EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät

Lernbasierte Computer Vision

Bachelorarbeit

A Comparison of Synthetic-to-Real Domain Adaptation Techniques

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Peter Trost, peter.trost@student.uni-tuebingen.de, 2019

Bearbeitungszeitraum: 24.05.2019-23.09.2019

Betreuer/Gutachter: Prof. Dr. Andreas Geiger, Universität Tübingen
Despoina Paschalidou, ETH Zürich
Dr. Yiyi Liao, Max-Planck-Institut für Intelligente Systeme Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Peter Trost (Matrikelnummer 4039682), August 16, 2019

Abstract

With autonomous driving being very popular and gaining a lot of publicity these days there is lots of money and research going into the topic of Artificial Neural Networks that decide how to drive these autonomous vehicles. Currently Deep Neural Networks are the go to architecture used for this task. These Networks need large amounts of data to learn how to properly classify objects in the driving environment so the system can make the right decisions. Currently the best results in training are achieved through supervised training, i.e. having the expected output (ground truth) for each given input. Creating datasets for autonomous driving applications that contain real world images and accurate ground truth is time intensive and therefore very costly. In contrast there are many approaches that can cheaply generate loads of synthetic images from virtual worlds with accurate pixel-level labeling. Models that are trained on these synthetic datasets perform worse in the real world due to the domain shift (changes in texture, lighting,...) between the synthetic and real domain. The research area of Domain Adaptation tries to adapt these two domains so that the difference between them is reduced in order to make it possible to train Deep Neural Networks on cheap synthetic images and still able to perform well in the real world. There are many approaches to Domain Adaptation while one is currently standing out and getting a lot of focus by the research community: Generative Adversarial Nets [GPAM+14]. This work shows three alterations of this network architecture, namely CycleGAN [ZPIE17], CyCADA [HTP+17] and SG-GAN [LLJX18] and compares them on the synthetic-to-real domain shift using GTA5 [RVRK16] and Cityscapes datasets [COR+16].

Acknowledgments

I want to thank everyone that helped me create this work. Especially Anastasia Bitter for her emotional and active support, Stephan Richter for explaining some aspects of the GTA dataset to me and Taesung Park for sending me a pretrained CycleGAN model. Further I want to thank my mentors Despoina Paschalidou and Yiyi Liao for their guidance and support and Professor Andreas Geiger for enabling me to work on this topic and the help in defining it precisely.

Contents

1. Introduction

With autonomous driving being highly popular these days and the need for large amounts of labeled data to train the deep neural networks running the autonomous cars, methods have been developed to generate that data. Real datasets like Cityscapes [COR+16] or Berkley Deep Drive [YXC+18] are expensive to assemble and even more expensive to label. Especially pixel-level annotations require a lot of working hours and can be inaccurate. In contrast synthetic approaches like the GTA5 [RVRK16] or SYNTHIA [RSM+16] datasets enable to create vast amounts of image data automatically and make the annotation process much faster and more accurate and therefore cheaper than their real counterparts. The drawback of using synthetic data to train models that are ran in autonomous cars is that these models don't perform well in real environments. This can result in wrong predictions, which must be avoided in order to make autonomous driving possible. The drop in performance stems from the domain shift, i.e. changes in appearance, texture and lighting of objects in synthetic images compared to objects in the real world seen through the car's cameras. An approach to create models to perform better in this scenario is domain adaptation. The idea behind domain adaptation is to adapt a specific source domain to another target domain (e.g. synthetic to real, image taken at night to image taken at daytime, images of a specific artistic style to another artistic style, ...). For autonomous driving this means making the synthetic image look more like an image of a real scene taken with a real camera. There are many techniques for adapting images from a synthetic to a realistic domain. Each having different approaches and using different methods. This work compares three current approaches on synthetic-to-real domain adaptation, namely CycleGAN [ZPIE17], CyCADA [HTP+17] and SG-GAN [LLJX18], all three of which are based on Generative Adversarial Networks as first proposed in [GPAM+14]. And tries to show their strengths and weaknesses when using them to translate images from the GTA5 dataset, a large scale synthetic dataset of images from a street view in the game Grand Theft Auto V [RVRK16], to images looking like the ones in Cityscapes, a large scale dataset of real street view images of european cities [COR+16]. In order to evaluate the resulting images, a DeepLabv3 [CPSA17] model pre-trained on the Cityscapes dataset (code repository: [GK]) is used to predict semantic segmentation maps. The Cityscapes evaluation code [eab] then yields Intersection over Union (IoU) results for these, comparing predicted pixels with ground truth pixels.

2. Foundations

This chapter will show and explain the foundations necessary for this work. The term **Domain Adaptation** aswell as what **synthetic** and **real** is referred to in the context of this work, will be defined and some examples shown. Furthermore the relevant Neural Network architecture **Generative Adversarial Networks** (GAN) will be described in detail.

2.1. Domain Adaptation

Collecting labeled data in the real world is expensive aswell as potentially inaccurate due to pixel level annotations being challenging for human annotators to do in a very detailed way. This is in contrast to synthetic data for which data collection and annotation can be highly automated. Examples for such synthetically generated datasets are the GTA dataset [RVRK16] and the SYNTHIA dataset [RSM+16]. The synthetic domain is different than the real domain (e.g. changes in texture, lighting, ...) though, which makes models trained on the synthetic data perform worse when applied in the real world. The research field of Domain Adaptation tackles this problem. As described in [Csu17], Domain Adaptation is the task of transfering a machine learning model that is working well on a source data distribution to a related target data distribution. This work focuses on the adaptation from synthetic-to-real images. Where synthetic images are images rendered from a virtual scene through a rendering engine like blender's "Cycles" [Pro] or graphics engine like Unity [Tec]. While real images are defined as taken from a real-world scene through some kind of camera. See Figure 2.1 for an example of synthetic and real images. Other examples of domains are an image of a painting in the style of a particular artist, images of an object from different viewpoints. A major challenge in synthetic-to-real Domain Adaptation is that there is generally no paired data (in contrast to works like [IZZE16] that work with domains that allow the use of paired data). Therefore synthetic-to-real Domain Adaptation methods have to be able to work with unpaired data.

Figure 2.1.: Example images for the the two domains relevant for this work. A real image from the Cityscapes dataset [COR+16] (left) and a synthetic image from the GTA5 dataset [RVRK16] (right)

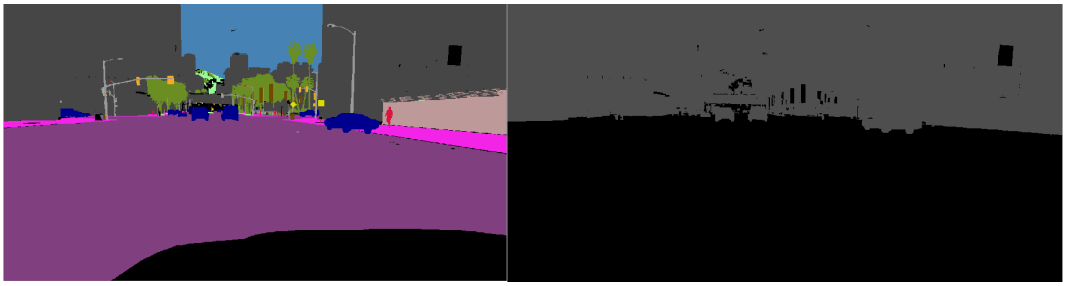

Figure 2.2.: Examples of semantic segmentation images. Image taken from the GTA5 dataset [RVRK16] (left) and the same image mapped to Cityscapes Id values (pixel values 0-19, adapted to brighter values to improve visibility). Streets in purple, sidewalks in pink, pedestrians red, cars blue in the left image which is mainly for visualization purposes. The right image is used for processing.

2.1.1. Example Applications

Semantic Segmentation

Semantic Segmentation is one of the most important aspects in autonomous driving. It is the task of semantically segmentating objects in an image, i.e. labeling each pixel according to what object it belongs to. See Figure 2.2 for examples. Adapting semantic segmentation models is one application of Domain Adaptation and the one focused on in this work.

Classification

Classification models or data that is fed into a classification model can also be adapted through Domain Adaptation. For example when adapting a model trained on classification of objects in pictures taken from a frontal view of that object to pictures taken from a side view (e.g. in [SKFD10])

2.2. Generative Adversarial Networks

Generative Adversarial Networks (GANs) implement a two-player-game:
A **Discriminator** learns from a given data distribution what is "real". The **Generator** generates data. The goal of the generator is to fool the discriminator into believing the generated data is "real" i.e. tries to create samples coming from the same distribution as the "real" data. The discriminator will label anything as "fake" that doesn't resemble the learned "real" data distribution. This can be described as a 2 classes classification, with classes real and fake or 1 and 0. This way GANs can learn to generate realistically looking images of faces, translate images of art from one style to another and improve semantic segmentation. The generative model generally uses *maximum likelihood estimation*. In [Goo17] it is described in the following way:

> The basic idea of maximum likelihood is to define a model that provides an estimate of probability distribution, parameterized by parameters θ. We then refer to the **likelihood** as the probability that the model assigns to the training data: $\prod_{i=1}^{m} p_{\text{model}}(x^{(i)}; \theta)$

The parameter θ that maximizes the likelihood of the data is better found in log space

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{m} p_{\text{model}}(x^{(i)}; \theta) \tag{2.1}$$

$$= \arg\max_{\theta} \log \prod_{i=1}^{m} p_{\text{model}}(x^{(i)}; \theta) \tag{2.2}$$

$$= \arg\max_{\theta} \sum_{i=1}^{m} \log p_{\text{model}}(x^{(i)}; \theta) \tag{2.3}$$

as the maximum of the function is at the same θ value and the now obtained sum aswell eliminates the possibility of having underflow by multiplying multiple very small probabilities together. Formally GANs are a structured probabilistic model containing latent variables **z** and observed variables **x**. The generator is defined by a function G with input **z** and $\theta^{(G)}$ as parameters, the discriminator by a function D with input **x** and $\theta^{(D)}$ as parameters. The discriminator tries to minimize its cost function $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ while only controlling $\theta^{(D)}$. This is analogous for the generator: he tries to minimize $J^{(G)}(\theta^{(D)}, \theta^{(G)})$ while controlling only $\theta^{(G)}$. In contrast to an optimization problem that has a solution that is the (local) minimum (a point in parameter space where all neighboring points have greater or equal cost), the GAN objective is a game. The solution to a game is a Nash equilibrium [Nas50], meaning that each player chooses the best possible option or strategy in respect to what the other player(s) choose. For GANs the Nash equilibrium is a tuple $(\theta^{(D)}, \theta^{(G)})$ that is a local minimum of $J^{(D)}$ with respect to $\theta^{(D)}$ and a local minimum $J^{(G)}$ with respect to $\theta^{(G)}$. The generator is a differentiable function G. When inputting a random noise sample **z** to the generator, G(z) yields a sample of **x** drawn from p_{model}. Typically a

deep neural network is used to represent G.
The game plays out in two scenarios. The first is where the discriminator D is given random training examples $\mathbf{x}$ from the training set. The goal of the discriminator here is for $D(\mathbf{x})$ to be near 1. In the second scenario, random noise $\mathbf{z}$ is input to the generator. The discriminator then receives input $G(\mathbf{z})$, a fake sample by the generator. The discriminator strives to make $D(G(\mathbf{z}))$ approach 0 while the generator tries to make that same value approach 1. The game's Nash equilibrium corresponds to the generators output $G(\mathbf{z})$ being drawn from the same distribution as the training data. Under the assumption that the inputs to the discriminator are half real and half fake, this corresponds to $D(\mathbf{x}) = \frac{1}{2}$ for all $\mathbf{x}$.

Training On each step, two minibatches are sampled: a minibatch of $\mathbf{x}$ values from the dataset and one of random noise $\mathbf{z}$. Then two gradient steps are made simultaneously (simultaneous stochastic gradient descent): one updating $\theta^{(D)}$ to reduce $J^{(D)}$ and one updating $\theta^{(G)}$ to reduce $J^{(G)}$.

Loss functions There are many different loss functions that can be used for GANs. See chapter 4 for the ones that the techniques in this work use.

2.2.1. The Generator

2.2.2. The Discriminator

2.2.3. Conditional GANs

Instead of having random noise as input to the Net, conditional GANs get a conditional input that determines the output. [IZZE16] generated images of shoes depending on drawn sketches thereof, translated grayscale images to color images and more (See Figure 2.3 for examples). This is the same for the methods compared in this work: They use a synthetic image as conditional input and generate an image that looks more realistic while preserving features and image elements of the original image.

2.2.4. Training

There are many different approaches on training GANs. Training details of the Techniques that this work focuses on can be found in Section 4.

2.2.5. Advantages and Disadvantages

Disadvantages

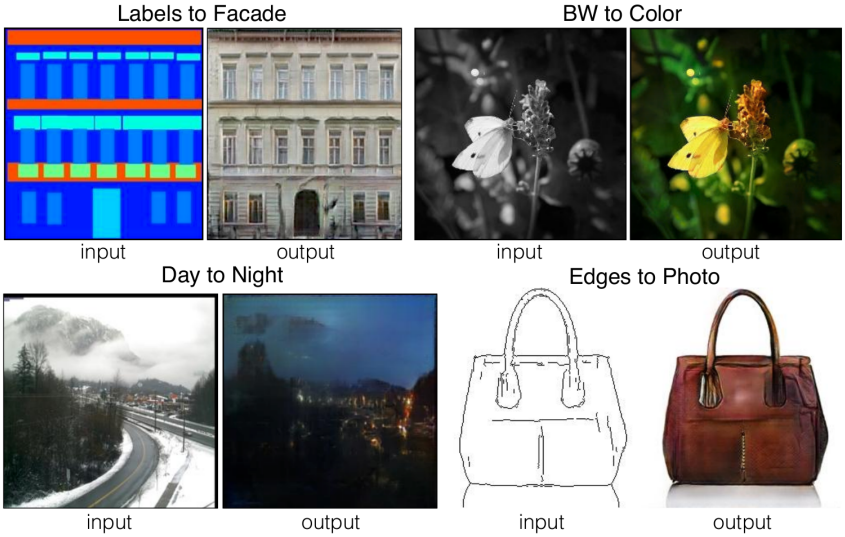

Figure 2.3.: Examples of conditional inputs and generated images (output) of conditional GANs as described in [IZZE16]

non-convergence. Mentioned in [Goo17] as "The largest problem facing GANs [...]": When training GANs, while updating one player so that he makes the best possible current move and goes downhill the loss curve it is possible that the counterfeit player gets updated into going uphill the loss curve. This is in contrast to optimization problems where one tries to find a (local) minimum of the loss curve for example through stochastic gradient decent (SGD). Because there is only one gradient instead of the two in GAN objectives, the model generally makes reliable downhill progress during training. The result is that GANs often oscillate in practice due to the nature of having two players playing against each other, each trying to achieve the optimal outcome for themselves.

mode collapse occurs when the generator learns to map different inputs to the same output. This results in generated data that is missing diversity. A solution is proposed in [ZPIE17]: Using a batch history of 50 images so that the discriminator also learns the distribution of images of the training data and can therefore make sure that the generator will not generate the same small set of images. Partial mode collapse refers to scenarios in which the generator makes multiple images containing the same color or texture themes, or multiple images containing different views of the same dog. See Figure 2.4 for an example.

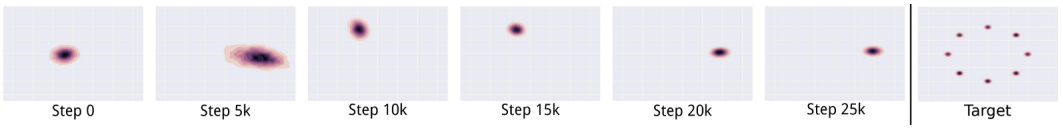

Figure 2.4.: Mode collapse example: Target data distribution of a toy dataset (right) and the data distribution of generated samples. Mind that the generated sample distribution hops from one mode to the next as the discriminator learns to recognize each mode as fake. Image from [MPPS16]

Advantages (As described in [Ahi19])

unsupervised. GANs are trained in an unsupervised manner. There is no ground truth (i.e. labeled data) or an external true or false feedback (reinforcement learning) necessary to train the network. This makes acquiring data cheaper and easier and therefore training of GANs aswell.

data generation. GANs learn to generate data that is indistinguishable from real data. This is useful for many applications such as generating images that can be used by artists to quickly generate a few samples of an idea, generate text that can then be adjusted to a specific purpose, aswell as generating audio and video.

learn density distribution. GANs learn a density distribution of given data. In contrast to other models that can only learn specific distributions, GANs can learn different diverse and complex data distributions.

discriminator is a classifier. The trained discriminator of GANs is a binary classifier and can be used as such. For example if the discriminator learns that images from real data contain a dog it can be used to classify if an image contains a dog or not.

3. Related Work

Following the initial GAN paper [GPAM+14], many additions and alterations have been made to improve and stabilize the training process of GANs and creating better results. While some approaches use additional objectives like a conditional input to generate specific outputs [IZZE16] on paired data, others use a cycle-consistency loss first proposed in [ZPIE17] making unpaired data possible aswell. This chapter shows some of these improved methods while first putting them into context of other methods of Domain Adaptation.

3.1. Approaches of Domain Adaptation

Domain Adaptation can be subcategorized according to [Csu17] and [WD18]. The following sections specify these subcategories and will give example works each.

3.1.1. Discrepancy based

Discrepancy based techniques try to adapt domains by fine-tuning the deep network with labeled or unlabeled target data. Discrepancy based techniques can further be subdivided in following categories:

Class Criterion

Statistic Criterion

Architecture Criterion

Geometric Criterion

3.1.2. Adversarial based

Generative models

Also known as GANs. Section 3.2 shows a few examples of interesting GAN techniques.

Non-generative models

3.1.3. Reconstruction based

Encoder-Decoder Reconstruction

Adversarial Reconstruction

Includes techniques that use a reconstruction error such as the cycle-consistency loss in CycleGAN (see Chapter 4 for more details).

3.2. Domain Adaptation with Generative Adversarial Networks

3.2.1. CoGAN

[LT16]

3.2.2. cGAN

[IZZE16]

3.2.3. PixelDA

[BSD+16]

4. Domain Adaptation Techniques

This chapter will give an overview over the three **Domain Adaptation Techniques** that will be compared in this work, namely **CycleGAN** [ZPIE17], **CyCADA** [HTP+17] and **SG-GAN** [LLJX18]. It will explain how these techniques work, what kind of Network Architecture they use and how training these Nets was approached. See table 4.1

All of the methods compared in this work are based on Generative Adversarial Networks which were initially proposed in [GPAM+14] and are all designed to be able to train on unpaired data which is necessary for the synthetic-to-real task focused on in this work. See chapter 2 for more context and details.

4.1. CycleGAN

4.1.1. Loss Functions

As proposed in [ZPIE17], CycleGAN adds a cycle-consistency loss which consists of an additional generator/discriminator pair with the idea that translating a previously translated image back to the source domain should yield the original image again. Formally: let $G : X \rightarrow Y$ be the mapping function of the Generator translating an image from the source X to the target domain Y. Now add mapping function $F : Y \rightarrow X$ for the second generator's translation from target back to the source domain. The goal of cycle-consistency is that for any given image $x \in X$ from the source domain, $F(G(x)) \approx x$ holds. This analogously also has to hold for any image $y \in Y$ of the target domain with $G(F(y)) \approx y$. As described in chapter 2, GANs implement a two player game. The loss function of this game for CycleGAN is described as

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[1 - \log D_Y(G(x))] \tag{4.1}$$

	Methods		
	CycleGAN	CyCADA	SG-GAN
cycle-consistency loss	✓	✓	✓
semantic loss	-	✓	(✓)
soft-gradient loss	-	-	✓

Table 4.1.: Comparison of loss functions ofDomain Adaptation Techniques **CycleGAN** [ZPIE17], **CyCADA** [HTP+17] and **SG-GAN** [LLJX18]

with D_Y being the discriminator that classifies samples $y \in Y$ as real or fake. This loss function is analogous for generator F and discriminator D_X. The cycle-consistency loss is defined as

$$\mathcal{L}_{\text{cyc}}(G,F) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1] \tag{4.2}$$

with $\|\cdot\|_1$ being the L1-norm, which is the sum of absolute deviations between x and $F(G(x))$ and y and $G(F(y))$. The full objective for the CycleGAN method is therefore:

$$\mathcal{L}(G,F,D_X,D_Y) = \mathcal{L}_{\text{GAN}}(G,D_Y,X,Y) + \mathcal{L}_{\text{GAN}}(F,D_X,Y,X) + \lambda\mathcal{L}_{\text{cyc}}(G,F) \tag{4.3}$$

with λ controlling how much focus lies on the cycle-consistency. The discriminators' goal is to label the samples generated by the generators as fake while the generator tries to generate samples that the discriminator will label as real. This results in the goal of discriminators to maximize Equation 4.3 and generators to minimize it. This leads to the goal for CycleGAN:

$$G^*, F^* = \arg\min_{G,F}\max_{D_X,D_Y} \mathcal{L}(G,F,D_X,D_Y) \tag{4.4}$$

4.1.2. Implementation

Network Architecture

The authors adopt the network architecture proposed in [JAL16]. The network contains two stride-2 convolutions, several residual blocks as described in [HZRS15], and two fractionally-strided convolutions with stride $\frac{1}{2}$. The authors use 6 blocks for 128×128 images and 9 blocks for 256×256 and higher resolution training images. Instance normalization, as well as a Discriminator with 70×70 PatchGANs is used. The latter aims to classify patches of 70×70 pixels of the images as real or fake. The advantage of this PatchGAN is that it has fewer parameters than full-image Discriminators and can be used on images of arbitrary size in a fully convolutional way.

Training Details

To stabilize the model training procedure the authors replaced the negative log likelihood objective by a least-squares loss as it is more stable during training and yields higher quality results. Formally train generator G to minimize

$$\mathbb{E}_{x\sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$$

and discriminator D to minimize

$$\mathbb{E}_{y\sim p_{\text{data}}(y)}[(D(y) - 1)^2] + \mathbb{E}_{x\sim p_{\text{data}}(x)}[D(G(x))^2]$$

By updating the discriminator using a history of 50 generated images model oscillation is reduced. They set $\lambda = 10$ in Equation 4.3, use the Adam solver with a batch size of 1 and train the networks from scratch using a learning rate of 0.0002 while linearily reducing it to zero between the 100th and the 200th epoch.

4.2. CyCADA

4.2.1. Loss Functions

CyCADA [HTP+17] adds a semantic loss to the CycleGAN approach to enforce semantic consistency (no more translating cats to dogs or drawing trees into the sky). This is achieved by doing semantic segmentation on the source image, then translating that image and doing semantic segmentation on the target image. The loss describes the difference between the two resulting label maps. Formally they begin by learning a source model f_X that does semantic segmentation on the source data. For K-way classification with cross-entropy loss this results in the following loss for the classification with L_X being the source labels:

$$\mathcal{L}_{\text{task}}(f_X, X, L_X) = -\mathbb{E}_{(x,l_X)\sim(X,L_X)} \sum_{k=1}^{K} \mathbb{1}_{[k=l_X]} \log\Big(\sigma(f_X^{(k)}(x))\Big) \tag{4.5}$$

where σ denotes the softmax function.
With equation 4.5 the semantic loss is defined as:

$$\begin{aligned} \mathcal{L}_{\text{sem}}(G, F, X, Y, f_X) = \mathcal{L}_{\text{task}}(f_X, F(Y), p(f_X, Y)) \\ + \mathcal{L}_{\text{task}}(f_X, G(X), p(f_X, X)) \end{aligned} \tag{4.6}$$

with $p(f_X, x)$ being the label predicted by source semantic segmentation model f_X on a sample $x \in X$. Together with Equation 4.3 this results in the full objective of the CyCADA approach:

$$\begin{aligned} &\mathcal{L}_{\text{CyCADA}}(f_Y, X, Y, L_X, G, F, D_X, D_Y) \\ &\quad = \mathcal{L}_{\text{task}}(f_Y, G(X), L_X) \\ &\quad + \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &\quad + \mathcal{L}_{\text{cyc}}(G, F, X, Y) + \mathcal{L}_{\text{sem}}(G, F, X, Y, f_X) \end{aligned} \tag{4.7}$$

And ultimately optimizing for a target model f_Y with:

$$f_Y^* = \arg\min_{f_Y} \min_{F,G} \max_{D_X, D_Y} \mathcal{L}_{\text{CyCADA}}(f_X, X, Y, L_X, G, F, D_X, D_Y) \tag{4.8}$$

4.2.2. Implementation

Network Architecture

Training Details

4.3. SG-GAN

4.3.1. Loss Functions

SG-GAN [LLJX18] further expands the above-mentioned losses by including a gradient-sensitive objective that emphasizes semantic boundaries by regularizing the generator to render distinct color/texture for each semantic region, and a patch-level discriminator that better understands differences in appearance distributions of different semantic regions (e.g. coarse texture of asphalt vs smooth and reflective of a vehicle). The soft gradient-sensitive objective is achieved by convolving gradient filters each upon an image and its semantic labeling. Typically Sobel filters are used:

$$C_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, C_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{4.9}$$

These filters extract vertical (y-direction) and horizontal (x-direction) edges in an image. To get the gradient C at each position, following formula is used:

$$C = \sqrt{C_x^2 + C_y^2} \tag{4.10}$$

With input image v and corresponding semantic labeling s_v, the gradient sensitive loss can be defined as follows:

$$l_{\text{grad}}(v, s_v, G_{V\to R}) = \|(|(|C_i * v| - |C_i * G_{V\to R}(v)|)|) \odot sgn(C_s * s_v)\|_1 \tag{4.11}$$

where C_i is the gradient filter for the image and C_s the one for its semantic labeling map and $G_{V\to R}$ is the generator that maps from Virtual to Real domain. $*$ describes the convolution of the filters over the image, $\odot$ is the element-wise multiplication, $|\cdot|$ is the absolute value- and sgn the sign function. The loss grows, the higher the difference between gradients of image v and image $G_{V\to R}(v)$. The element-wise multiplication with $sgn(C_s * s_v)$ enforces that the loss focuses on semantic boundaries only. For the soft gradient loss it may be believed that v and $G_{V\to R}(v)$ share similar texture within semantic classes and therefore one will have edges where the other does aswell. Define following formula for the soft gradient-sensitive loss in which β controls how much belief is set into the texture similarities:

$$\begin{aligned} l_{s-\text{grad}}(v, s_v, G_{V\to R}, \alpha, \beta) = \|(|(|C_i * v| - |C_i * G_{V\to R}(v)|)| \\ \odot (\alpha \times |sgn(C_s * s_v)| + \beta)\|_1 \\ s.t. \quad \alpha + \beta = 1 \quad \alpha, \beta \geq 0 \end{aligned} \tag{4.12}$$

With Equation 4.12 define the full soft gradient-sensitive loss with generator $G_{R\to V}$ from real to virtual and semantic labeling for real images R as S_R and S_V for virtual images V:

$$\begin{aligned}\mathcal{L}_{\text{grad}}(G_{V\to R}, G_{R\to V}, V, R, S_V, S_R, \alpha, \beta) = \mathbb{E}_{r\sim p_{\text{data}}(r)}[l_{s-\text{grad}}(r, s_r, G_{R\to V}, \alpha, \beta)] \\ + \mathbb{E}_{v\sim p_{\text{data}}(v)}[l_{s-\text{grad}}(v, s_v, G_{V\to R}, \alpha, \beta)]\end{aligned} \tag{4.13}$$

The final objective with λ_c, λ_g controlling importance of cycle consistency- and soft gradient-sensitive loss relative to adversarial loss and semantic-aware Discriminators SD_R, SD_V:

$$\begin{aligned}\mathcal{L}(G_{V\to R}, G_{R\to V}, SD_V, SD_R) = \mathcal{L}_{\text{GAN}}(G_{V\to R}, SD_R, V, R) \\ + \mathcal{L}_{\text{GAN}}(G_{R\to V}, SD_V, R, V) \\ + \lambda_c \mathcal{L}_{\text{cyc}}(G_{V\to R}, G_{R\to V}, V, R) \\ + \lambda_g \mathcal{L}_{\text{grad}}(G_{V\to R}, G_{R\to V}, V, R, S_V, S_R, \alpha, \beta)\end{aligned} \tag{4.14}$$

Which results in the optimization target:

$$G^*_{V\to R}, G^*_{R\to V} = \arg\min_{\substack{G_{V\to R}\\ G_{R\to V}}} \max_{\substack{SD_R\\ SD_V}} \mathcal{L}(G_{V\to R}, G_{R\to V}, SD_V, SD_R) \tag{4.15}$$

4.3.2. Implementation

The above mentioned semantic-aware discriminator enforces higher-level semantic consistencies. For example compared to the virtual world, real world may have less illumination. Instead of darkening the tone of the whole image it is instead more accurate to keep the sky bright and only darken e.g. the road. To achieve this semantic-aware judgement of realism, the discriminators' last layers' number of filters is transited to the number of semantic classes. Semantic masks are then applied upon these filters in order to make them focus on different semantic classes. Usually the last layer's feature map of the discriminator is a tensor $\mathbf{T}$ with shape $(w,h,1)$ with w being the width and h being the height. The SG-GAN approach changes this to (w,h,s) where s is the number of semantic classes. The semantic labeling of the image is converted to one-hot style and resized to (w,h). This results in a mask $\mathbf{M}$ with shape (w,h,s). Now multiplying $\mathbf{T}$ and $\mathbf{M}$ element-wise will result in each filter within $\mathbf{T}$ only focussing on one particular semantic class. Summing up $\mathbf{T}$ along the last dimension will result in a tensor of shape $(w,h,1)$ which then can be further processed as in the standard discriminator.

Network Architecture

Training Details

Unpaired data. As all of these techniques use a cycle-consistency loss, it is possible to use unpaired datasets which makes data acquisition easier and reduces costs. This results in more training data and therefore makes better models possible.

5. Experiments

In this chapter the three Domain Adaptation Techniques **CycleGAN** [ZPIE17], **CyCADA** [HTP+17] and **SG-GAN** [LLJX18] will be compared by analysing similarities and differences. Furthermore pretrained models of each architecture were used to generate translated images on which a pretrained DeepLabv3 [CPSA17] model then was used to perform semantic segmentation and the results will be compared on their Intersection over Union scores.

5.1. Datasets

5.1.1. Synthetic dataset: Playing for Data: Ground Truth from Computer Games

The GTA5 (Grand Theft Auto V) dataset is proposed in [RVRK16]. It contains 24966 images taken from a street view in the game Grand Theft Auto V by Rockstar Games [Gam]. The images are provided with 1914×1052 pixels and are containing moving cars, objects, pedestrians, bikes, have changing lighting and weather conditions aswell as day and night scenes. For all of these images the authors provide ground-truth semantic label maps that are compatible with the classes of the Cityscapes dataset [COR+16]. Detouring, i.e. injecting a wrapper between the game and the graphics hardware to log function calls and reconstruct the 3D scene is used to create the images. This also enables a faster labeling process as objects in a scene can be assigned an object ID through which assigned labels are propagated to other images containing this same object. Due to being more realistic than other existing synthetic street view datasets (e.g. SYNTHIA [RSM+16]) the GTA5 dataset is very popular for training machine learning models related to autonomous driving and is therefore used in this work. Example images and corresponding label maps are shown in Figure 5.1

5.1.2. Real dataset: The Cityscapes Dataset for Semantic Urban Scene Understanding

The Cityscapes dataset [COR+16] is a large scale dataset containing car dashcam view images from 50 european cities. It includes 30 classes relevant for autonomous driving. The images include scenes in spring, summer and fall seasons and under different weather conditions. There are 5000 images provided together with fine

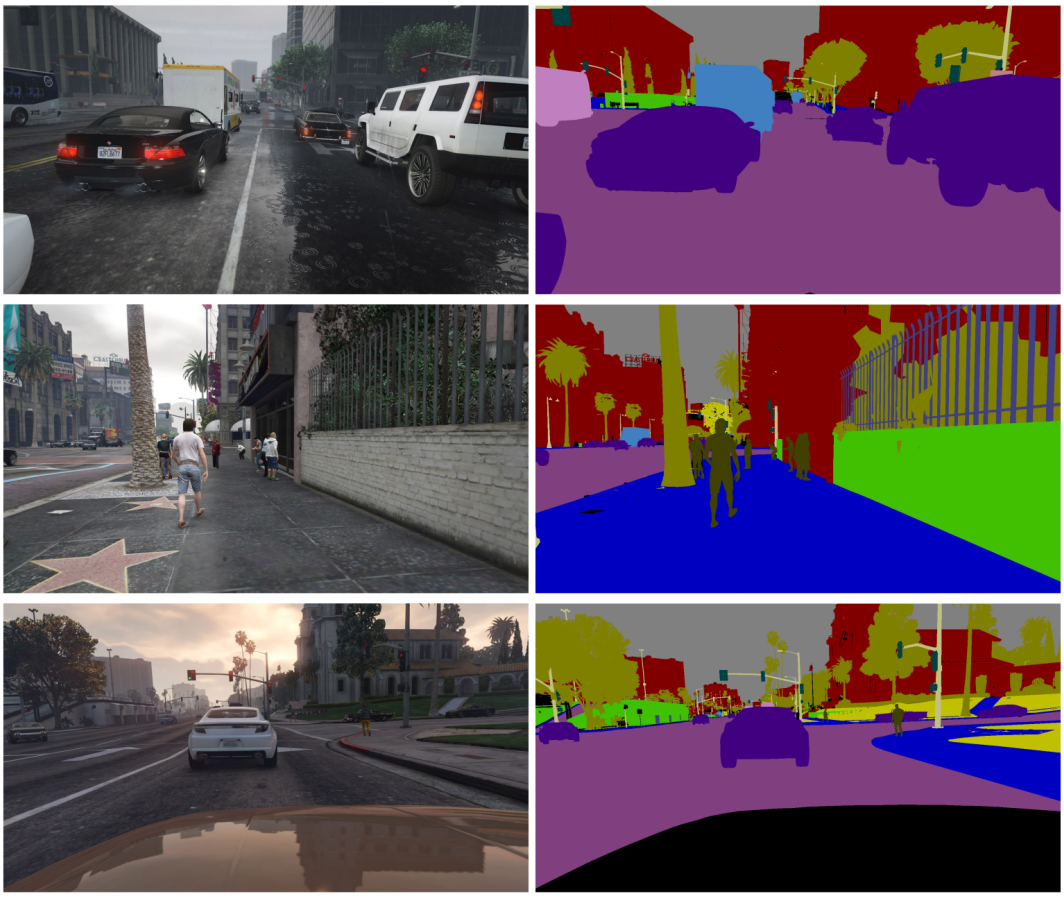

Figure 5.1.: Example images (left) and corresponding ground-truth semantic label maps (right) provided in the GTA5 dataset [RVRK16].

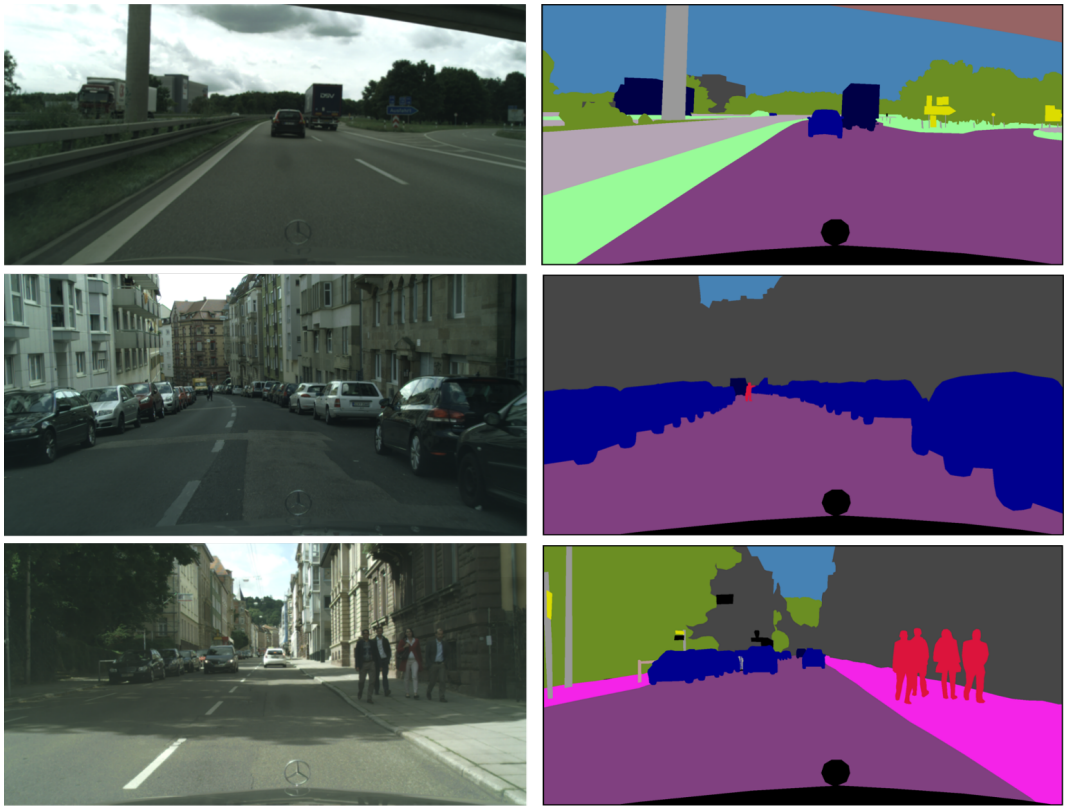

Figure 5.2.: Example images (left) and corresponding ground-truth semantic label maps (right) provided in the Cityscapes dataset [COR+16].

annotations and 20000 together with coarse annotations. Due to the large amount of labeled data from a dashcam view and the inclusion of scenes with different weather and lighting conditions this dataset is often used to train deep neural networks that are related to autonomous driving. Due to the popularity and the GTA5 dataset containing compatible label maps, this work uses Cityscapes as the real dataset for the experiments. See Figure 5.2 for Example images.

5.2. Comparison Benchmark

5.2.1. Semantic Segmentation

As this work is focused on datasets containing driving scenes the methods compared are applied to images that will then be performed semantic segmentation on. For the comparison Intersection over Union (IoU) is used. Intersection over Union is a metric often used to compare semantic segmentation methods' performance. All of the

compared techniques were evaluated using IoU. It follows the following formula:

$$\frac{\text{predicted pixels} \cap \text{ground truth pixels}}{\text{predicted pixels} \cup \text{ground truth pixels}}$$

where predicted pixels are the pixels predicted for a specific class by the semantic segmentation model and ground truth pixels are the pixels containing the ground truth for that image. Usually there are multiple different classes to predict and therefore it is common to calculate the mean IoU (mIoU) over all images that have predictions. To compare how well classes themselves are predicted by a model, one can also calculate the class IoU (cIoU).

5.3. Methodology

For the comparison each technique was used to translate a sample of 500 images from the GTA dataset to the Cityscapes domain. For CycleGAN and SG-GAN each, the authors provided pre-trained models. For CyCADA pretranslated images are provided in the project repository [HX]. To translate images with SG-GAN and CycleGAN the provided code [Li],[eaa], respectively and for the semantic segmentation task an implementation [GK] of DeepLabv3 [CPSA17] was used. To compute the IoU values the DeepLabv3 implementation uses the benchmark code provided by Cityscapes [eab]. All computations were run on a machine with Ubuntu 18.04 using an NVIDIA Geforce GTX 1070 with 8GB RAM. The images were scaled down to 512×256 pixels due to memory limitations while computing the CycleGAN samples.

5.4. Results

5.4.1. Quantitative

As seen in Table 5.1, CyCADA is the only method that improved average values for category and class semantic segmentation of the DeepLabv3 model. CyCADA improved the performance on per category average compared to untranslated GTA data by 2.2% points, CycleGAN decreased it by 2% points and SG-GAN by 4.2% points. While SG-GAN was only able to improve the “nature” category and CycleGAN additionaly improved flat aswell, CyCADA was able to improve 4 of the 7 categories tested. For per class average values, CycleGAN and SG-GAN decreased performance by 2.6% points and 1.1& point, respectively while CyCADA improved it by 1.6% points. For the per class comparison, CycleGAN improved performance for 7 of the 19 tested classes compared to untranslated GTA5. SG-GAN was able to improve the scores for 8 out of 19 classes while having the best values for “bus”, “building”, “traffic sign” and “vegetation”. CyCADA improves performance for 10 classes while performing approximately the same as untranslated GTA images for

"vegetation". It holds the top values for "building", "fence", "motorcycle", "road", "terrain", "train", "truck" and "wall". The biggest improvement for CyCADA is category "rain" where it improves semantic segmentation accuracy by 25.5% points compared to GTA.

5.4.2. Qualitative

See Table 5.3.

5.5. Discussion

category Scores

	Methods			
category	GTA5	CycleGAN	CyCADA	SG-GAN
construction	0.636	0.598	**0.674**	0.628
flat	0.740	0.861	**0.894**	0.735
human	**0.546**	0.402	0.440	0.487
nature	0.543	0.615	**0.622**	0.585
object	**0.085**	0.067	0.073	0.080
sky	**0.872**	0.822	0.832	0.644
vehicle	0.641	0.557	**0.680**	0.609
average	0.580	0.560	**0.602**	0.538

class Scores

	Methods			
class	GTA5	CycleGAN	CyCADA	SG-GAN
bicycle	**0.100**	0.038	0.041	0.051
building	0.620	0.509	**0.634**	0.513
bus	0.209	0.136	0.190	**0.257**
car	0.627	0.570	0.640	**0.642**
fence	0.100	0.102	**0.125**	0.083
motorcycle	0.195	0.125	**0.297**	0.256
person	**0.524**	0.369	0.398	0.461
pole	0.0	0.0	0.0	0.0
rider	**0.312**	0.160	0.144	0.247
road	0.658	0.752	**0.775**	0.631
sidewalk	**0.434**	0.365	0.339	0.381
sky	**0.872**	0.822	0.832	0.644
terrain	0.282	0.361	**0.374**	0.292
traffic light	**0.210**	0.178	0.187	0.185
traffic sign	0.090	0.126	0.120	**0.158**
train	0.025	0.115	**0.280**	0.222
truck	0.387	0.371	**0.511**	0.332
vegetation	0.595	0.620	0.595	**0.653**
wall	0.162	0.186	**0.225**	0.186
average	0.337	0.311	**0.353**	0.326

Table 5.1.: Intersection over Union results for evaluation on untranslated GTA5 images and translated images by CycleGAN, CyCADA and SG-GAN respectively. Rounded to 3 decimal places. Maximum value per row is written in big font.


void
road
sidewalk
building
wall
fence
pole
traffic light
traffic sign
vegetation
terrain
sky
person
rider
car
truck
bus
train
motorcycle
bicycle

Table 5.2.

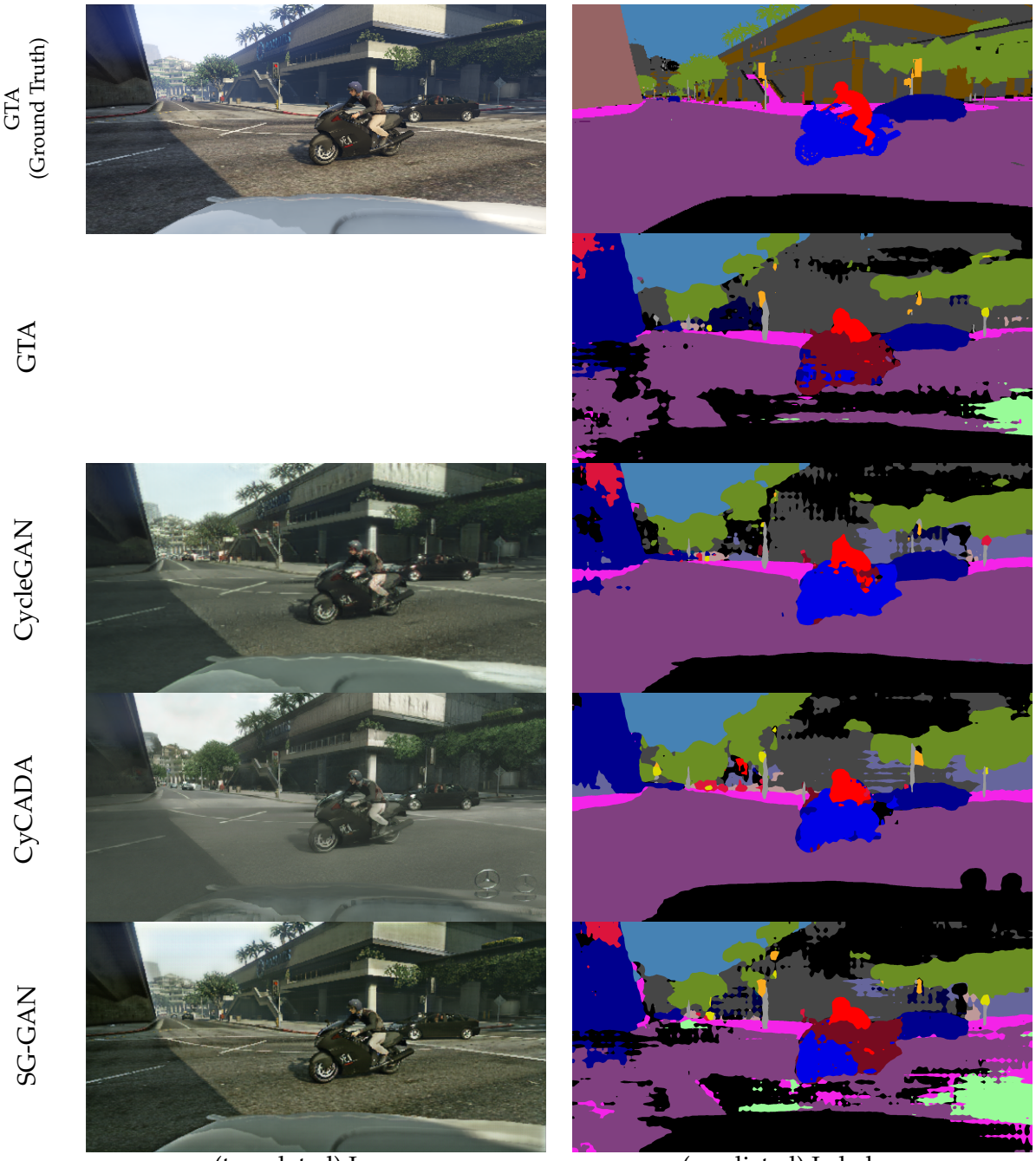
GTA
(Ground Truth)
GTA
CycleGAN
CyCADA
SG-GAN
(translated) Image
(predicted) Labelmap

Table 5.3.

6. Conclusion

6.1. Outlook and Future Work

A. Blub

Bibliography

[Ahi19] Kailash Ahirwar. *Generative Adversarial Networks Projects*. Packt, 2019.

[BSD+16] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CoRR*, abs/1612.05424, 2016.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[CPSA17] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[Csu17] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *CoRR*, abs/1702.05374, 2017.

[eaa] Jun-Yan Zhu et al. CycleGAN lua repository. `https://github.com/junyanz/CycleGAN`. Accessed: 2019-01-08.

[eab] Marius Cordts et al. Cityscapes repository. `https://github.com/mcordts/cityscapesScripts`. Accessed: 2019-26-07.

[Gam] Rockstar Games. Grand Theft Auto V website. `https://www.rockstargames.com/V/`. Accessed: 2019-26-07.

[GK] Fredrik Gustafsson and Erjan Kalybek. Deeplabv3 repository. `https://github.com/fregu856/deeplabv3`. Accessed: 2019-26-07.

[Goo17] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[HTP+17] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *CoRR*, abs/1711.03213, 2017.

[HX] Judy Hoffman and Xian Xu. CyCADA repository. `https://github.com/jhoffman/cycada_release`. Accessed: 2019-01-08.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[IZZE16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[JAL16] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[Li] Peilun Li. SG-GAN repository. `https://github.com/Peilun-Li/SG-GAN`. Accessed: 2019-01-08.

[LLJX18] Peilun Li, Xiaodan Liang, Daoyuan Jia, and Eric P. Xing. Semantic-aware grad-gan for virtual-to-real urban scene adaption. *CoRR*, abs/1801.01726, 2018.

[LT16] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. *CoRR*, abs/1606.07536, 2016.

[MPPS16] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *CoRR*, abs/1611.02163, 2016.

[Nas50] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950.

[Pro] Blender Project. Cycles rendering engine. `https://www.cycles-renderer.org/`. Accessed: 2019-10-07.

[RSM+16] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[RVRK16] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[SKFD10] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In Kostas Daniilidis, Petros

Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 213–226, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Tec] Unity Technologies. Unity graphics engine. `https://unity.com/`. Accessed: 2019-10-07.

[WD18] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *CoRR*, abs/1802.03601, 2018.

[YXC+18] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018.

[ZPIE17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.