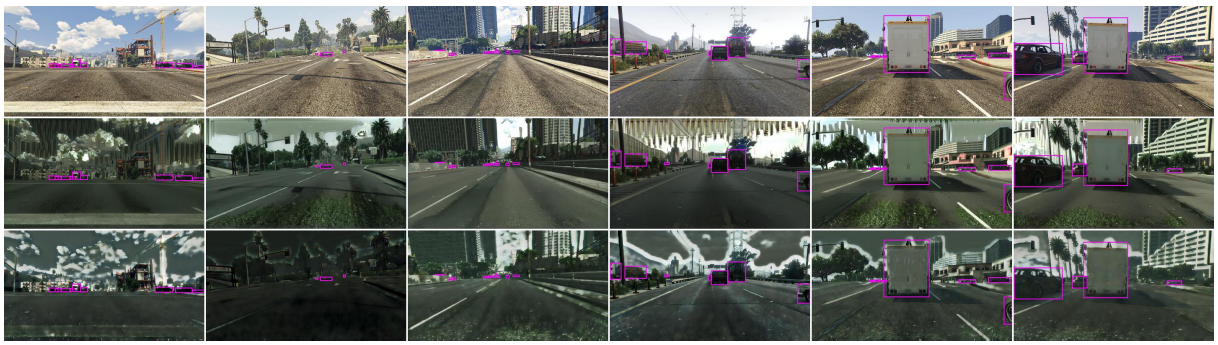

Figure 22 – Samples of translated images with their respective bounding boxes. The real source training images are shown in the first row and their respective fake-target versions are shown in the second and third row for the CycleGAN and AST-AdaIN models, respectively.

A.1.2 Driving in adverse weather

In Tables 8 and 9 are shown additional and detailed results from the Cityscapes to Foggy-Cityscapes datasets. Figure 23 depicts more translation samples.

Method	Real S	Real T	Fake C	Fake A	AP person	rider	car	truck	bus	train	mcycle	bicycle	mAP
Source-only	✓				28.9	40.5	36.1	18.1	27.5	10.0	22.8	32.9	27.1
DA-Faster (CHEN et al., 2018)	✓	✘			28.5	42.0	42.6	21.8	43.9	28.9	22.0	35.2	33.1
Strong-Weak DA (SAITO et al., 2019)	✓	✘			29.2	41.0	44.9	24.6	37.6	21.3	28.1	35.9	32.8
OURS-C			✓		32.7	44.0	52.4	**31.2**	51.5	39.9	28.6	34.5	39.4
OURS-A				✓	27.9	39.9	44.5	21.1	38.5	20.5	23.2	30.6	30.8
OURS-C+A			✓	✓	33.1	45.4	51.8	29.2	48.4	41.4	29.9	35.8	39.5
OURS-S+C	✓		✓		**33.5**	44.3	**53.0**	30.0	**53.0**	**51.6**	**30.4**	36.3	**41.5**
OURS-S+A	✓			✓	31.1	44.5	47.4	24.2	40.2	35.0	29.7	33.9	35.7
OURS-S+C+A	✓		✓	✓	34.4	**45.6**	52.9	30.1	50.1	48.8	30.3	**37.2**	41.2
Target-only		✓			35.9	47.6	56.0	35.6	57.5	50.9	36.0	39.1	44.8

Table 8 – Cityscapes→FoggyCityscapes: Results from training with annotated Cityscapes (S) and non-annotated Foggy Cityscapes (T), evaluating solely on Foggy Cityscapes. The check-mark denotes which data was used during training, and if crossed, only non-annotated data was used. Fake data was acquired by translating Cityscapes using CycleGAN (C) and AST-AdaIN (A).

Method	Real S	Real T	Fake C	Fake A	AP person	rider	car	truck	bus	train	mcycle	bicycle	mAP
Source-only	✓				33.6	45.0	47.1	26.5	44.8	26.1	31.1	36.5	36.3
DA-Faster (CHEN et al., 2018)	✓	✘			31.1	44.3	47.7	28.0	51.3	33.9	28.9	36.5	37.7
Strong-Weak DA (SAITO et al., 2019)	✓	✘			33.4	45.3	50.6	29.9	49.4	32.5	33.2	39.2	39.2
OURS-C			✓		33.0	43.2	53.8	31.7	53.4	36.3	31.4	34.3	39.6
OURS-A				✓	31.5	42.8	50.4	28.3	48.9	27.2	26.2	33.5	36.1
OURS-C+A			✓	✓	34.3	46.3	54.7	33.0	53.1	39.7	33.2	37.6	41.5
OURS-S+C	✓		✓		35.9	47.1	**56.3**	34.7	**58.1**	50.5	**36.1**	38.5	44.6
OURS-S+A	✓			✓	34.5	47.1	52.7	32.9	51.1	37.0	34.7	36.7	40.8
OURS-S+C+A	✓		✓	✓	**36.1**	**48.0**	55.6	**35.6**	57.7	**52.5**	35.2	**38.8**	**44.9**
Target-only	✓	✓			37.5	50.1	57.5	35.9	59.1	48.7	39.7	41.8	46.3

Table 9 – Cityscapes→FoggyCityscapes: Results from training with annotated Cityscapes (S) and non-annotated Foggy Cityscapes (T), evaluating on both source and target dataset. The check-mark denotes which data was used during training, and if crossed, only non-annotated data was used. Fake data was acquired by translating Cityscapes using CycleGAN (C) and AST-AdaIN (A).

A.1.3 Cross-camera adaptation

Full results are shown in Table 10 and additional translation samples are shown in Figure 24.

To support the use of the latest metric version (Pascal VOC 2012), we evaluated the scenario using the fake data from CycleGAN on the target domain using the previous metric version (Pascal VOC 2007), achieving 40.3% in mAP (in contrast to 39.4%). For the state-of-the-art works, evaluating on the same domain with the previous metric version, the mAP for DA-Faster and Strong-Weak DA was 27.6% and 34.3% (in contrast to 33.1% and 32.8%), respectively.

Figure 23 – Samples of translated images with their respective bounding boxes. The real source training images are shown in the first row and their respective fake-target versions are shown in the second and third row for the CycleGAN and AST-AdaIN models, respectively.

Method	Real		Fake		car AP (C→ K)				car AP (K→ C)			
					Protocol 1		Protocol 2		Protocol 1		Protocol 2	
	S	T	C	A	T	S+T	T	S+T	T	S+T	T	S+T
Source-only	✓				74.3	71.9	74.1	68.2	29.1	80.0	28.6	65.2
DA-Faster (CHEN et al., 2018)	✓	✘			71.3	68.8	70.5	64.5	31.8	81.2	30.6	66.5
Strong-Weak DA (SAITO et al., 2019)	✓	✘			64.3	63.2	70.1	64.9	36.1	**82.1**	35.0	**68.8**
OURS-C			✓		75.0	72.0	74.8	67.1	34.9	77.0	36.0	66.2
OURS-A				✓	72.3	69.3	71.4	64.0	**38.7**	78.5	36.8	66.1
OURS-C+A			✓	✓	75.9	73.0	**76.5**	69.0	34.5	79.6	35.6	67.4
OURS-S+C	✓		✓		76.1	73.3	75.9	69.4	36.3	81.3	**36.9**	68.5
OURS-S+A	✓			✓	75.4	72.9	74.9	68.8	35.1	80.6	35.7	68.0
OURS-S+C+A	✓		✓	✓	**76.3**	**73.7**	**76.5**	**70.1**	36.7	80.8	35.5	68.3
Target-only		✓			88.2	-	84.9	-	58.9	-	58.9	-
	✓	✓			-	86.1	-	78.3	-	86.1	-	78.3

Table 10 – Cityscapes→KITTI (C→K): Results from training with annotated Cityscapes (S) and non-annotated KITTI (T), evaluating solely on KITTI (T) or on both (S+T). KITTI→Cityscapes (K→C): Results from training with annotated KITTI (S) and nonannotated Cityscapes (T), evaluating solely on Cityscapes (T) or on both (S+T). The checkmark denotes which data was used during training, and if crossed, only non-annotated data was used. Fake data was acquired by translating the source training set using CycleGAN (C) and AST-AdaIN (A).

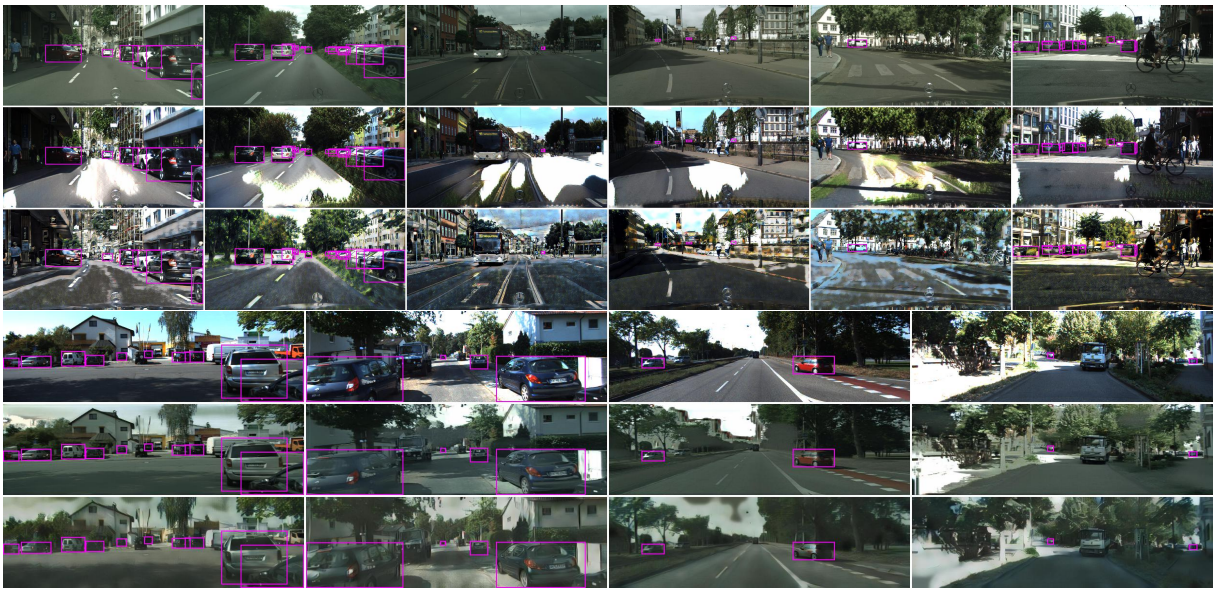

Figure 24 – Samples of translated images with their respective bounding boxes. Top: the translation from Cityscapes to KITTI, and vice-versa for the bottom one. The real source training images are shown in the first row and their respective fake-target versions are shown in the second and third row for the CycleGAN and AST-AdaIN models, respectively.