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

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 7, 2019

Abstract

With autonomous driving being very popular and gaining a lot of publicity right now 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 gets a lot of focus from the research community: Generative Adversarial Nets [GPAM+14]. This work shows two alterations of this network architecture (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 the 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 GTAV [RVRK16] or SYNTHIA datasets [RSM+16] 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 lighting and others of objects in the 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 domain to another domain (e.g. synthetic to real, image taken at night to image taken at daytime, images of an object taken from one perspective to another perspective, ...). 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/Grad-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 GTAV 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 pre-trained deeplabv3 [CPSA17] model (code: [DLR]) is used to predict semantic segmentation maps and the cityscapes evaluation code [CSR] 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** means in the context of this work, will be defined and some examples shown. Furthermore the relevant **Neural Network** architectures will be shown and explained. Specifically **Convolutional Neural Networks** and **Generative Adversarial Networks**.

2.1. Domain Adaptation

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. In this work we will focus on the adaptation from synthetic to real images. When talking about a synthetic image we are implying it was rendered from a virtual scene through a rendering engine like for example blender's "Cycles" [Cycc] or graphics engine like Unity [Uni]. We define real images 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, and many more.

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)

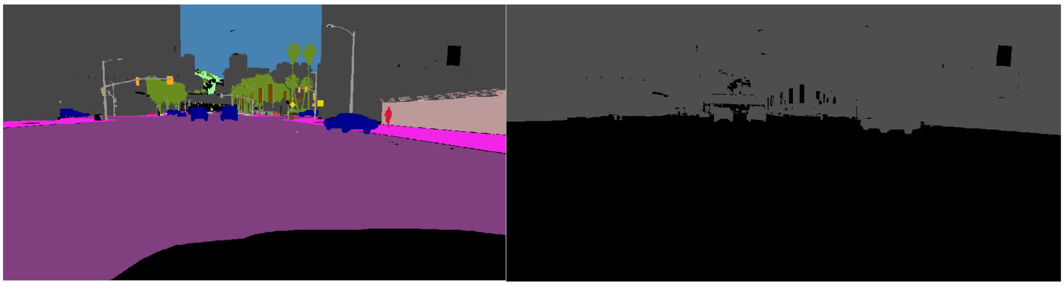

Figure 2.2.: Examples of a semantic segmentation images. Image taken from the GTA5 dataset [RVRK16] (left) and same image mapped to Cityscapes Id values (pixel values 0-19, adapted to brighter values here to improve visibility). Streets in purple, sidewalks in pink, pedestrians red, cars, blue in the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

2.2. 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.

2.3. 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). 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 distribituion, parametereized 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 we now have a sum which aswell eliminates the possibility of having underflow by multiplying multiple very small probabilities together. Formally GANs are a structured probabilistic model (more info: Chapter 16 of Goodfellow et al. 2016) containing latent variables z and observed variables x. The generator is defined by a function G that takes **z** as input and uses $\theta^{(G)}$ as parameters, the discriminator by a function D that takes **x** as input and uses $\theta^{(D)}$ as parameters. Both players have cost functions that are defined in terms of both players' parameters. The discriminator wishes to minimize $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ and must do so by controlling only $\theta^{(D)}$. 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. Here 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 simply a differentiable function G. When **z** is sampled from some simple prior distribution, 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 **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 **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 approach 1. The games Nash equilibrium corresponds to $G(\mathbf{z})$ 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 **x**.

Training On each step, two minibatches are sampled: a minibatch of **x** values from the dataset and one of random noise **z**. Then two gradient steps are made

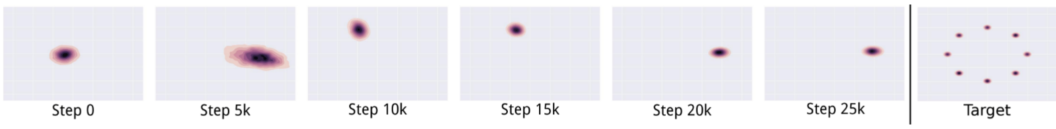

Figure 2.3.: target data distribution of a toy dataset (right) and the data distribution of generated samples. Mind that the generated sample distribution hops from one mode to the next as the discriminator learns to recognize each mode as fake. Image from [MPPS16]

simultaneously (simultaneous SGD): one updating $\theta^{(D)}$ to reduce $J^{(D)}$ and one updating $\theta^{(G)}$ to reduce $J^{(G)}$.

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

2.3.1. Disadvantages

non-convergence 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 2.3 for an example.

2.3.2. Advantages

(From [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 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 for audo and video.

learn density distribution. GANs learn density distribution of data. In contrast to other models that can learn specific distributions, GANs can learn diverse, 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 as to classify if an image contains a dog or not.

3. Related Work

There are many approaches of adapting images from one domain to a different one. After the initial release of the GAN paper [GPAM+14] a lot of research focused on using and improving the GAN architecture and creating Benchmarks to evaluate the diverse approaches. This chapter will show some of these approaches.

3.1. Domain Adaptation with Generative Adversarial Networks

Following the initial GAN paper [GPAM+14], many additions and alterations have been made to improve and stabilize the training process 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.

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.

All of the methods compared in this work are based on Generative Adversarial Networks which were initially proposed in [GPAM+14] and are explained in Section 2. **CycleGAN** [ZPIE17] 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 source domain. The goal of cycle-consistency is that for any 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

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

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 follow:

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

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

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

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 4.2 and generators to minimize it. This leads to the goal for CycleGAN:

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

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 following loss for the classification with L_X being the source labels:

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

With this 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:

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

SG-GAN [LLJX18] expands this 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 and horizontal 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. $*$ describes the convolution of the filters over the image, $\odot$ is the element-wise multiplication, $|\cdot|$ is the absolute value-, sgn the sign function and $\|\cdot\|$ the L1-norm. The loss grows, the higher the difference between gradients of image v and image $G_{V\to R}(v)$ generated by the generator from virtual to real. The element-wise multiplication with $sgn(C_s * s_v)$ enforces that the loss focuses on semantic boundaries only. For the soft gradient loss we may believe 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. We can define following formula for the soft gradient-sensitive loss in which β controls how much we belief in 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 this we can define the full soft gradient-sensitive loss with generators virtual to real and real to virtual $G_{V\to R}$, $G_{R\to V}$, 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:

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

Which results in the optimization target:

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

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. **TODO: maybe add more detail** 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 *Cycle-consistent Generative Adversarial Network* [ZPIE17], *Cycle Consistent Adversarial Domain Adaptation* [HTP+17] and *Semantic-aware Grad-GAN for Virtual-to-Real Urban Scene Adaption* [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 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 [GTA]. 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 label images that are compatible with those 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.

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 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 labeling maps, this work uses Cityscapes as the real dataset for the experiments.

5.2. Comparison Benchmark

5.2.1. Intersection over Union (IoU)

The Intersection over Union is a metric often used to compare semantic segmentation methods. 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 [CyCa]. Due to CyCADA only providing 22k translated images instead of the full 25k images provided in the GTA dataset, from the randomly chosen 500 images only 450 are available here. This has to be regarded in the following comparison. To translate images with SG-GAN and CycleGAN the provided code [SG],[Cycb] and for the semantic segmentation task an implementation [DLR] of deeplabv3 [CPSA17] was used. To compute the IoU values the deeplabv3 implementation uses the benchmark code provided by Cityscapes [CSR]. All computations were run on Ubuntu 18.04 using CPU only in VirtualBox on a Windows Machine due to issues with dependencies while running the code on the tcml cluster of the university of tuebingen [tcm], a cluster for machine learning applications.

5.4. Results

The results are not as expected. As seen in Table 5.1 semantic segmentation on CyCADA-translated images is just approximately 0.9 % points more accurate than untranslated GTA images. SG-GAN is around 10 % points less accurate than CyCADA and GTA5. Translating the images with SG-GAN worsened semantic segmentation results of the Deeplabv3 model.

	Methods		
	GTA5	CyCADA	SG-GAN
category Scores			
construction	**0.740**	0.719	0.651
flat	0.709	**0.894**	0.639
human	**0.522**	0.438	0.475
nature	0.606	**0.617**	0.584
object	**0.100**	0.077	0.085
sky	**0.894**	0.841	0.518
vehicle	0.661	**0.694**	0.641
average	0.604	**0.611**	0.513
class Scores			
bicycle	**0.127**	0.055	0.062
building	**0.732**	0.669	0.608
bus	0.224	**0.345**	0.276
car	0.652	**0.665**	0.620
fence	**0.201**	0.186	0.169
motorcycle	0.304	0.365	**0.383**
person	**0.501**	0.419	0.455
pole	0.0	0.0	0.0
rider	**0.460**	0.314	0.366
road	0.657	**0.781**	0.582
sidewalk	**0.370**	0.314	0.315
sky	**0.894**	0.841	0.518
terrain	**0.304**	0.299	0.260
traffic light	**0.197**	0.174	0.181
traffic sign	**0.156**	0.108	0.143
train	0.013	0.002	**0.024**
truck	0.267	**0.364**	0.286
vegetation	0.657	0.606	**0.663**
wall	0.222	**0.298**	0.260
average	**0.365**	0.358	0.325

Table 5.1.: Intersection over Union results for evaluation on untranslated GTA5 images and translated images by CyCADA and SG-GAN respectively. Rounded to 3 decimal places

6. Conclusion

To conclude...

A. Blub

Bibliography

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

[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.

[CSR] Cityscapes repository. `https://github.com/mcordts/cityscapesScripts`. Accessed: 2019-26-07.

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

[CyCa] CyCADA repository. `https://github.com/jhoffman/cycada_release`. Accessed: 2019-01-08.

[Cycb] CycleGAN lua repository. `https://github.com/junyanz/CycleGAN`. Accessed: 2019-01-08.

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

[DLR] 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.

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

[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.

[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.

[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.

[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.

[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.

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

[tcm] tcml cluster universität tübingen. `https://uni-tuebingen.de/fakultaeten/mathematisch-naturwissenschaftliche-fakultaet/fachbereiche/informatik/lehrstuehle/kognitive-systeme/projects/tcml-cluster/`. Accessed: 2019-01-08.

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

[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.