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), September 18, 2019

Abstract

Due to advances in deep learning and progress in high-performance computing, self driving cars gain increasing popularity. Currently, Deep Neural Networks perform well on tasks such as semantic segmentation and object recognition. These are relevant tasks for the car as it needs to make decisions depending on its environment. Deep Neural Networks need large amounts of data to learn how to properly classify or recognize objects in the driving environment so the system can make the right decisions. Currently, the best results in training are achieved via 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 trained on synthetically generated datasets however do not perform well in the real world due to the domain shift (changes in texture, lighting, etc.). 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 different approaches for Domain Adaptation. Because it generates realistic looking images Generative Adversarial Networks [GPAM+14] has recently gained attention by the research community. 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]. Models of the named techniques pre-trained on the translation from GTA5 to Cityscapes data were used to translate a sampleset of images. Semantic segmentation is then performed on these translated samples. To evaluate the techniques' performances the results of the semantic segmentation is compared to the ground truth of the corresponding images.

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 used for decision-making process in the autonomous cars, various 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 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.

1.1. Domain Adaptation

An approach to create models to perform better in this scenario is domain adaptation. The task of 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, etc.). 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.

1.2. Contributions

This thesis 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]. Furthermore, it tries to showcase the 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.

Conclusions The experiments show that the CyCADA model improves average semantic segmentation performance while SG-GAN and CycleGAN worsens it. The results suggest that in order to successfully perform Synthetic-to-Real Domain Adaptation the models have to be carefully trained.

2. Foundations

In this chapter, we discuss the foundations necessary for this work. The term **Domain Adaptation** as well as the definition of **synthetic** and **real**, as referred to throughout this study, 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

Real-world data labelling can be expensive as well as inaccurate, as pixel level annotations are particularly challenging for human annotators. 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 GTA5 dataset [RVRK16] and the SYNTHIA dataset [RSM+16]. The synthetic domain is different than the real domain (e.g. changes in texture, lighting, etc.) 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 transferring a machine learning model that is working well on a source data distribution to a related target data distribution. This thesis focuses on the adaptation from synthetic to real images for the task of semantic segmentation. 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 2.2 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 this context paired data consists of tuples of images that have some sort of relation. For example, a shoe and an outline thereof, or an RGB image and the same image in greyscale. Paired data is used for example in [IZZE16]. See Figure 2.1 for some examples. Networks trained on paired data can directly learn features of an image from the pair. For unpaired data, the networks have to learn a general understanding of the relation between the domains. Therefore, synthetic-to-real Domain Adaptation methods have to be able to work with unpaired data.

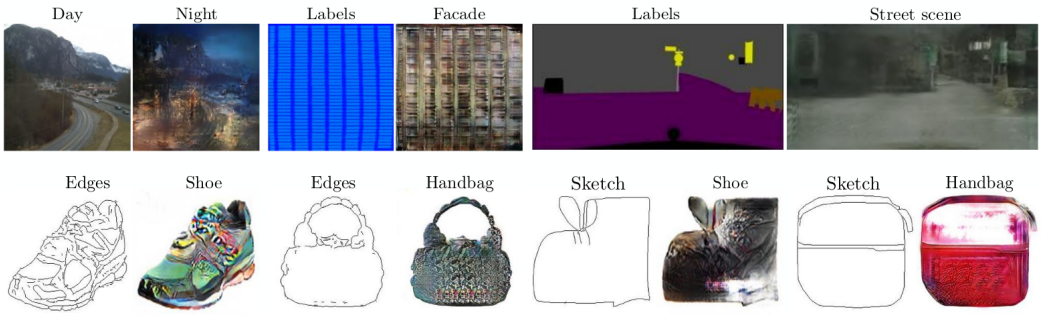

Figure 2.1.: Examples of paired data provided in [IZZE16]

Figure 2.2.: 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)

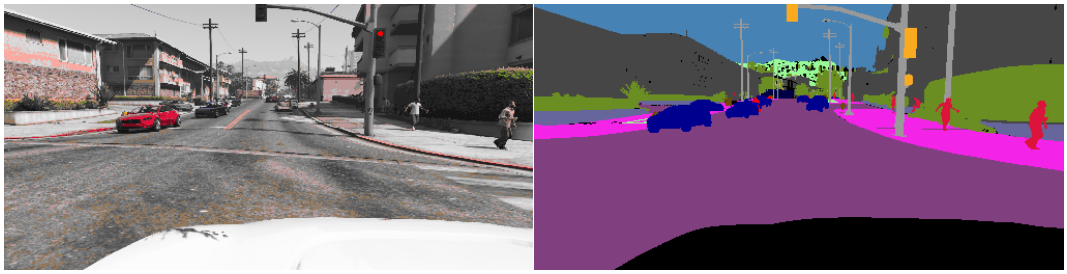

Figure 2.3.: Examples of semantic segmentation. Image taken from the GTA5 dataset [RVRK16] (left) and the corresponding semantic segmentation (right). Streets in purple, sidewalks in pink, pedestrians red, cars blue in the left image which is mainly for visualization purposes. The right image is used for processing.

2.1.1. Example Applications

This section gives two possible applications of Domain Adaptation and a definition thereof.

Semantic Segmentation

Semantic Segmentation is an important task in autonomous driving. According to [SS01], Image segmentation in the context of computer vision is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. See Figure 2.3 for examples. Adapting images from synthetic to real domains is one application of Domain Adaptation and the one focused on in this work.

Classification

Classification is the task of assigning an observed object to the correct category of a set of categories according to a learned categorization. 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 binary classification problem, 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*. Following, the notation introduced in [Goo17], below we formulate the maximum likelihood estimation.

> 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 as well 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 **x** from the training set. The goal of the discriminator

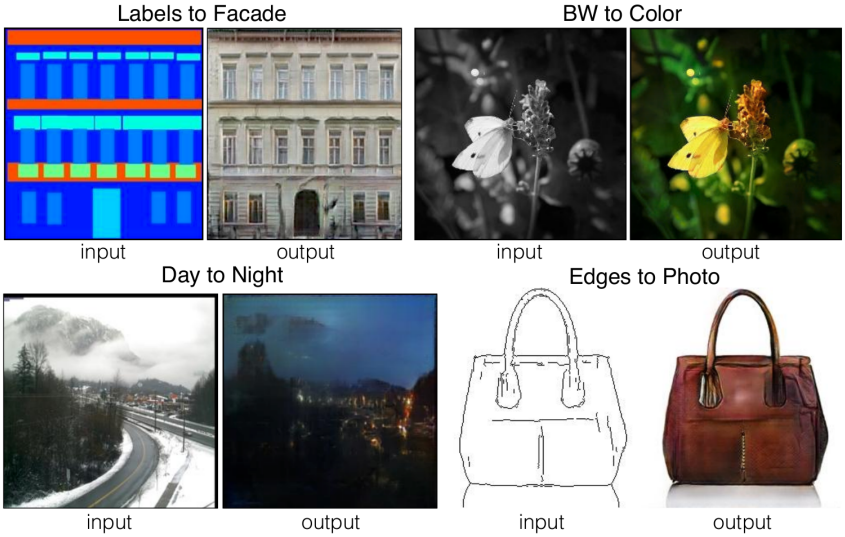

Figure 2.4.: Examples of conditional inputs and generated images (output) of conditional GANs as described in [IZZE16]

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 $J^{(G)}$.

2.2.1. Conditional GANs

In addition to random noise as input to the network, 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 2.4 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.

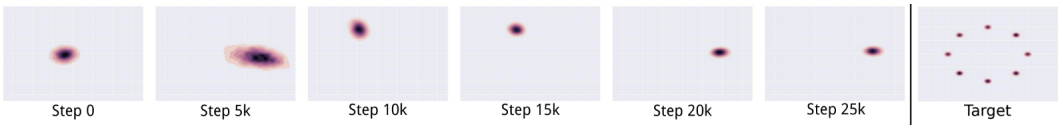

Figure 2.5.: Mode collapse example: Target data distribution of a toy dataset (right) and the data distribution of generated samples. Mind that the generated sample distribution hops from one mode to the next as the discriminator learns to recognize each mode as fake. Image from [MPPS16]

2.2.2. Challenges and Benefits

Challenges

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 2.5 for an example.

Benefits

(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 as well.

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, as well 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.

This thesis focuses on Generative Adversarial Networks because they can generate realistically looking images. Therefore, it is easier for a human to qualitavely evaluate how well a model is able to perform Domain Adaptation compared to Domain Adaptation techniques. They may adapt, for example, through learning a mapping of features from one domain to another. This mapping is stored within the networks structure in the form of weights and biases which makes it harder to qualitatively evaluate the performance of an adaptation.

3. Related Work

There are many different approaches on Domain Adaptation. This chapter will define a categorization thereof according to [Csu17]

3.1. A Categorization of Domain Adaptation techniques as proposed in [Csu17]

Domain Adaptation can be subcategorized and defined according to [Csu17]. The following subsections specify these subcategories, together with definitions as stated in [WD18] and 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. They can further be subdivided in following categories:

Class Criterion. Uses the class label information to transfer knowledge between domains. Approaches that are commonly used when labeled target data is available are soft label and metric learning. An example for this is [THDS15]. The authors accomplish domain adaptation by computing an estimated marginal distribution over the target domain using some labeled target data and optimize the network to minimize the distance between source and target domain distributions. If no labeled target data is available, pseudo labels and attribute representation are possible approaches. [ZYCW15] propose the Deep Transfer Network (DTN) which models and matches both the marginal and the conditional distributions of source and target domains. Pseudo labels are used here in order to compute the distance between conditional distributions of the source and target domain using a conditional Maximum Mean Discrepancy (MMD). An example for synthetic-to-real domain adaptation is [LXG+16]. In this work, authors use pseudo labels and attribute representation (among others) to train a deformable part-based model (DPM) on the synthetic-to-real domain adaptation task between the SYNTHIA [RSM+16] and Cityscapes [COR+16] datasets.

Statistic Criterion. Uses mechanisms like MMD as used in [ZYCW15] and correlation alignment (CORAL) among others to align the statistical distribution shift between the source and target domains. CORAL as proposed in [SFS15] aligns the second-order statistics of source and target data distributions through a linear transformation. [SS16] extend this to use with deep neural networks. The authors construct the CORAL loss "by constructing a differentiable loss function that minimizes the difference between source and target correlations". They show that their proposed technique can be applied to more diverse problems than the original CORAL because it learns a non-linear transformation.

Architecture Criterion. The network is modified in order to make it able to learn more transferable features. A prominent technique is batch normalization (BN) as propsed in [IS15] where a layer is added to the network in order to reduce internal covariate shift. Further techniques include weak-related weight, domain-guided dropout and others.

Geometric Criterion. Assumes that the relationship of geometric structures in source and target domain can reduce the domain shift and bridges these domains according to their geometrical properties. [CBG13] uses this assumption to define a so called "interpolating path" between the source and target domains which they use in order to adapt their model.

3.1.2. Adversarial based

Uses a discriminator as described in Chapter 2 to distinguish between samples from source and target domain. With it an adversarial objective to minimize the distance between empirical source and target mapping distributions is used to encourage domain confusion. Adversarial based techniques can be further subcategorized according to if generative models are used or not.

Generative models. Mainly GANs as described in Chapter 2. The techniques compared in this work are all GAN-based and will be explained in more detail in Chapter 4.

Non-generative models. The feature extractor learns a discriminative representation using the labels of the source domain and maps the target to the same space through a domain-confusion loss, thus resulting in domain-invariant representations. [HWYD16] propose "the first unsupervised domain adaptation method for transferring semantic segmentation FCNs [(Fully Convolutional Networks)] across image domains". The authors techniques maps the source and target domain features

to a common label space using an adversarial learning procedure that learns to distinguish between the two domains in order to guide the distance minimization of the two domains' representations.

3.1.3. Reconstruction based

Assumes that the data reconstruction of source and target data can be helpful for improving the performance of Domain Adaptation. The reconstructor can ensure that inter-domain representations such as high-level features (edges, curves, etc.) and intra-domain representations (texture, lighting, etc.) do not change.

Encoder-Decoder Reconstruction. Encoder-decoder techniques use stacked autoencoders (SAEs) to combine the encoder network for representation learning with a decoder network for data reconstruction. [GKZ+16] propose the Deep Reconstruction-Classification Network (DRCN). This technique learns to predict labels on source domain images in a supervised way and reconstruct target data in an unsupervised way. The goal is to correctly label images in the target domain supported by the auxiliary task of data reconstruction.

Adversarial Reconstruction. Includes techniques that use a reconstruction error such as the cycle-consistency loss in CycleGAN (see Chapter 4 for more details) which measures the distance between the original and reconstructed images within each domain.

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 networks was approached. See table 4.1 for an overview of what losses the techniques use.

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 of Domain 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

The authors use the task loss on labeled source data to pretrain the source task model f_X that performs semantic segmentation on the source images. Then they

combine their image space GAN losses with semantic segmentation consistency and cycle consistency losses to perform pixel-level adaptation of the images. From this the parameters for the translations G and F and the discriminators D_X and D_Y are obtained. Additionaly an initial version of the the task model f_Y trained on translated source images and their corresponding source labels is learned. To conclude training the authors learn a feature discriminator D_{feat} which guides the representation update of f_Y. This is done in order to adapt the feature space of f_Y to obtain features that are aligned between translated source images and real target images. The generator and discriminator losses are weighted equally for all feature space adaptation. If the discriminator's accuracy exceeds 60% over the last 100 iterations the generator is updated to reduce volatile training.

Network Architecture

The authors use VGG16-FCN8s [LSD14] as well as DRN-26 [YKF17] during their experiments.

Training Details

Using the FCN8s architecture the authors train their source semantic segmentation model for 100k iterations using SGD, a learning rate of 0.001 and momentum of 0.9. For DRN-26 they use SGD as well with the same hyper-parameters but for 115k iterations. Both times they crop the images to 600×600 and train on batches of 8 images. They used the original hyper-parameters and network architecture of CycleGAN [ZPIE17] for cycle-consistent image adaptation. The images were scaled to 1024 pixels width and randomly cropped to patches of 400×400 for training. They trained for 20 epochs. The authors used SGD with momentum 0.99 and learning rate 0.00001 for feature level adaptation training.

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 as well. 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

The images are resized to 256×512. The authors adapt the architecture from [IZZE16] for the generator as well as the PatchGAN from the same work for the semantic-aware discriminator.

Training Details

To stabilize training of their semantic aware discriminators SD_R and SD_V, they use a history of refined images as proposed in [SPT+16]. To generate higher quality images and stabilize training further the authors use least square objective for adversarial loss instead of log likelihood as proposed by [MLX+16]. They set $\lambda_c = 10$, $\lambda_g = 5$ and α,β as $(1,0)$ for the first three epochs and change it to $(0.9,0.1)$ after that in equation 4.14. The sobel filters in equation 4.9 are used as gradient filters $\mathbf{C}_i$ in equation 4.12 and

$$C_x = \begin{pmatrix} 0 & 0 & 0\\ -1 & 0 & 1\\ 0 & 0 & 0 \end{pmatrix}, C_y = \begin{pmatrix} 0 & 1 & 0\\ 0 & 0 & 0\\ 0 & -1 & 0 \end{pmatrix} \tag{4.16}$$

for $\mathbf{C}_s$. In order to avoid sparse classes the authors cluster 30 [COR+16] into 8 categories. Training is done with a batch size of 1 and learning rate 0.0002.

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 as well 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

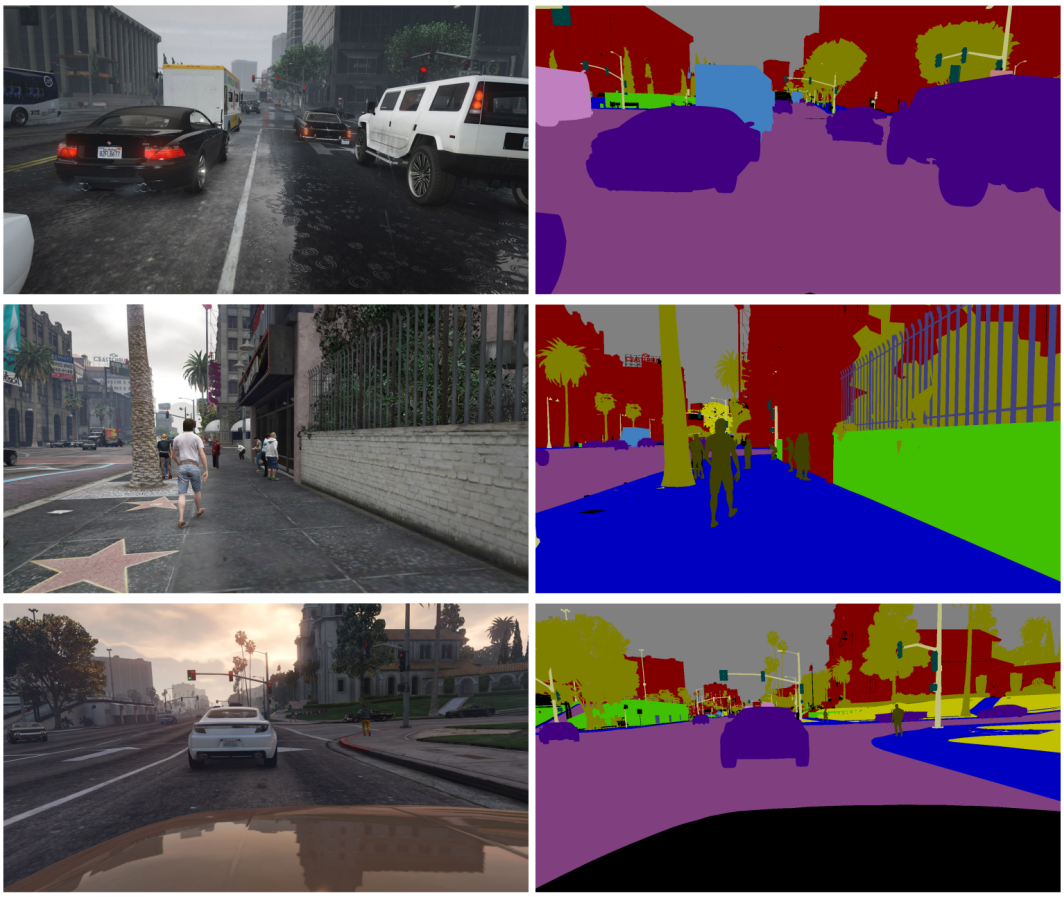

Figure 5.1.: Example images (left) and corresponding ground-truth semantic label maps (right) provided in the GTA5 dataset [RVRK16].

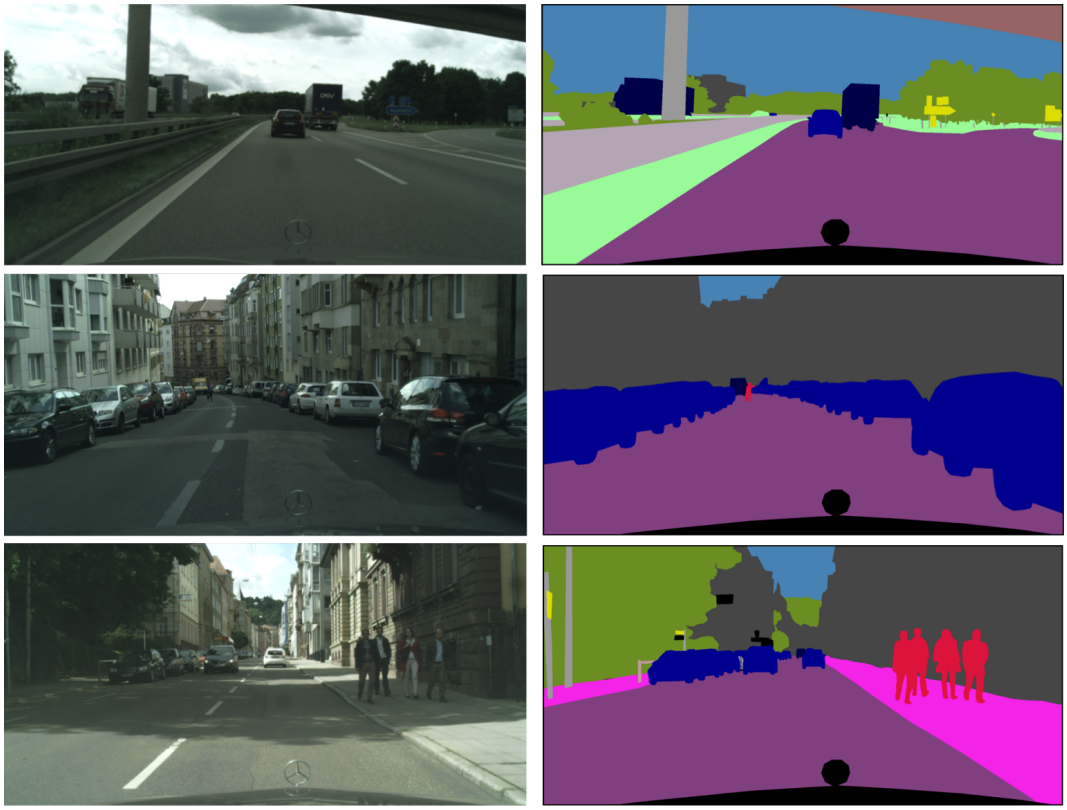

Figure 5.2.: Example images (left) and corresponding ground-truth semantic label maps (right) provided in the Cityscapes dataset [COR+16].

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 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 code provided in the repositories [Li] and [eaa] was used. 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 5.2, 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 as well, 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 "train" where it improves semantic segmentation accuracy by 25.5% points compared to GTA. See Table 5.1 for an example that showcases the superior performance of semantic segmentation on the CyCADA translated image for the train category.

5.4.2. Qualitative

Table 5.3 shows a qualitative comparison for a single frame translated through each method and with corresponding predicted labelmap. One can see that all of the techniques darkened the overall appearance of the image while SG-GAN still has brighter color than CycleGAN and CyCADA. CyCADA smooths out the road the most out of all of the techniques and SG-GAN least. SG-GAN has a lot of noisy structure in the translated images. Also they have a bright glow between the sky and other semantic classes. The only technique that learned the mercedes star and generates it in most of the images is CyCADA. Overall, the SG-GAN images look very sharp and close to the original GTA images while CycleGAN and CyCADA smooth out the textures more. The pedestrians on the right side of the images are more clearly generated in CyCADA than CycleGAN. The predicted labelmap of the ground truth GTA image clearly shows that the DeepLabv3 model trained on Cityscapes performs worse in the synthetic domain. For SG-GAN it predicts large portions of the road as sidewalks and generates noisy label images. The DeepLabv3 model is able to successfully predict streets and sidewalks for CyCADA and CycleGAN images. While it predicts some regions other than the ego car as void for SG-GAN and CycleGAN, CyCADA only has one small region that it cannot predict as belonging to one of the main classes. The CyCADA label map contains more vegetation than the ground truth.

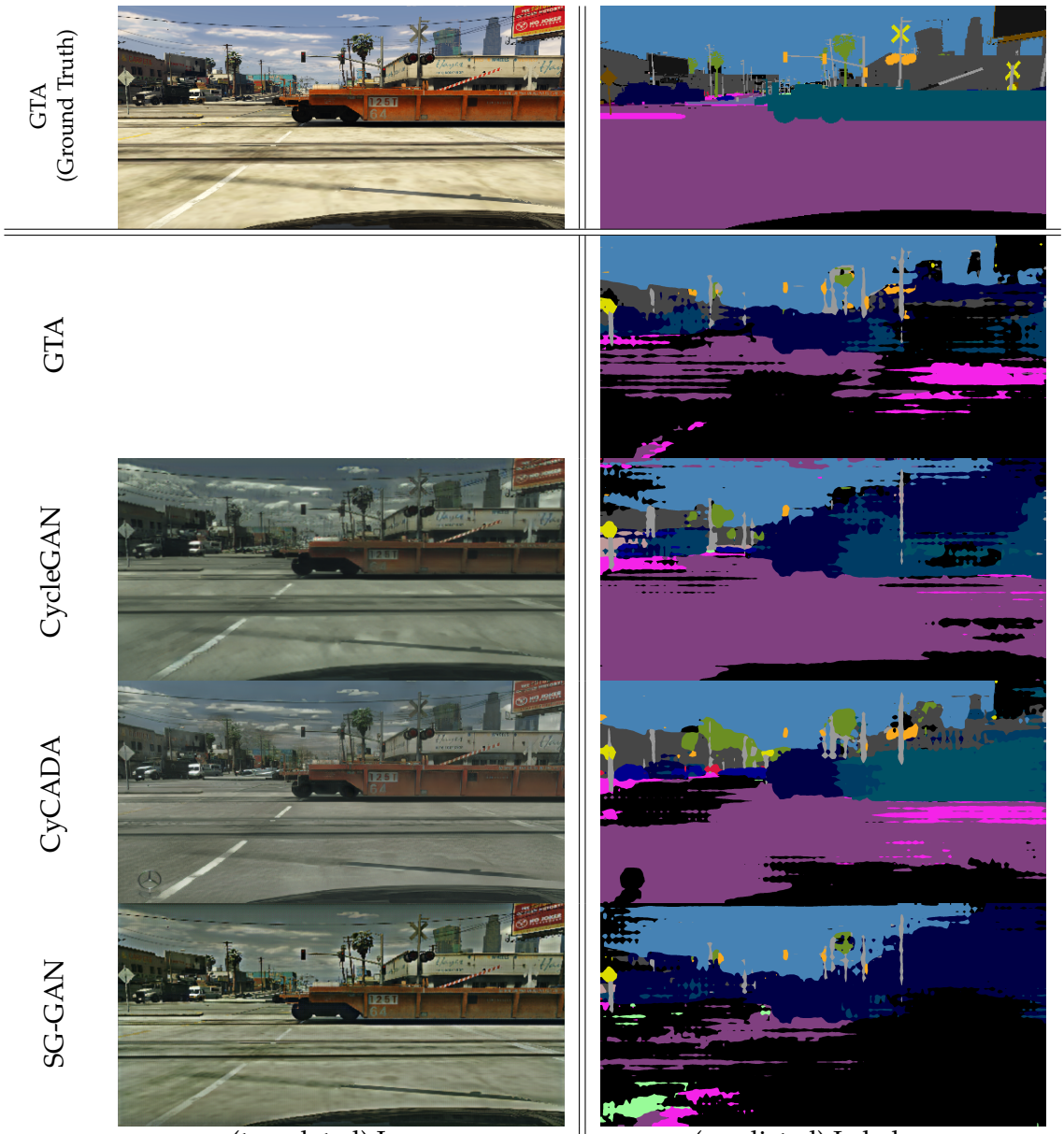

Table 5.1.: An example of an image containing a train. Comparing the images translated by the different methods and the corresponding semantic segmentation. It is easy to see that CyCADA performs best in this comparison on semantic segmentation of the train (colored dark teal).

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 5.2.: 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.

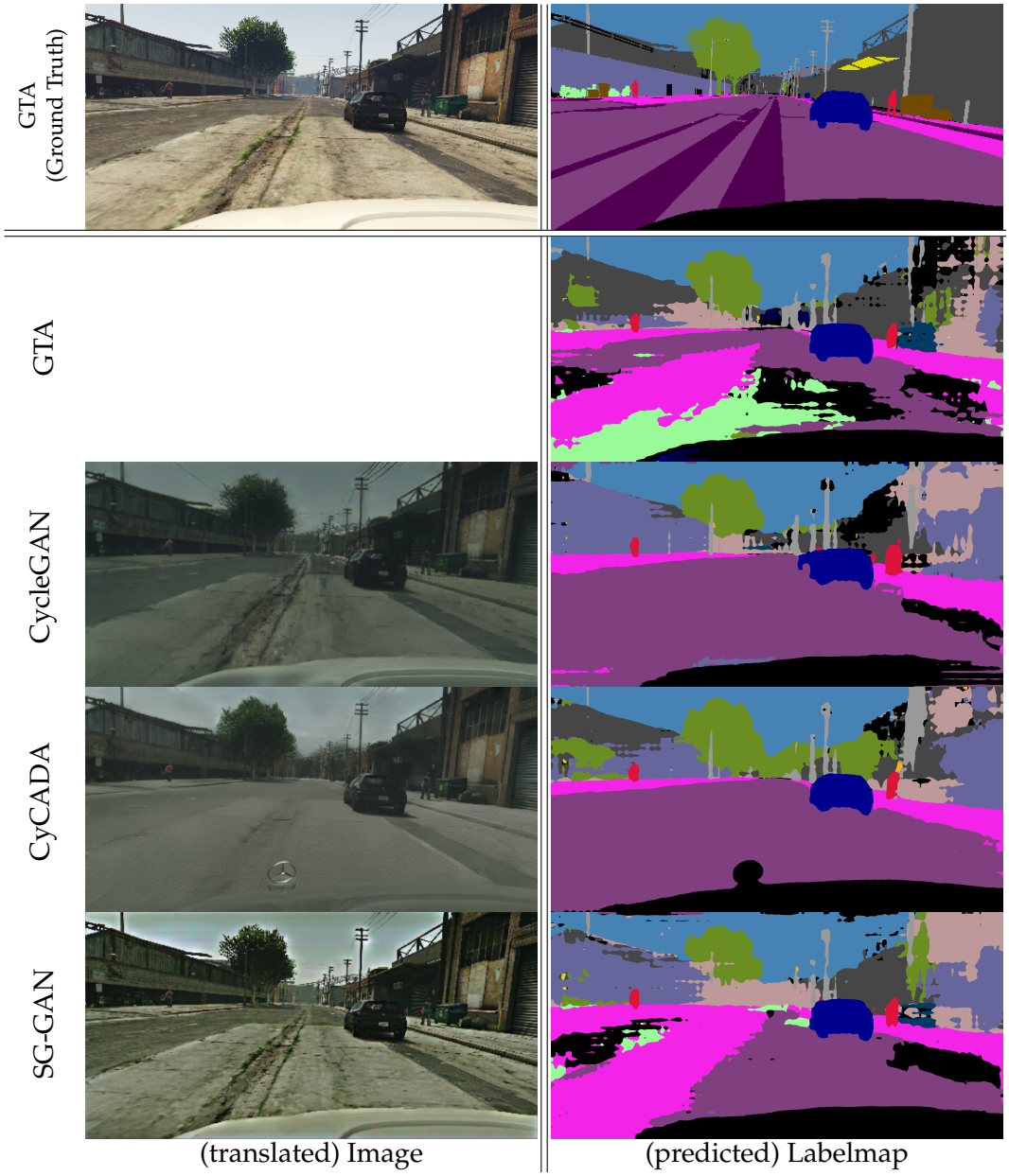

Table 5.3.: Ground Truth image and corresponding Ground Truth labelmap from the GTA dataset (top) and the images translated by the different techniques (left side) with their corresponding predicted labelmaps (right side)

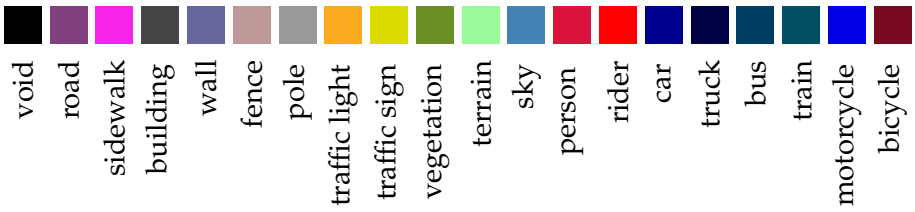

Table 5.4.: Color coding of semantic classes used in semantic segmentation labelmaps.

5.5. Discussion

CyCADA performed best on average for both category and class Scores in this comparison. From the translated images used in this comparison it looks like CyCADA was able to smoothen textures more than the other methods giving them a more realistic look. This is especially obvious when looking at the "flat" category scores where it improves semantic segmentation by as much as 15.4%. The untranslated GTA images having the best scores for human, sky and object categories is probably due to noise and artifacts generated by the adaptation techniques. This results in reduced details for instances of the human category which is generally already smaller (pixel-wise) than the other categories. SG-GAN creates a lot of noise into the sky which makes correct prediction of sky labels a lot harder for the DeepLabv3 model. SG-GAN performing best for cars and busses might be due to generating distinct pixels around semantic boundaries which may result in better distinction between instances of cars and busses far in the background and their surroundings. The "glow" is especially visible between the sky and its neighboring semantic classes. This might be the reason for SG-GAN to perform best for traffic signs as they are easier to seperate from the background. Overall, the results are not as expected. All of the techniques compared in this work show in their corresponding papers that they improve the performance of semantic segmentation models compared to unadapted images for GTA to Cityscapes domain adaptation. This is not the case here. This is probably due to the models used in this work not being the ones the authors actually used in their works. Therefore, it can't be ensured that they were trained the way the authors describe which may result in the decrease in average performance for CycleGAN and SG-GAN.

6. Conclusion

This work gave a foundation on domain adaptation and Generative Adversarial Networks, an overview over domain adaptation techniques and compared three of these techniques (namely CycleGAN [ZPIE17], CyCADA [HTP+17] and SG-GAN [LLJX18]) on their capabilities to adapt synthetic images from the GTA dataset [RVRK16] to real looking images from the Cityscapes dataset [COR+16] domain. This was achieved by using pre-trained models for the techniques to translate a sample set of 500 images from the GTA images to the Cityscapes domain. A pre-trained DeepLabv3 [CPSA17] model was used to perform semantic segmentation on these generated images. The resulting predicted semantic segmentation images were then compared to the corresponding ground truth label maps of the GTA dataset using the code provided in the Cityscapes repository [eab]. This yields the Intersection over Union values for several categories and classes which the quantitative comparison is based on. Of the three methods CycleGAN, SG-GAN and CyCADA only the latter one was able to improve semantic segmentation through adaptation from the synthetic GTA images to the real Cityscapes domain.

6.1. Outlook and Future Work

The field of domain adapaption for semantic segmentation from synthetic to real domains is a very popular field of research especially regarding the many resources and research that currently go into autonomous driving. In order to make autonomous vehicles reliable and consistent the currently available domain adaptation techniques will still need more improvement. A possibility to improve upon this work could be to train own models for each of the compared techniques. This ensures a fair comparison. Another would be to incorporate more techniques and other domain adaptation tasks such as indoor synthetic to real adaptation.

A. Appendix

Bibliography

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

[CBG13] Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. Dlid: Deep learning for domain adaptation by interpolating between domains. 2013.

[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.

[GKZ+16] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. *CoRR*, abs/1607.03516, 2016.

[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.

[HWYD16] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR*, abs/1612.02649, 2016.

[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.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 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.

[LSD14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[LXG+16] Antonio M. López, Jiaolong Xu, Jose Luis Gomez, David Vázquez, and Germán Ros. From virtual to real world visual perception using domain adaptation - the DPM as example. *CoRR*, abs/1612.09134, 2016.

[MLX+16] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 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.

[SFS15] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. *CoRR*, abs/1511.05547, 2015.

[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.

[SPT+16] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. *CoRR*, abs/1612.07828, 2016.

[SS01] George Stockman and Linda G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[SS16] Baochen Sun and Kate Saenko. Deep CORAL: correlation alignment for deep domain adaptation. *CoRR*, abs/1607.01719, 2016.

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

[THDS15] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. *CoRR*, abs/1510.02192, 2015.

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

[YKF17] Fisher Yu, Vladlen Koltun, and Thomas A. Funkhouser. Dilated residual networks. *CoRR*, abs/1705.09914, 2017.

[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.

[ZYCW15] Xu Zhang, Felix X. Yu, Shih-Fu Chang, and Shengjin Wang. Deep transfer network: Unsupervised domain adaptation. *CoRR*, abs/1503.00591, 2015.