POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Addressing Domain Shift between Real and Synthetic data for Semantic Segmentation with GANs

Supervisors

Prof. Tatiana TOMMASI

(Ext) Antonio MASTROPIETRO

Candidate

Walter MAFFIONE

April 2020

Abstract

Autonomous driving is without a shadow of doubt the future of automotive and it currently needs efficient solutions. Deep learning methods have shown excellent results for object classification and detection, but the specific task needed for driving is the more challenging semantic segmentation: each pixel of a depicted scene should be recognized as belonging to a specific object. For this setting, it is crucial to collect a large amount of per-pixel labeled data, which need an expensive manual classification process. Synthetic datasets from simulators can be used in order to reduce the amount of data required and they come with free annotation by design. However, the big style difference between synthetic and real images does not allow a direct knowledge transfer across the two domains and it asks for specific domain adaptation solutions. This work starts from a review of the most recent literature on deep learning for semantic segmentation and it proposes to boost domain adaptation techniques based on Generative Adversarial Networks (GANs) for style transfer by combining them with features-based adaptive strategies that take into consideration category-level adaptation and self-supervision. The experimental analysis considers a model learned on synthetic labeled images from the GTA V videogame and applied on the real Cityscapes dataset collected in different cities in Germany. The obtained results show how the proposed combination of methods can be useful in case of limited availability of real world annotated images.

Acknowledgements

"The first step is to establish that something is possible; then probability will occur"
Elon Musk

In this section I want to thank all the people who helped me in these years and who supported me, both for the studies and for the development of this work.This part will be in Italian in order to be easily understood from everyone

Ringrazio innanzitutto tutti i miei professori, in particolare la prof. Tommasi che si è resa disponibile ad aiutarmi in questo lavoro di tesi. Grazie alla loro saggezza e passione mi hanno trasmesso conoscenza per gli argomenti affrontati e acceso in me la curiosità necessaria per rimanere sempre informato sul mondo circostante e per poter dare un contributo alla comunità.
Desidero inoltre ringraziare l'azienda AddFor e Antonio Mastropietro per l'aiuto datomi nello svolgimento di questa tesi e nel migliorare le mie capacità comunicative, per avermi dato accesso e disponibilità a tutte le risorse necessarie al conseguimento di questo lavoro. Un grazie speciale va prima di tutto alla mia famiglia, per il supporto emotivo ed economico, grazie al quale ho potuto concludere i miei studi senza preoccupazioni. Ringrazio la mia fidanzata Marlisa che con il suo affetto e calore mi ha accompagnato lungo questa strada, tenendomi alto il morale anche nei momenti difficili e aiutandomi a superare ogni possibile ostacolo. Un pensiero particolare lo dedico all'aiuto, alle risate e ai momenti di spensieratezza dei miei amici e colleghi che hanno reso questo percorso sereno e più semplice da affrontare.

Table of Contents

List of Tables

List of Figures

Chapter 1

Indroduction

1.1 Autonomous Driving

Autonomous driving is without a shadow of a doubt the future of automotive. The advantages that derives from a self driving car are astonishing, the main important can be less deaths on the streets and a better human experience while sitting on the car. To get to the point when cars will guide entirely alone however a lot of steps needs to be done and will require still some years. There is the need that every single little detail is taken in consideration and every single model, validated and tested, in order to ensure sufficient security and reliability. In order to obtain a vehicle able to guide by itself there is the need to obtain semantic information about the environment around the vehicle, and different sensors can be used in order to achieve this task. Some of those sensors can be cameras for visual recognition task, because it's important to recognize the drivable area, pedestrians and car on the street, road signs, etc; or radar at short-long range distance in order to perceive accurate depth information. From an union and analysis of the different information collected by the sensors of the car is possible with automatic control techniques to create algorithm able to allow the car to drive by itself.
In this work we refer to the task of Computer Vision called Semantic Segmentation [1], that by taking in input images of the road from the camera in front of the car, output a prediction which contains a classification for each pixel of the image as belonging to a series of classes that are taken in consideration in the training phase of the model.

1.2 Limitations of Semantic Segmentation

In order to train a model of Semantic Segmentation that will then insert in a vehicle with the aim of substitute the human eye, there are many precautions to consider. First of all, we have to think to the nature of such architectures, in fact the most important features that a real-time scene understanding model should have, are precision and inference speed. In order to entrust our lives to such models we need to be sure that those systems are reliable and extremely accurate, however this is not yet the case, this technology remains a open research field, and even Tesla, the company with the most advanced resources for self-driving car, proposes its auto-pilot as a beta program.
The difficulty to generalize to new data unseen by the model in the training phase is a common problem in Deep Learning, however it represents a far more serious problem when there are human lives that depend on it.
Encoding billions of data of world roads to a single model composed of several convolutional layers it is not conceivable, there is too much information that needs to be compressed in a few millions of parameters. For this reason models that will be then used in a real scenario should be geographically scalable.

One current problem of Semantic segmentation is data collecting.
Roads and streets around the world are very different, both between different countries, both between regions or cities of the same country, and even inside the same city. In order to get a robust model there is the need to collect the most possible amount of data, because we would like that our car will drive autonomously in different scenarios. This process is very expensive for companies that needs to collect data for all the streets that they would like their self-driving car will cover. Moreover also weather and light conditions really matter when using CNN-based models, for this reason the amount of data required is even greater.
As an example, if we train a Semantic segmentation model with data containing images of some streets of the city of Turin taken in a rainy and cloudy day, when we try to test the same model on images taken in a sunny day in Rome, the model will absolutely have poor performance. This is due to the domain shift of the two different cities, colours and brightness of the pixels are really different, objects and topography of the streets are even more diverse, considering also to use the same camera with the same shooting angle. Convolutional filters of the model overfit to the training data and this means that they will not be able to generalize to a new domain completely different [2].
The networks need to see data similar to those that will be used at test time, despite the property of *independent and identically distributed* for training and test data, on which machine learning is based, does not exist in the real world.

Notwithstanding these problems, the biggest limitation of Semantic Segmentation, is the labeling process. In order to train a model we need to provide it in input the image and the corresponding label, on the image thanks to the learnt parameters of the network it's output a prediction. A loss function will then calculate the error committed with respect to the true label, and using an optimizer such as *stochastic gradient descent* the error is minimized by updating the parameters of the network through backpropagation algorithm.
For Semantic segmentation the true label is composed of different layers, one for each class that has to be considered. For each of them one person, needs to highlight every single pixel of the image that contain that specific class. This process require a lot of time, for Cityscapes dataset [3], public available most used dataset in research, this process required about 1.5 hours for every single image. For this reason this is currently the main bottleneck of Semantic Segmentation for urban scene.

1.3 Generalizing with Synthetic data

Synthetic data from simulators can be used in order to train Semantic Segmentation model providing a data augmentation to the real dataset.
One of advantages of synthetic data is that thanks to the rendering engine it is possible to create realistic virtual environment in order to collect data as close as possible to real ones, without requiring a physical process that does it. Moreover from virtual environment is possible to change weather and light conditions easily, allowing to collect more diverse data.
The biggest advantage of synthetic data is that there is no need to segment by hand the labels, as these are created autonomously by the rendering engine that needs to know where create 3D objects in the scene, for those reasons the amount of time needed to collect training data is drastically cut off.

However there are still some problems when applying models trained with synthetic data on real scenarios. In fact synthetic and real data may be very different, even if the realism and quality of synthetic data is extremely high, there may be some low details like textures, reflex and brightness that can reduce the performance on the real data. This problem known as domain shift, it's possible to tackle with Domain Adaptation [4] that tries to learn a shared representation between domains in order to learn from synthetic data and generalize on real ones.

In this work I decided to elaborate on Adversarial based techniques [5] of Domain adaptation, due to their relevance in the research community in recent years.
In particular after a thorough study of the literature and state-of-the-art techniques that try to solve the gap between synthetic and real data, I noticed that several

good ideas were proposed but a follow-up to these ideas is hardly proposed.
For this reason I decided to mix some good and promising techniques in order to reduce the gap between two datasets publicly available, which were used as benchmark in those relative articles. My purpose is to obtain competitive results with the one proposed in such papers. For the synthetic dataset was used GTA V [6], data taken from the famous open-world videogame set in Los Angeles, while for the real dataset was used Cityscapes [3].
The algorithm is based on a two-steps process, the first step is a pixel level adaptation, trying to apply Cityscapes style to the synthetic dataset, by using an Image to Image transition model called cycleGAN [7], in order to create an adapted dataset that will be used in the next step. The main part of the algorithm is based on CLAN [8], composed of a semantic segmentation model based on DeepLab v2 [9], on which is applied a Domain adaptation module on the output predictions that thanks to a Discriminator model produce a weighted adversarial loss which has the effect of aligning the marginal distributions of two domains, making convolutional layers able to generalize on real data that is provided without labels during training.
To this baseline is added a Self-supervised task of Rotation on the output of the model [10], the predictions after an up-sample layer in order to get squared shape are randomly rotated in one of the possible outcome [0, 90, 180, 270], and an auxiliary network is trained in order to learn which rotation has been given to the prediction.
This last process has the effect of making convolutional filters able to learn spatial invariant features, allowing a better generalization for the target domain.
Several experiments have been produced, firstly in an Unsupervised learning scenario in order to obtain the most possible information from synthetic data and to get comparable results with the ones proposed in the different articles. Then experiments have been produced in a Semi-supervised approach, considering using only a fraction of the training real labels in order to provide more useful results from an industrial point of view. The results obtained outperform the ones proposed in the original articles, by training the models only half of the time as done in the papers due to time and computational requirements. Moreover, semi-supervised experiments show that using only a fraction of the real label available, thanks to domain adaptation from synthetic data it's possible to get results better than a model trained only on the full real labeled dataset.
I'd like to thank AddFor S.p.a. for providing me a guide and resources in order to produce this work.

Chapter 2

Generative Adversarial Networks

2.1 Introduction

Generative Adversarial Networks (GANs) belongs to the family of Generative models. They are used in a great vastness of applications, from the generation of new images that tries to resemble a distribution of real training data [5], to more advanced techniques useful for different tasks, such as domain adaptation [4], style transfer [11], text2image, adversarial examples, deep fakes and more. They are considered as "the most interesting idea in the last 10 years of machine learning" by Y. Le Cun, one of the fathers of Deep Learning.

GANs were first formulated in 2014 by Ian Goodfellow and his colleagues [5], in its easier form as it was proposed, a Generative Adversarial Network it's composed of two neural networks that continuously try to beat each other in a so called minimax game. The first player of the game is the Generator (G), whose objective function is to generate new data with the same statistics of a training set, starting from a random noise vector, in such a way that they are indistinguishable from the real ones. The other opponent is the Discriminator (D), that instead has the task to distinguish the real data from the ones generated by the Generator.

More formally, given two differentiable functions G and D represented by two multi-layer perceptrons, with parameters respectively θ_g and θ_d. We can define a latent variable z as input of G, whose task is mapping to the data space $G(\boldsymbol{z}; \theta_g)$ in order to learn the distribution p_g over data $\boldsymbol{x}$. On the contrary $D(\boldsymbol{x}; \theta_d)$ outputs a single value, that represents the probability that $\boldsymbol{x}$ came from the data rather than p_g. D is trained in order to maximize the probability of assigning the correct label for both training examples and samples generated from G, while G is trained in

order to minimize the probability of being guessed by D, or rather $\log(1-D(G(\boldsymbol{z})))$.

The value function corresponding to the GAN training is reported in 2.1

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \quad (2.1)$$

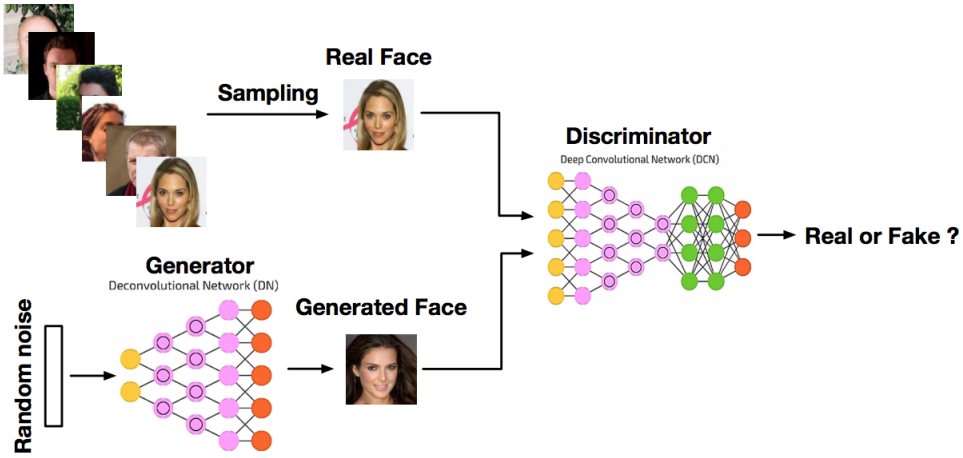


Figure 2.1: Architecture of a GAN for face image generation, from [12]

Procedure of training of Generative Adversarial Networks is presented in Algorithm 1.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. In the original paper is used $k = 1$, as the least expensive option.

for number of training iterations **do**
  for k steps **do**
    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for
  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for
The gradient-based updates can use any standard gradient-based learning rule

2.2 Generative Models

Generative models are able to define how a set of data is generated in terms of probabilistic model, by sampling from this model could be possible to create new data similar to the first one.
Given a set of data instances X and a set of labels Y, a Generative model captures the joint probability $p(X, Y)$, or just the distribution of X, $p(X)$ if there are no labels. In other words, these models learn to create data similar to that given in input to them. A Discriminative model instead is able to capture the conditional probability $p(Y|X)$ so assigning a label class to a specific sample.
All types of generative models try to learn the true data distribution of a training set, so as to generate new data points with same statistics. But it is not always possible to learn the exact distribution of our data either implicitly or explicitly and so we try to model a distribution which is as similar as possible to the true data distribution. For this reason, we can exploit the power of neural networks to learn

a function which can approximate the model distribution to the true distribution, and Generative Adversarial Networks are one the possible techniques to do that.

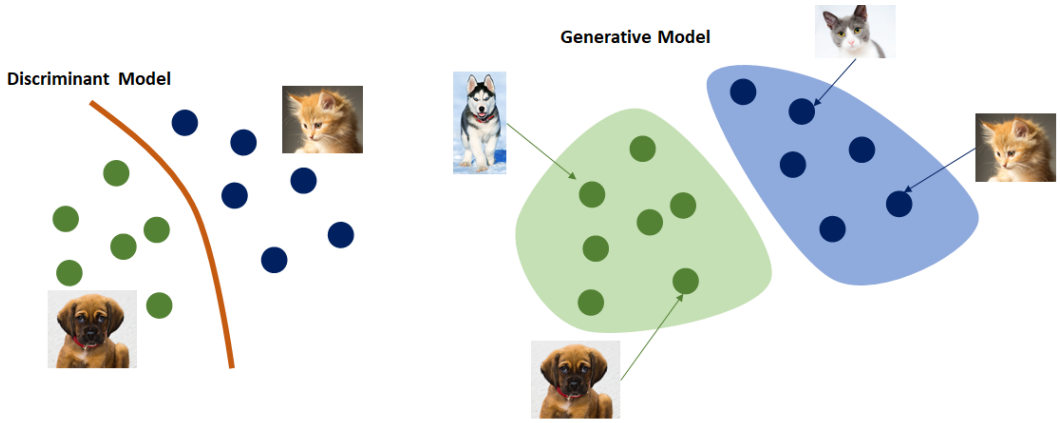


Figure 2.2: A discriminative model on the left defines a boundary in order to distinguish dogs from cats, a generative model on the right, instead is able to generate new samples similar to existing ones

2.3 Problems in training GANs

Generative Adversarial Networks are obtaining impressive results nowadays, generating images that could be very difficult even for an human eye to distinguish that are fakes. However the training process of a GAN is very hard and reach an equilibrium in the mini-max game between the two component is complex.
GAN models suffer of different problems, which can be categorized into three different categories, vanishing gradients, mode collapse and non convergence.
Generative Adversarial Networks are based on a zero-sum non-cooperative game, also called mini-max. In short, if one of the player wins the other loses.
Generator and discriminator try to beat each other, like in the game guards and thieves, in which one opponent wants to maximise an objective function while the other one wants to minimize it (equation 2.1). In game theory, the GAN model converges when discriminator and generator reach an equilibrium, optimal point for the mini-max game, called Nash equilibrium. This in an unstable equilibrium point, thinkable as a saddle point in a 3D dimensional space. A Nash equilibrium happens when one player will not change its action regardless of what the opponent may do.
So due to the adversarial game, each player of the game may undo the progress achieved during training, making convergence very difficult to achieve.

Equation 2.1 in practise, doesn't provide enough gradient to the Generator, that will not learn well the training distribution. In the first iterations of the training loop, when G is poor, it's too easy for D to reject samples generated because they are clearly different from training data. This is due the saturation of $log(1 - D(G(z)))$, and this problem is known as vanishing gradient. A possible solution to this problem can arise by instead of minimizing $log(1 - D(G(z)))$, by maximize $log(D(G(z)))$ for D, this provides much stronger gradients for G in early stage of training and results in the same fixed point of the dynamics of the training. In practise this is achieved during training by changing label values for real and fake data.
Another solution to provide better gradients to G during training is the usage of different loss functions, in order to achieve more stable training and better results.
Another hard problem to solve that appears when training GANs is mode collapse. Mode collapse appear when the generator collapses, generating always a limited variety of samples, . It's possible to see an example in figure 2.3. This happens because during training the generator may find some optimal parameters that generate images that always fool the discriminator, leading to a collapse of the modes and becoming unable to generate more varied samples. Mode collapse can be seen as a overfitting of the generator.

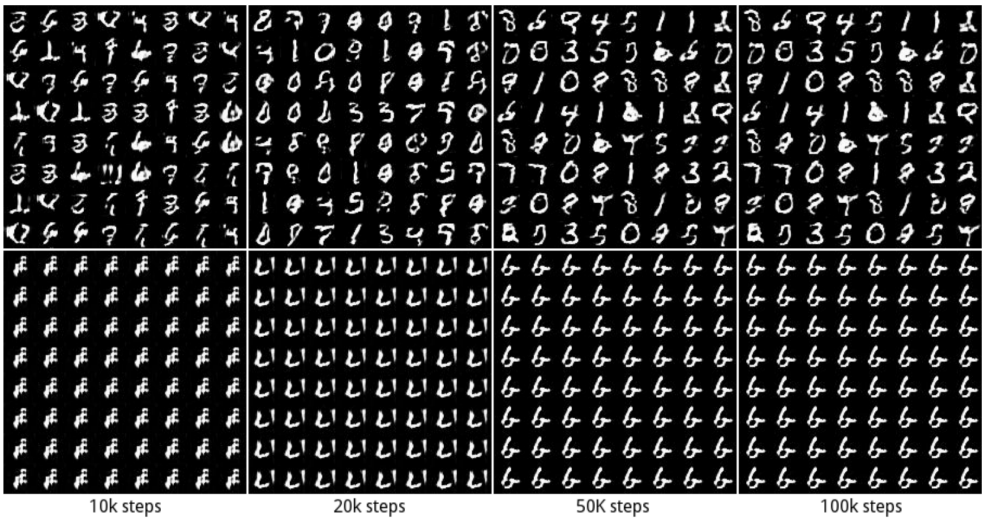


Figure 2.3: Example of mode collapse. Two different GANs, in the first row a GAN is able to create all possible modes (number 0-9), in the second row a GAN collapses in generating one single number. [13]

GANs are highly sensible to hyperparameters, tuning parameters in order to find values that bring the best results takes time and patience. It's very easy to fall in a unstable training and non convergence due to a wrong choice of initial

parameters that generate low quality results.
Moreover another big problem in the GAN training is that differently from discriminative models and CNN, loss functions did not tell anything about the trend of the learning process. For this reason we need to examine the generated images manually to verify the learning process. This makes model comparison harder leading to difficulties in picking the best model in a single run.

2.4 Possible Solutions

Different solutions are proposed in literature trying to solve the problems of GANs. Some of them are based on regularization tricks, which are not based on mathematical evidence but rather on heuristic evidence based on experiment made by the research community.
Others instead are based on a more theoretical assumption such as a change of the loss function that can provide more meaningful information for the generation of images, trying to eliminate vanishing gradients problems. Even architecture changes provide boost in performance in the generation of high quality images, more diverse generated samples and a fairer game between generator and discriminator that provides a general higher stability of the training process of the model.

2.5 Loss Function

The choice of the loss function is a design decision that significantly impacts performance in GANs. In the original paper has been proved global optimally and convergence of GANs but it was also highlighted the instability problem which can arise in the training phase. The loss function of a GAN, when the discriminator is optimized, is related to two probability measurement metrics, Kullback-Leibler (KL) divergence and Jensen-Shannon (JS) divergence. Minimizing JS divergence between real and generated distribution is possible to optimize the generator considering an optimal D. However, it has been proved that using original loss function will result in the vanishing gradient for G, in the early steps real and generated distribution differ too much and the JS divergence has a low gradient that will allow D to prevail and G not to learn. While using the alternative loss function proposed (minimizing -log(D(G(z)) instead of log(1-D(G(z))) will incur the mode collapse problem (same G(z) for different z) and fluctuating gradients that cause instability to the models.
These problems cannot be solved changing the architecture, but only changing the loss function, the aim is to find a cost function with smoother and non-vanishing gradients, which is intensively researched in order to improve learning stability and ability of the model.

Some of the loss functions that have been seen to work well with GANs are:

- Wasserstein GAN (with gradient penalty)
- Least Squared GAN
- Spectral Normalization with Hinge Loss

2.5.1 Wasserstein GAN

WGAN [14] proposes the Wasserstein distance to measure the difference between the data distributions of real and generated images. Intuitively, it measures the effort to transform one data distribution to another.
Wasserstein distance (or Earth mover) is able to reflect distance also when two distributions are far, for example in the early iterations of a GAN training, when for the discriminator is easier to distinguish if a generated image is fake or not. This provides much meaningful and smooth gradients during training, remedying to the problem of vanishing gradient. Mathematically, Wasserstein distance looks more desirable as a cost function with respect to the one proposed in the original paper.
The Wasserstein distance is defined as:

$$W(p_r, p_g) = \inf_{\gamma \in \prod(p_r, p_g)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma} \|\mathbf{x} - \mathbf{y}\|, \tag{2.2}$$

where $\prod(p_r, p_g)$ is the set of all possible joint distributions and $\gamma(\mathbf{x}, \mathbf{y})$ whose marginals are p_r and p_g. However, this equation for the Wasserstein distance is highly intractable, for this reason authors demonstrate that the distance can also be estimated as:

$$\max_{w\sim\mathcal{W}} \mathbb{E}_{\mathbf{x}_{p_r}}[f_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z}[f_w(G(\mathbf{z}))], \tag{2.3}$$

where f_w takes the role of the critic D with some constraints that requires to be a 1-Lipschitz function, $\mathbf{z}$ is the input noise for G. So w are the parameters of D and D aims to maximize equation (2.3) in order to make the optimization distance equivalent to Wasserstein distance. When D is optimized, the task of G will be to minimize the Wasserstein distance, or rather

$$-\min_G \mathbb{E}_{\mathbf{z}\sim p_z}[f_w(G(\mathbf{z}))] \tag{2.4}$$

The discriminator in the original work is used as a binary classifier but D used in WGAN needs fit the Wasserstein distance, which is a regression task. Thus, the sigmoid in the last layer of D is removed in the WGAN.
The benefits of WGAN are that the training process is more stable and less sensitive to the choice of hyperparameters and network architecture. Moreover, the loss of

the discriminator appears to relate to the quality of images created by the generator. Finally, by adding a gradient penalty to D is possible to obtain much a much stable network.

2.5.2 Least Squared GAN

Authors of Least Squared GAN (LSGAN) [15] propose a new approach to remedy the vanishing gradient problem, by proposing a new cost function based on least square loss for the discriminator instead of sigmoid cross entropy loss.
We can define the proposed loss function for G and D as:

$$\begin{aligned} \min_D \mathcal{L}_D &= \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_r}[(D(\mathbf{x})-b)^2] + \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_z}[(D(G(\mathbf{z}))-a)^2], \\ \min_G \mathcal{L}_G &= \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_z}[(D(G(\mathbf{z}))-c)^2], \end{aligned} \tag{2.5}$$

where a is the generated sample label, b is the real sample label and c is a hyperparameter that G wants D to recognize the generated samples as the real samples by mistake.
The new decision boundary that arises by minimizing the loss for D, penalizes large error arising from those generated samples that are far away from the decision boundary, which pushes towards the decision boundary those bad generated samples. This allows to get better image quality in generating images.
Moreover, by penalizing the generated samples that are far away from the decision boundary, it is possible to provide better gradients when updating the generator, remedying the vanishing gradient problems for training G.

2.5.3 Spectral Normalization with Hinge Loss

In order to further stabilize the training of GANs, authors of Spectral Normalization GAN (SN-GAN) propose the usage of weight normalization. This technique does require a light computational effort and it's easily applicable to existing GANs.
Spectral normalization is calculated as:

$$\bar{\mathbf{W}}_{SN}(\mathbf{W}) = \frac{\mathbf{W}}{\sigma(\mathbf{W})}, \tag{2.6}$$

where $\mathbf{W}$ represents weights on each layer of the discriminator D and $\sigma(\mathbf{W})$ is the L_2 matrix norm of $\mathbf{W}$. The paper proves this will make $\|f\| \leq 1$, requirement that allows a better stability of the network. Spectral Normalization has been found to work in the best way with Hinge loss.

2.6 GAN Architectures

Several approaches have been tried in order to overcome the several issues of adversarial training, such as the usage of a different loss function, as we saw before. Although some of them improve stability and vanishing gradient problems, none of them eliminated the issue of mode collapse entirely and theoretical reason remains an area of active research.
Moreover the traditional architecture of GAN produces very low quality images due to the well known problems of fully connected layers, that are not suited for learning hierarchical structure like the ones present inside the images.
For this reason, it is necessary not only to change the optimization functions but also the entire architecture and structure of the networks.

2.6.1 Deep Convolutional GAN

Deep Convolutional GANs (DCGAN) [16] is a popular architecture of GAN which replaced the original fully connected network as baseline model for GANs. It mainly removes all max pooling and fully connected layers, replacing them with convolutional layers.
For the generator that has the task of starting from a latent vector to create an image similar to the one present in a training set, the upsampling task is performed with the usage of transposed convolutional layers. We can see an example in the figure 2.4 below.
The discriminator instead is composed of downsampling task performed by convolutional layers.

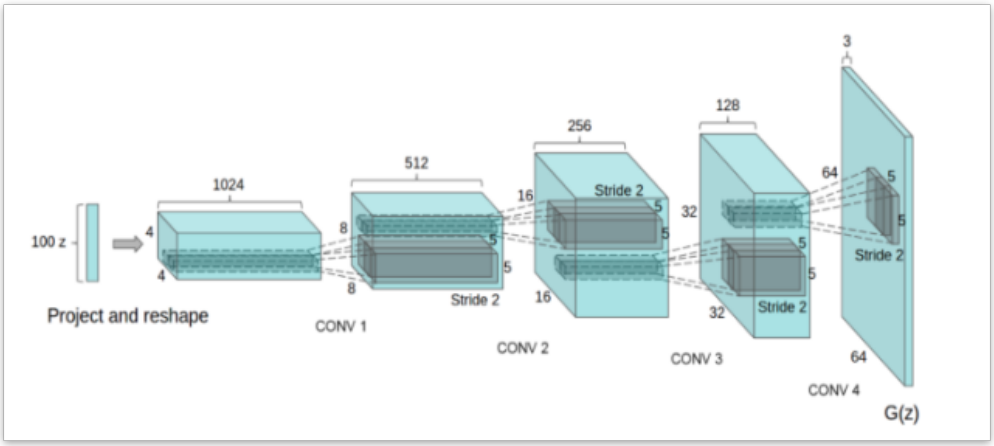


Figure 2.4: Generator architecture of a DCGAN

Authors propose the usage of batch normalization for all the layers except for the output layer of the generator and the input layer of the discriminator. The activation functions used are ReLu for generator, except for the output which uses tanh, and Leaky-ReLu for discriminator.
It's a very simple architecture that is possible to combine with different loss functions and regularization tricks in order to produce good quality images.

2.6.2 Progressing Growing GAN

The generation of images in high resolution was a big problem in early ages of GANs. The higher the resolution of an image is, the easier it becomes for the discriminator to distinguish real images from the fakes. This makes also mode collapse more likely. For those reasons researchers at NVIDIA studied a method to overcome those issues, developing the PROGAN [17]. The main feature of the PROGAN is the Progressing Growing mechanism of the network. Instead of attempting to train all layers of the generator and discriminator at once, as it's done normally, the network starts with low-resolution images (4x4) and the progressively grows in order to handle higher resolution, by adding iteratively layers to the network. This incremental nature allows the training to first discover large-scale structure of the image distribution and then shift the attention to an increasingly finer scale detail, instead of having to learn all scales simultaneously.
By increasing the resolution gradually, we are asking the network to learn much simpler a piece of the overall problem, this, in combination with some other training details, reduces the chance for mode collapse and stabilizes training. Another improvement of the progressing growing is the faster training process, this because fewer layers have simply less parameters in them, and only the final set of training iterations are done with the right resolution. Authors found that their PROGAN generally trained twice to 6 times faster than a corresponding traditional GAN, depending on output resolution.
Differently from DCGAN, Progressing growing GAN uses the nearest neighbors for upsampling and average pooling for down-sampling. These are simple operations with no learnable parameters, both then followed by two convolutional layers. When new layers are added, the parameters in the previous layers remain trainable. In order to prevent shocks in pre-existing lower layers when adding a new top one, the new layer is linearly "faded in" like a residual block, controlled by a parameter α depending on the number of iterations passed. In this way, the network can adapt itself to the new layer (Figure 2.5).

Moreover authors introduced some training details that stabilize the training process and could be taken in consideration in order to improve existing or different GAN architectures.
Instead of worrying about covariance shift and so using BatchNormalization layers, authors have found that this is not an issue in GANs, while the real problem is to guarantee a fair game between G and D in order to prevent vanishing or exploding gradients. For those reasons they used two different approaches, neither of which include learnable parameters.

- PIXEL NORMALIZATION: using pixel normalization instead of batch normalization as it's normally done, it has the benefit of not requiring trainable

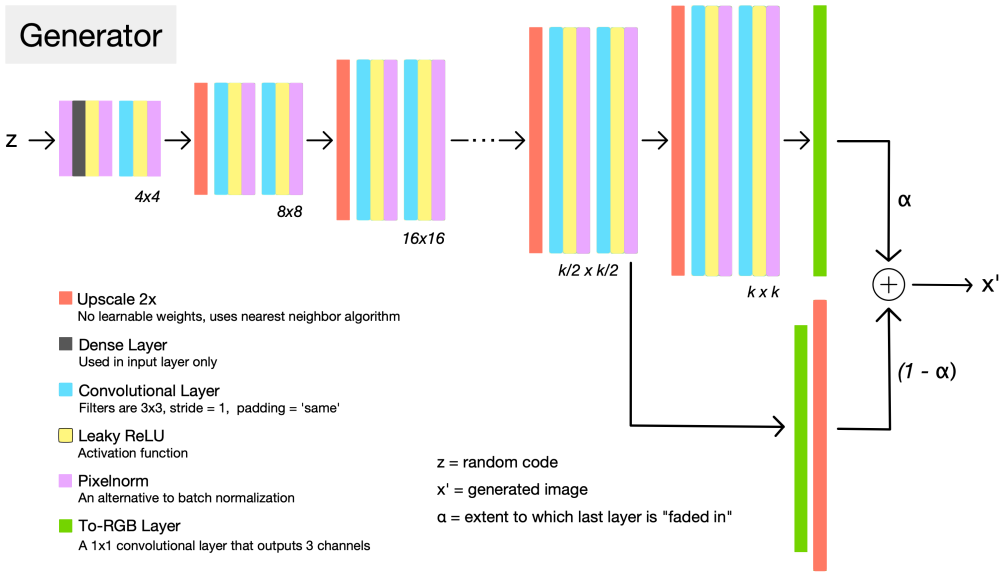


Figure 2.5: Generator architecture of a PROGAN

parameters. In order to prevent the escalation of signal magnitudes (gradients) as result of competition between D and G, the feature vector in each pixel is normalized to unit length in the generator after each convolutional layer.

- EQUALIZED LEARNING RATE: Instead of using weight initialization, authors have found that scaling weights at run-time produce better results. Commonly used network optimizers like RMSProp or Adam, normalize a gradient update by its estimated standard deviation, thus making the update independent of the scale of the parameter. As a result, if some parameters have a larger dynamic range than others, they will take longer to adjust. So it is necessary that layers learn at a similar speed. To achieve this equalized learning rate, they scale the weights of a layer according to how many weights that the layer has. They do this using the same formula as is used in He initialization [18], except that they do it in every forward pass during training, rather than just at initialization. By doing this, no fancy initialization is needed despite a standard normal distribution.

- MINI-BATCH STANDARD DEVIATION: One of the many problems of GANs is the difficulty of producing samples with a wide variation such as in the training data, problem also known as mode diversity. In order to combat this issue, it is possible to allow the discriminator to compute statistics across the batch and use this information as help to distinguish between real and fake images. This encourages the generator to produce more diverse images, trying to ensemble real data. In PROGAN this is done with a Minibatch Standard

Deviation layer, inserting near the end of the discriminator, it computes the standard deviations of the feature map pixels across the batch and it appends them as an extra channel, with no trainable parameters nor new hyperparameters.

PROGAN is independent from the loss function used, but WGAN-GP is the one that produced better results.
The speedup from progressing growing increases as the output resolution grows, and it produced stunning results in high resolution images (1024x1024) for CelebaHQ dataset.
Even if the results are generally high compared to earlier work on GANs and the training is stable in large resolution, there is yet a long way to true photo-realism. Semantic sensibility and understanding dataset-dependent constraints, such as certain objects being straight rather than curved, leaves a lot to be desired. There is also room for improvement in the micro-structure of the images.

2.7 Regularization tricks

In order to obtain better stability of the GAN models, researchers found different tricks that are not based on mathematical assumptions rather on heuristic and the so called "learn by doing".
In GAN papers, the loss function to optimize for G is $min(log(1 - D(G(z))))$, minimizing the probability that the generated sample will be classified as fake from D, but in practice it is better to $max(logD(G(z)))$, or rather maximizing the probability for D to make an error when classifying the generated sample. This provides better stability to the network and meaningful gradients to G during training, the easiest way to do that is to flip labels while training the generator.
In order to generate better quality images, it was found that it is better to sample the latent vector z from a Gaussian distribution than a Uniform.
When training is better to construct different mini-batches for real and fake, so not using batches with both real and fake images to give in input to the discriminator.
Stochastic Gradient Descent optimizer works better for the discriminator, while Adam for the generator.
Moreover, conditioning the training with labels, if present, allows to get better quality images. Finally it has been found that sometimes training D more than G provides more stable training.

2.8 Evaluation metrics

There is still no clear consensus on which GAN algorithm perform objectively better than others. This is partially due to the lack of robust and consistent

metric, as well as limited comparisons which put all algorithms on equal footage, including the computational budget to search over all hyperparameters. Many researchers focused on qualitative comparison, such as comparing the visual quality of samples. Unfortunately, such approaches are subjective and possibly misleading, for this reason computing a fair comparison between methods implies access to some metrics. The most used nowadays are IS (Inception Score) and FID (Fréchet Inception Distance), both assume access to a pre-trained classifier, Inception Net trained on ImageNet.

- INCEPTION SCORE (IS): it offers a way to quantitatively evaluate the quality of generated samples. It is based on the fact that a good model should generate samples for which, when evaluated by the classifier, the class distribution has low entropy. At the same time, it should produce diverse samples covering all classes.

- FRÉCHET INCEPTION DISTANCE (FID): it is computed by considering the difference in embedding of true and fake data given by (a specific layer) of Inception Net. Assuming that the coding layer follows a multivariate Gaussian distribution, the distance between the distributions is reduced to the Frèchet distance between the corresponding Gaussians. Unlike IS, FID can detect intra-class mode dropping, i.e. a model that generates only one image per class can score a perfect IS but will have a bad FID.

Both measures however are unable to detect overfitting, a "memory GAN" which stores all training samples and generates replicating them would score perfectly.
Even when the metric is fixed, a given algorithm can achieve very different scores, when varying the architecture, hyperparameters, random initialization (i.e. random seed for initial network weights), or the dataset.
In the paper "Are GAN Created Equal? A Large-Scale Study" M.Lulic, K.Kurach [19] – authors computed a large scale experimental evaluation, performing a huge hyperparameters optimization for each model (architecture was always the same but objective function changed) and dataset (CIFAR10).
Important observations come from that work, firstly it results that there is no algorithm which clearly dominates others. Secondly, for an interesting range of FID scores, a "bad" model trained on a large budget can outperform a "good" model trained on a small budget. Finally, when the budget is limited, any statistically significant comparison of the models is unattainable.
Authors also suggest that differences between different methods may occur during testing of bigger networks (more params) on higher resolution and higher complex datasets, after the choice of the optimization method, the number of training steps, and possibly other optimization hyperparameters.
Authors observed that the performance of each model heavily depends on the

dataset in which it was trained and no model strictly dominates the others.
In my personal opinion this means two things, firstly that current metrics are ineffective since they should define more difference even for simple dataset and secondly that a GAN less sensible to hyperparameters optimization is needed in order to perform well on a vast amount of dataset and to be considered better than others. Despite these problems, however, it is suggested using a loss function that at least in theory and mathematically resolves the problems of vanishing gradient, mode collapse and unstable training, despite they produce similar results on easy dataset according to FID, it does not mean that they are all comparable in terms of stability and mode diversity. For this reason, it is highly recommended to use loss function such as Least Squared or WGAN-GP instead of standard GAN loss function.
All GANs problems must be incorporated into a good hypothetical metric in order to ensure a fairer comparison between different models.
Nowadays in case of using GANs in a different application from the standard Image Generation, the best way to understand how network learning proceeds is still a plot of intermediate results.

2.9 Applications

Image generation is only one of the possible applications of Generative Adversarial Networks. GANs have become more and more mainstream in the last years, recently they have become known to the public for the deep fake application.
Another very interesting application of GANs is the image-to-image process, in which images can be manipulated by translating from a domain to another. Due to the ability of adversarial networks of reducing gap between distributions, GANs are widely used in Domain Adaptation techniques.
Video manipulation is one of the most critical aspects and applications of GANs. Today there are already cases of manipulation of videos for political or defamatory purposes thanks to Generative Adversarial Examples. Ethical and social aspects of these applications should make us reflect on the power of this tool, for this reason, applications of fake image or video detection need to be developed in parallel or rather in a priority mode. Through the advancement of these technologies will be more and more difficult for the human eye to distinguish real from fake content, such that super-human classification task as yet be proved.
Computer vision is not the only application field of GANs, researchers of Open AI developed a tool for generating text, as natural language processing application.
Scott Reed, et al. in their 2016 paper entitled "Generative Adversarial Text to Image Synthesis" also provides an early example of text to image generation of small objects and scenes including birds, flowers, and more.
Image in-painting allows to reconstruct portion of image that are missing, always using GANs.
Others interesting application are super-resolution, 3D object generation and video prediction.

Chapter 3

Semantic Segmentation

3.1 Introduction

Semantic segmentation is a key component in many visual understanding systems. Its main applications are medical image analysis, autonomous vehicles, video surveillance, authentication systems and robotic perception [20].
Before the Deep Learning age there were different techniques to perform semantic segmentation, such as Markov random fields [21] or sparsity based methods. Then with the rise of Artificial and Convolutional Neural Networks, Deep Learning models outperform previous methods and became standard approach for this kind of tasks.
Semantic segmentation is used when there is the need of obtaining fine semantic object information, for example for understand where precisely appear a road sign on a street. It's a more complex task than image classification or object detection. In semantic segmentation the objective is to classify every single pixel of an image as belonging to a set of specific classes [1]. Deep learning methods for semantic segmentation can be grouped in different categories, where the most important and used today are:

- Fully Convolutional Networks
- Encoder-Decoder based models
- Dilated Convolutional model and DeepLab family

The most used loss function that needs to be minimized in order to update the weights of the network is the multi-class cross entropy loss.
Given a set of images X with corresponding ground truth labels Y of resolution hxw, and C classes, we can define the segmentation loss as:

$$\mathcal{L}_{seg}(X, Y) = -\sum_{h,w} \sum_{c \in C} Y^{(h,w,c)} \log(P(X)^{(h,w,c)}), \tag{3.1}$$

where $P(X)$ is the prediction of the network for an image of X.

As it has been discussed in the introduction, the main problem of Semantic segmentation is the collection of ground truth labels. In fact in order to provide labels for semantic segmentation there is the need that a human, for each class that need to be considered, has to evidence each pixel of an image that belongs to that specific class. This process is extremely expensive and in the worst cases may require a few hours for single picture [3]. This is the current main bottleneck of semantic segmentation, and in this work we try to overcome this issue by using domain adaptation techniques exploiting synthetic labeled images.

3.2 Fully Convolutional Networks

Long et al. [1] proposed one of the first deep learning works for semantic image segmentation, by using a fully convolutional network (FCN).
A fully convolutional network, as the name may let guess (Figure 3.1), is composed only of convolutional layers, which enables to take an image of arbitrary size and produce a segmentation map of the same size. The authors modified existing CNN architecture, an AlexNet [22], to manage non-fixed sized input and output, by replacing all fully-connected layers with the fully-convolutional layers. As a result, the model outputs a spatial segmentation map instead of classification scores.

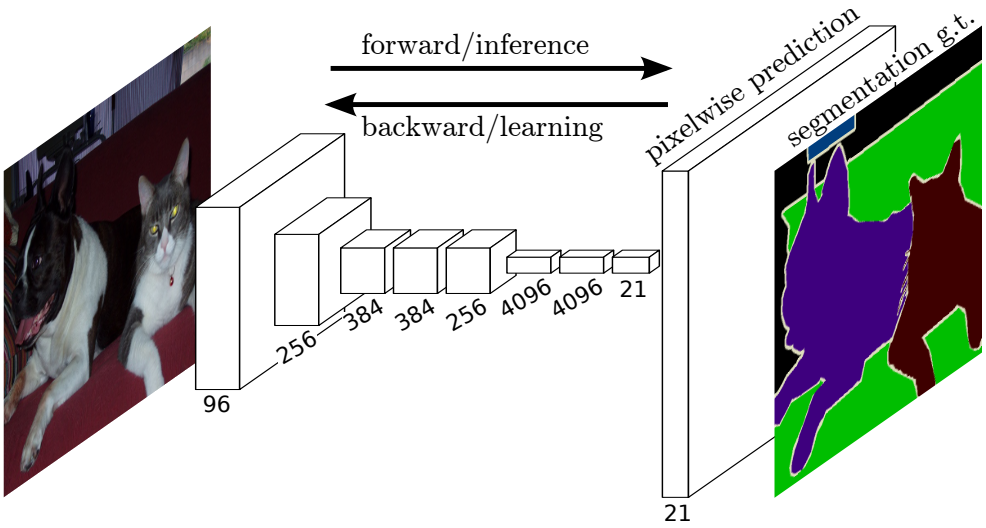


Figure 3.1: Fully Convolutional Network model, from [1]

A convolutional layer is composed of several three-dimensional arrays also known as filters or kernels, of size h × w × d, where h and w are spatial dimensions of

the filter, and d is the feature or channel dimension.
In order to extract features is performed a convolution between an input feature map (initially is the input image with size h x w x 3 (rgb)) and the relative filter. Parameters of the filters are learned during training by minimizing the pixel-wise segmentation loss. Stride of convolution, kernel size, number of filters and padding are hyperparameters of the convolutional layer.
After that convolutional layers extract the features map, are added pooling layers that have the task of reducing the number of parameters and dimensionality but maintaining spatial and semantic information. Finally activation functions allow to provide non linearity inside the network.
On top of the encoder network is appended a decoder module with transposed convolutional layers to upsample the coarse feature maps into a full-resolution segmentation map.
In a convolutional network, earlier layers tend to learn low-level concepts while later layers develop more high-level and defined feature maps. In order to maintain expressiveness, is typically increased the number of feature maps as we get in deep trough the network.
A convolutional layer has the task of extracting features from an input image, but differently from a classification task where we just want to know what object is present in an image, in semantic segmentation we also want to know where this object is located. For this reason we need to reconstruct the original image resolution by performing upsample operations on top of the final feature maps.
However, because the encoder module reduces the resolution of the input by a factor of 32, the decoder module struggles to produce fine-grained segmentations. For this reason authors decide to adding skip connections between layers of the network, in order to provide the necessary details to reconstruct accurate shapes for segmentation boundaries. A skip connection connects the output of one layer with the input of an earlier layer. They are able to recover more fine-grain details, lost due too many convolutions. We can see an example of skip connections in figure 3.2.

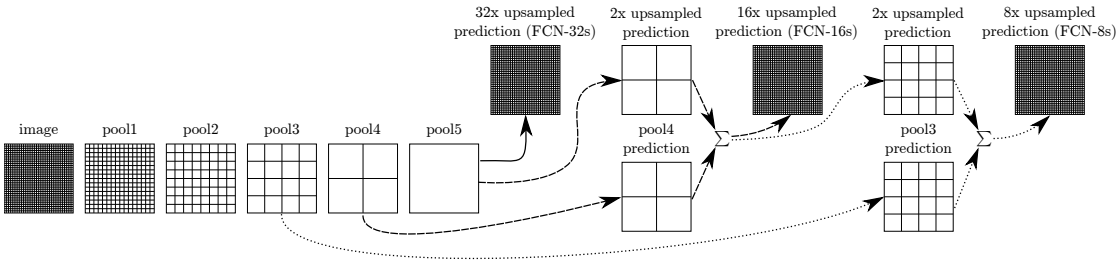


Figure 3.2: Skip connections of FCN network, from [1]

3.3 DeepLab v2

Authors of DeepLab v2 [9] propose three different solutions in order to further enhance performance of semantic segmentation models based on convolutional neural networks.
The network architecture is based on a ResNet101 [23], a deep residual convolutional neural network that with the help of skip and residual connection, is able to use more layers and exploits deeper extracted information to enhance performance of the model.
The first proposed solution introduced is the usage of a new type of convolution called, 'atrous convolution'. It highlights convolution with upsampled filters, a powerful tool in order to obtain dense predictions. Atrous convolutions allow to explicitly control the resolution of the output feature maps. The field of view of the filters is enlarged in order to incorporate larger context without increasing the number of parameters or the amount of computation. It find the best trade-off between context assimilation (large field-of-view) and accurate localization (small field-of-view).
The equation of atrous convolution is reported below:

$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k]w[k]. \tag{3.2}$$

The *rate* parameter r corresponds to the stride with which we sample the input signal. Standard convolution is a special case for rate $r = 1$, while for $r > 1$ we have the atrous convolution. Thanks to a larger receptive field is possible to use a simpler upsample process with respect to a FCN network, when we need to reconstruct the prediction resolution.

The second introduced solution is the atrous spatial pyramid pooling (ASPP). It robustly segment objects at multiple scales, by probing an incoming convolutional feature layer with filters at multiple sampling rates and effective fields-of-views, thus allow to capture objects as well as image context at multiple scales.

Finally, the last improvement is the usage of a fully connected Conditional Random Field (CRF) in order enhance prediction results. This solution improve the localization of object boundaries by combining methods from DCNNs and probabilistic graphical models, by exploiting two kernels. The first one depends on pixel value difference and pixel position difference, which is a kind of bilateral filter. Bilateral filter has the property of preserving edges. The second kernel only depends on pixel position difference, which is a Gaussian filter.
The commonly deployed combination of max-pooling and downsampling in DCNNs achieves invariance but has a toll on localization accuracy that is overcame with a

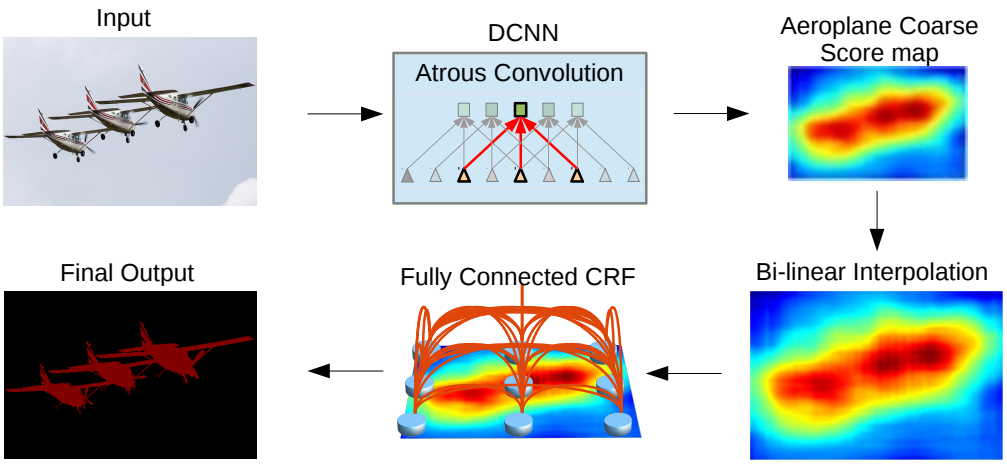


Figure 3.3: DeepLab architecture [9]

fully connected Conditional Random Field (CRF). It shows both quantitatively and qualitatively to improve localization performance.
DeepLab v2 has achieved very competitive results compared with current state-of-the-art approaches.

3.4 Metrics

A good model of Semantic Segmentation should be evaluated with different metrics. Model accuracy is probably the most important, but in real-time applications also the speed (inference-time) of the model is an important factor. However it is a more tricky evaluation because it depends on the hardware and on the experiment conditions.
Another interesting measurement is the memory requirements of the model. In fact a good model can even obtain a high level of accuracy, but if that requires too much memory usage, it can hardly be used in a real world scenario, where small devices with limited hardware are increasingly in demand.
Looking in more detail some accuracy metrics, the most used are:

- Pixel Accuracy (PA):
  Defined as the ratio of the pixels correctly classified divided the total number of pixels

- Mean Pixel Accuracy (MPA)
  Per-class mean of pixel accuracy

- Intersection over Union (IoU)
  Defined as the area of intersection between the predicted segmentation map and the original ground truth divided bu the union of both

- Mean-IoU
  Currently the most used metrics in semantic segmentation, mean of IoU over all classes

- Dice coefficient
  Defined as twice the intersection of predicted and real label map, divided by the total number of pixels in both images.

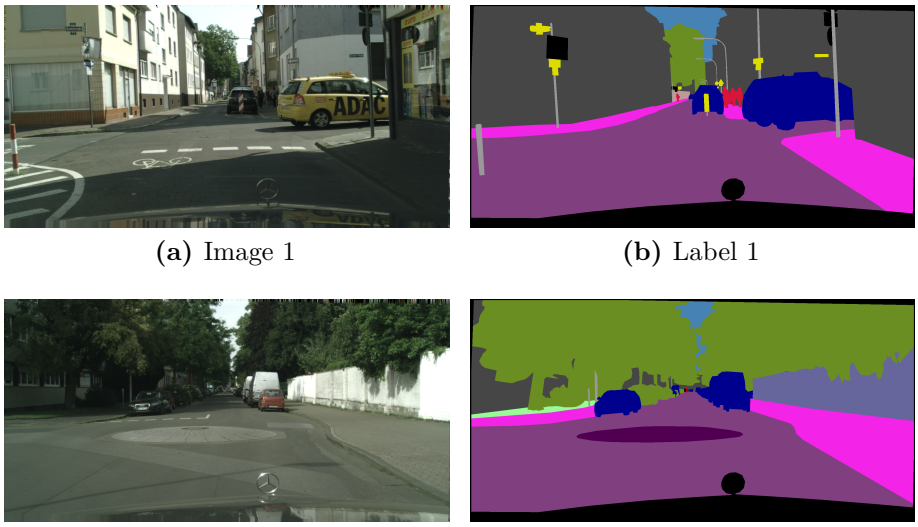
3.5 Datasets

Looking in more detail the most used dataset for semantic segmentation, the majority of them are composed of set of 2D images.
In this work the focus is on autonomous driving, for this reason we will see in more detail such datasets, but it is good to mention also other datasets that are commonly used in order to measure performance of Semantic segmentation models. These datasets are PASCAL Visual Object Classes (VOC) [24] and Microsoft Common Objects in Context (MS COCO) [25]. They are some of the most used datasets in computer vision, not just for semantic segmentation but also for object

detection and classification.
Let's see now in more detail the most used dataset for semantic segmentation oriented to autonomous driving research. The category has been divided in real or synthetic datasets.

3.5.1 Real datasets

Considering real datasets for semantic segmentation and autonomous driving, the most used in research is Cityscapes [3].
Cityscapes is a dataset that contains diverse set of stereo video sequences recorded in different cities, mainly in Germany.
It's composed of 5000 images, divided in 3000 for training, 500 for validation and 1500 for testing. Every image it's high quality pixel annotated with a set of 30 classes. The labeling process required 1.5 hours in average for person for single image.
This is also the dataset that has been used during my different experiments in this work.
Others dataset used by researchers are KITTY [26], dataset of 400 images released in 2012, CamVid [27] and Mapillary.
In order to not introduce bias in the models, there is the needs to obtain more sparse and diverse datasets. Cityscapes it's a good dataset because contains images of different cities and not a single one, however the majority of them are taken in cloudy-day environment. It would be more useful if there were multiple images containing different light conditions, for example also night images.
In my opinion, there is the needs to obtain a community image database for autonomous driving, with queries for light and weather conditions, geographical location, images resolution. Although the human segmentation process in order to obtain such labeled data it's extremely expensive, the benefits that would come from an international shared job could be incredible.
An idea that came to my mind, is the possibility of using Google Street View images, removing the need of moving a sensors equipped car around different cities. Using a good scraping and pre-processing process could be possible to obtain different street images around the world relatively for free, requiring just a labeling process.



(a) Image 1 **(b)** Label 1

(c) Image 2 **(d)** Label 2

Figure 3.4: Samples of Cityscapes dataset with corresponding semantic mask labels. [3]

3.5.2 Synthetic datasets

One of the biggest problem of real dataset is that in order to collect big and diverse data there is the need to get psychically a car, add some cameras on it, and drive the car in the streets in order to obtain some data. Moreover, the biggest problem of such approach is that such collected data needs to be fine annotated pixel per pixel. This process is very expensive for companies because it requires hours for every single image, and it's the current bottleneck of Semantic segmentation.
One possible solution could be the usage of synthetic data from simulators, where from such engine could be easily possible to change light and environment of the scene. The biggest advantage of synthetic data is that there is no more the needs to segment by hand the images, because the corresponding ground truth of an image will be created automatically by the system due the fact that the rendering engine needs to know where to draw each object on the scene.
However using virtual-world data to generalize on real world perception tasks it's not an easy task, due to the big difference that can exist between the two domains. Domain adaptation that we'll see in more detail in the next chapter, try to solve this domain shift problem.
In literature there are present different public available datasets that are used for this kind of tasks.
The most used synthetic dataset for semantic segmentation in urban scene is the

one proposed in [6]. In such dataset about 25000 images with corresponding labels has been collected from the famous realistic open-world video game *Grand Theft Auto V*, video game set in a virtual city of Los Angeles.
The class labels of the dataset are compatible with CamVid and Cityscapes real datasets.
Pixel-wise accurate labels are collected with a technique known as *detouring*, where a wrapper is injected between the game and the operating system allowing to record, modify and reproduce rendering commands, so without the need of trace boundaries of the classes by hand.
In the proposed paper, authors show that a model trained with the game data and just 1/3 of CamVid dataset outperform models trained on complete CamVid training set.
This dataset it's also the one used in my experiments of Domain Adaptation.

Another famous dataset used is SYNTHIA [28], which consists of a collection

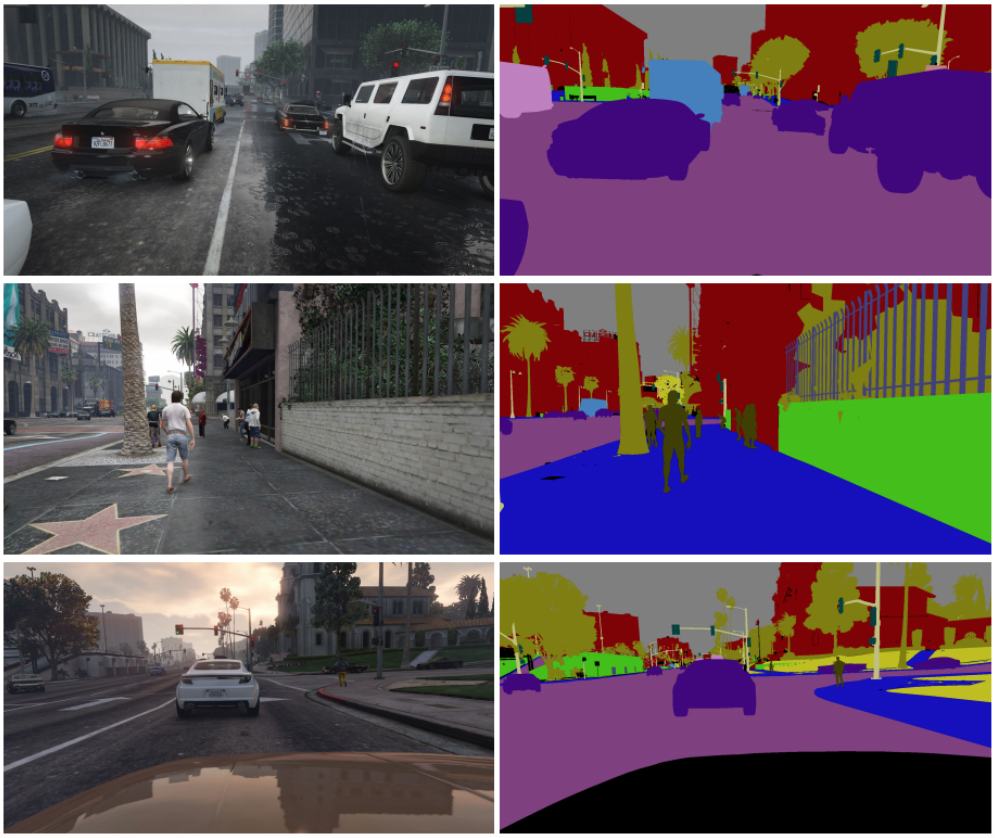


Figure 3.5: Samples of GTA 5 datasets [6]

of photo-realistic frames rendered from a virtual city, and it comes with precise pixel-level semantic annotations for 13 different classes: misc, sky, building, road,

sidewalk, fence, vegetation, etc.
Other synthetic dataset used, especially in the industrial sector, are those based on virtual simulators, similar to those used for videogame environment generation. The majority of these are based on Unreal Engine, an advanced framework and platform used in order to develop 3D dynamic scene environment.
CARLA [29] is a famous open-source simulator based on Unreal from which is possible to collect realistic synthetic data that can be then used in order to train semantic segmentation networks.
The realism of simulators is progressively growing but we are still far from the ability to create images indistinguishable from real world. The real world is to much complex that with today knowledge and techniques cannot be totally simulated.
For these reason, today, some techniques that allow a generalization and the ability to transfer knowledge from the virtual world data to the real world are required.

Chapter 4

Domain Adaptation

4.1 Introduction

In order to take advantage of synthetic data for training a model that will then used in a real scenario, there is the need to make the model able to generalize on real images. These types of problems are called Domain Adaptation problems.

Domain Adaptation (DA) is a sub-discipline of machine learning which deals in a scenario where a model trained on a source domain, is used in a different but related target domain. From Figure 4.1 it's possible to see a better understanding

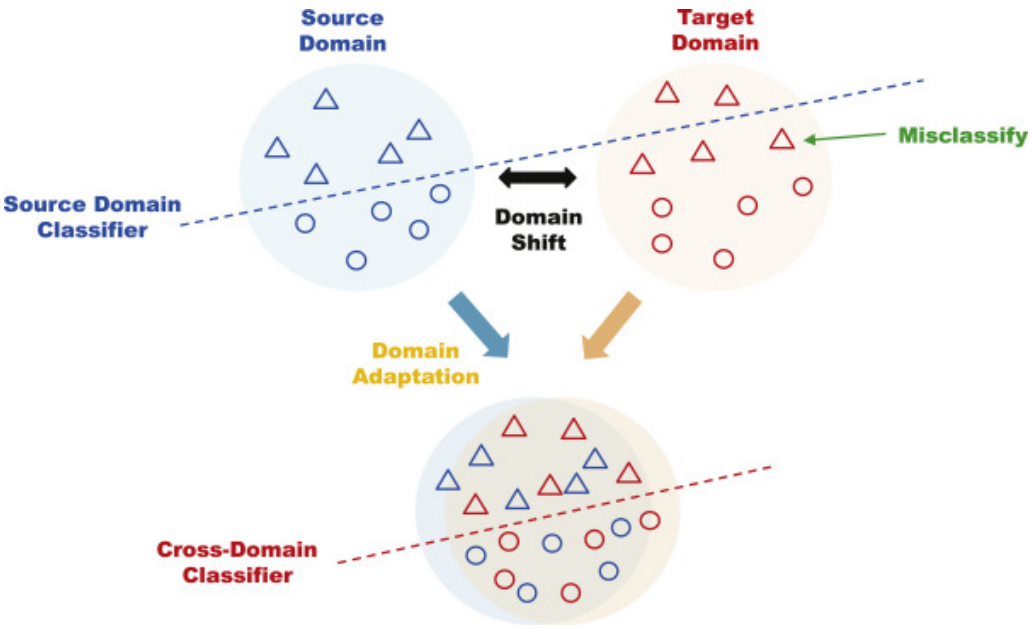


Figure 4.1: Example of domain adaptation [2]

of what does it mean Domain Adaptation.
Given for example a binary classification problem and two distributions representing respectively source and target domains, the objective function is to obtain a cross domain classifier that fits well for both domains.
The source domain classifier fits well for the source domain, the two classes are well separated, but when it is applied to the target domain some samples are misclassified, due to domain shift. This means that the distributions of the two domains are not aligned, there are some difference in the samples of the two domains, that cause inability for the source classifier to generalize on the target domain.
The goal of Domain Adaptation is to learn a Cross-Domain classifier that is able to classify well in both source and target domains.

From a mathematical point of view, using notation of [4], a domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, ..., x_n\} \in \mathcal{X}$. Given a specific domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ consists of a feature space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$, which from a probabilistic perspective can also be seen as a conditional probability distribution $P(Y|X)$. Normally the learning of $P(Y|X)$ is achieved in a supervised manner from the labeled data $\{x_i, y_i\}$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$.
Assuming two domains: the source domain $\mathcal{D}^s = \{\mathcal{X}^s, P(X)^s\}$, is used as training set with labeled data, and the target domain $\mathcal{D}^t = \{\mathcal{X}^t, P(X)^t\}$ which may be used with or without labels can be used both for training and test set. Each domain is together with its task: the former is $\mathcal{T}^s = \{\mathcal{Y}^s, P(Y^s|X^s)\}$, and the latter is $\mathcal{T}^t = \{\mathcal{Y}^t, P(Y^t|X^t)\}$. Similarly, $P(Y^s|X^s)$ can be learned from the source labeled data $\{x_i^s, y_i^s\}$, while $P(Y^t|X^t)$ can be learned from labeled target data and unlabeled data.

4.1.1 Settings of Domain Adaptation

Domain adaptation may be applied with different settings and scenarios, and it can be further divided in different categories [4].
For the case of traditional machine learning, the domains are the same $\mathcal{D}^s = \mathcal{D}^t$ and likewise the tasks $\mathcal{T}^s = \mathcal{T}^t$.
Domain Adaptation is based on the assumption that the tasks are the same, i.e., $\mathcal{T}^s = \mathcal{T}^t$, and the differences are only caused by domain divergence, $\mathcal{D}^s \neq \mathcal{D}^t$.

Based on the different domain divergences that may exist between domains, DA can be split in two main categories: homogeneous and heterogeneous DA.
In homogeneous DA the feature spaces and their dimensionality are the same,($\mathcal{X}^s = \mathcal{X}^t$), ($d^s = d^t$), and the difference are only on the data distributions ($P(X)^s \neq P(X)^t$).

In heterogeneous DA instead, the feature spaces and their dimensionality may change between source and target domains, ($\mathcal{X}^s \neq \mathcal{X}^t$), ($d^s \neq d^t$).
Based on the amount of target label available we can further categorize Domain Adaptation in:

- Supervised DA, target labels available for training
- Semi-supervised DA, both labeled and unlabeled target data
- Unsupervised DA, no target labels available for training

Domain adaptation may also be applied in One-Step, when source and target domains are directly related, or through Multi-step DA, using multiple processes and crossing intermediate domains in order to achieve the goal.
We can see a representative scheme of the different settings in 4.2

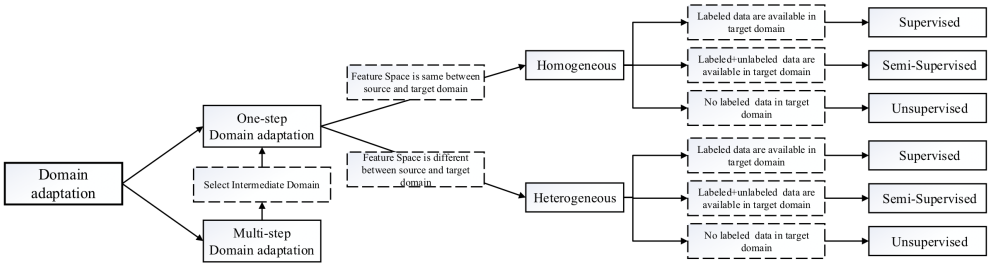


Figure 4.2: Overview of different settings of domain adaptation [4]

Finally there are three main techniques for one-step Domain Adaptation:

- Discrepancy-based
- Adversarial-based
- Reconstruction-based

Discrepancy-based DA assumes that fine-tuning the deep network model with labeled or unlabeled data can diminish the shift between the domains. It's mainly based on statistic criterion, distribution shift can be reduced through different techniques, the most used are: maximum mean discrepancy (MMD), correlation alignment (CORAL), Kullback-Leibler (KL) divergence. These techniques will not be studied in detail as they are outside the scope of this work.

Adversarial-based DA approach instead is based on the Adversarial loss of the GAN [5]. A domain discriminator tries to classify whether a data point is from the source or target domain, while the generator tries to fool the discriminator giving

it in input samples of the target domain as source ones.

Lastly Reconstruction-based DA assumes that a data reconstruction of source or target domain may improve performance of domain adaptation. Fall in this categories applications of image-to-image translation, whose become famous in the last years thanks to Generative Adversarial Networks.

4.2 Image to Image Translation

The most interesting application of Reconstruction-based Domain Adaptation is Image to Image translation. A model tries to create a shared representation between source and target domain, mapping from a domain to another.
In this section we refer on applications based only on Generative Adversarial Networks, as they're the main focus of this work, however it is good to mention that even others generative methods, such as stacked auto-encoders, or variational auto-encoders, can perform the same task, still with lower performance than GAN's.

4.2.1 Pix2Pix

Pix2Pix was introduced in 2016 by researchers at Berkeley University [11], it's the first work that produces relevant results using conditional adversarial networks as general-purpose solution to image-to-image translation problem.
Differently from traditional GAN that maps from a random noise vector z to an output image y, $G : z \to y$, conditional GANs learn a mapping from an observed image x and random noise vector z, to y, $G : \{x, z\} \to y$. In order to learn a mapping between an input image and an output image it uses a training set of aligned images pairs between the two domains.
The generator is trained in order to fool the discriminator producing images that are indistinguishable from the real ones. The discriminator instead, is trained in order to distinguish if input images are real or fake.
Therefore, it is possible to express the objective function of a conditional GAN as:

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{x,y}[\log D(x, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))],\end{aligned} \tag{4.1}$$

where G tries to minimize this loss against an adversarial D that tries to maximize it, i.e. $G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D)$. Authors propose also the use of a L1 distance loss for the generator, that provides better quality images, generating less blurring images.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]. \tag{4.2}$$

The final objective function so is:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G). \tag{4.3}$$

The generator architecture is based on an encoder-decoder network [30], in particular a U-Net model [31].

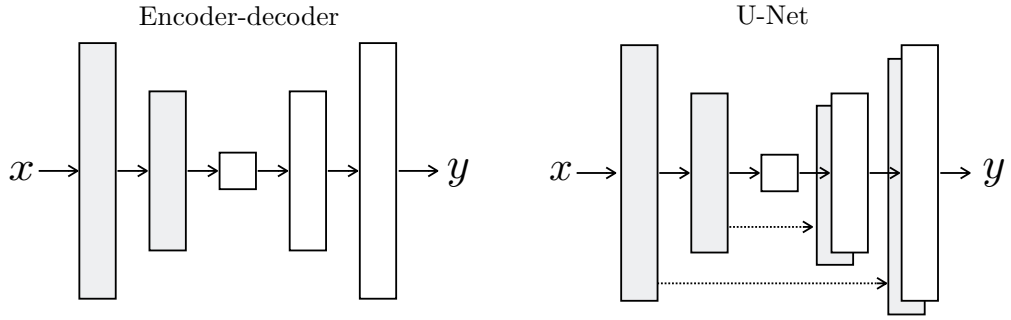


Figure 4.3: Possible architectures for the generator [11]

In this kind of network, the input is passed through a series of convolutional and pooling layers, that progressively downsample the input until a bottleneck, then the process is reversed through upsampling layers 4.3 in order to reconstruct the new image with the same dimension of the input. In the U-Net architecture are also added skip connections between layers, that provide more meaningful information in the reconstruction task of the image. Specifically, skip connections are added between each layer i and layer $n - i$, where n is the total number of layers. Each skip connection simply concatenates all channels at layer i with those at layer $n - i$. The discriminator is based on a PatchGAN architecture [32], which tries to classify different patches of NxN of the input image of D as real or fake, and then average all response to provide the output loss.

Pix2Pix may be used on a variety of tasks and datasets, even of small dimension (about hundreds), including colorization, style transfer, sketch to photo, etc. In Figure 4.4 we can see some of those applications.

Pix2Pix may also be used in a semantic segmentation task by mapping between the domain of road images to the domain of the semantic label. However the results are poor in terms of mIoU due to the fact that is minimized a reconstruction loss instead of a segmentation one.

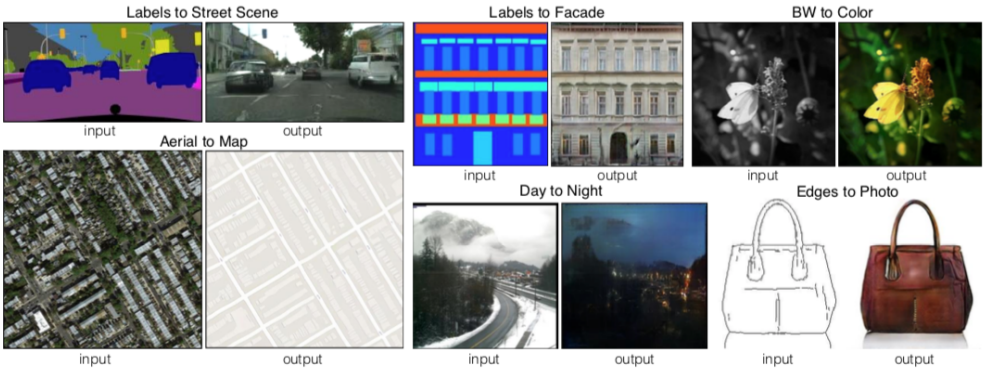


Figure 4.4: Some of the different applications of pix2pix [11]

4.2.2 CycleGAN

CycleGAN is an unsupervised method of image-to-image translation model [7], able to map between domains without the needs of a paired training set, thus allowing a more practical application. This because building aligned datasets is not an easy and fast task. Example in Figure 4.5.

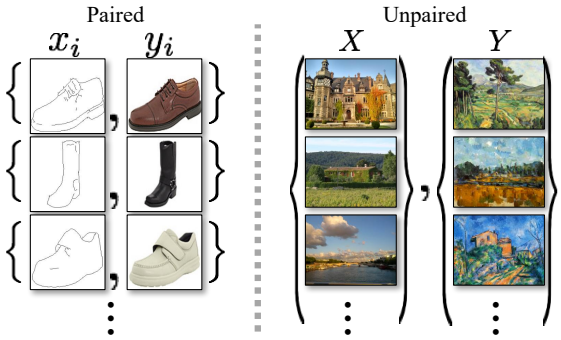


Figure 4.5: Difference between paired and unpaired datasets [7]

The model is based on two GANs, each of which deals with a specific domain. Given a set of training images in domain X and a different set in domain Y, two mapping function are learned, $G : X \rightarrow Y$ which is adversarially trained with discriminator D_Y and $F : Y \rightarrow X$ using discriminator D_X.
D_Y helps G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F.
Moreover is added a cycle consistency loss for both generators in order to further enhance the translation process. Cycle consistency born from the assumption that

when, e.g. we want to translate a sentence from Italian to English and the translate it back from English to Italian, we would like to arrive to the original sentence. In the same way we would like that $x \to G(x) \to F(G(x)) \approx x$ and vice-versa for F and domain Y.

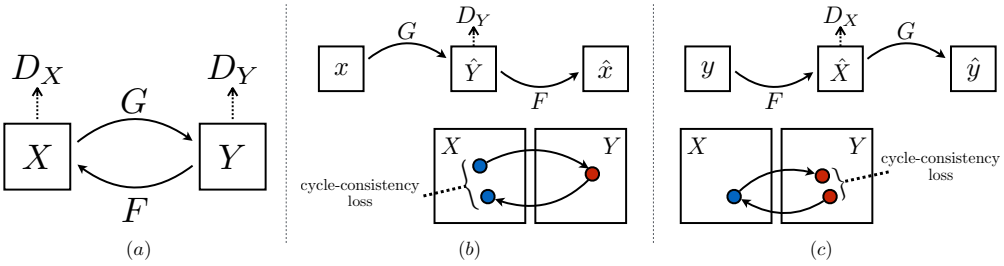


Figure 4.6: Base model of cycleGAN [7]

Therefore we can see the objective functions from a more formal point of view. For the mapping function $G : X \to Y$ and its discriminator D_Y, we can express the objective adversarial function as:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = &\mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))] \end{aligned} \tag{4.4}$$

where the task of G is to generate images from domain X to domain Y such that D_Y cannot be able to distinguish those from the real images of Y. Same adversarial loss for the opposite GAN. For the cycle-consistency loss it's used a L1 loss distance in order to enhance $x \to G(x) \to F(G(x)) \approx x$ and vice-versa.

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = &\mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]. \end{aligned} \tag{4.5}$$

Therefore the full objective is:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = &\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F) \end{aligned} \tag{4.6}$$

where λ controls the relative importance of the two objectives.
Finally, the task to solve is:

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y). \tag{4.7}$$

Networks architecture of the model are the same used in pix2pix [11].
Results of cycleGAN are very promising, the possibility of not requiring paired

datasets, guarantees more practical application than pix2pix. However the method succeed only on translation tasks that involve texture and color changes, but have poor results on tasks that require geometric translations. For this reason the two datasets of the two domains needs to be consistent and not too different in terms of geometries.

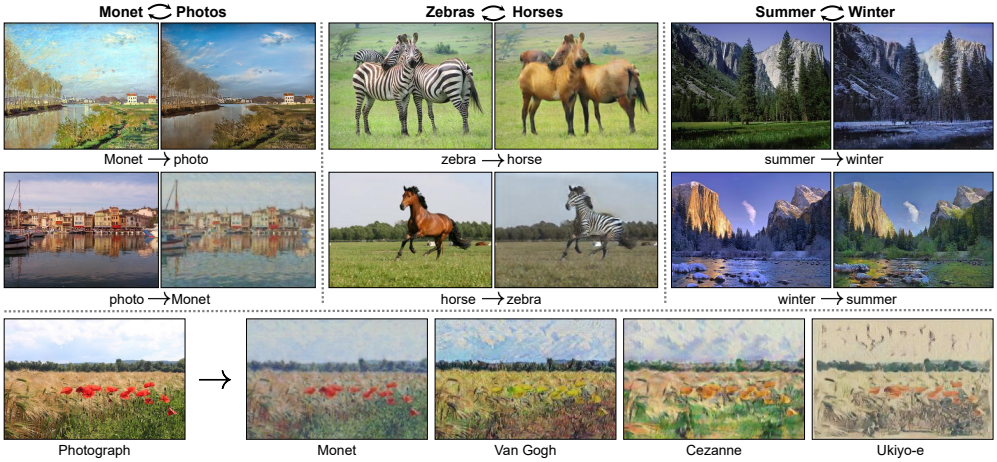


Figure 4.7: Some applications of cycleGAN model[7]

Results of cycleGAN can be further enhanced by adding a perceptual loss as done in Perceptual Adversarial Networks [33].

4.3 Domain Adaptation for Semantic Segmentation

In the context of Self-Driving car there is the needs to collect enough data in order to perform visual pattern recognition and understanding of the world around the vehicle. In previous chapters we've seen how this process is complicated and requires Semantic Segmentation algorithms in order to predict which pixel of the image frame corresponds to that specific class. Synthetic data may help in the collection of labeled data but they have the downside of not representing enough the real world.
For this reason Domain Adaptation algorithms may help the task when the amount of real labeled data is not enough to guarantees sufficient consciousness on the environment.
Therefore, synthetic labeled data may help by generalizing on a real unlabeled or semi-labeled dataset. In the literature there are different techniques that involve domain adaptation both at a pixel or a feature level.
In this section we'll see in more detail some of the techniques I've studied that have been used as a baseline in order to provide my different experiments. Every methods is used in a Unsupervised Domain Adaptation settings, so no labeled target data is used during the training of the models.

4.3.1 AdaptSegNet

This model is based on the assumption that the outputs of a segmentation model are structured and share more spatial and layout similarities between source and target domains than the input images. For this reason it will be more difficult for an adversarial discriminator to distinguish if the input given comes from source or target distribution and this will bring more meaningful gradients to the features of the generator model (encoder network) bringing a better adaptation between domains.
Therefore AdaptSegNet [34] address the pixel level adaptation problem in the output prediction space by adding a domain adaptation module in the last or middle layer of a segmentation model.

The model is composed of two parts: a segmentation network G and one or more domain adaptation module that contains the discriminators D_i with i that refer to the level. Given two domains: the source domain X_s with corresponding semantic label Y_s and the target domain X_t with no labels, the task of the domain adaptation module is to provide gradients to G in order to make it generate similar segmentation distribution for the target domain to the source domain. This is done by adversarially training discriminator and generator (segmentation network) and

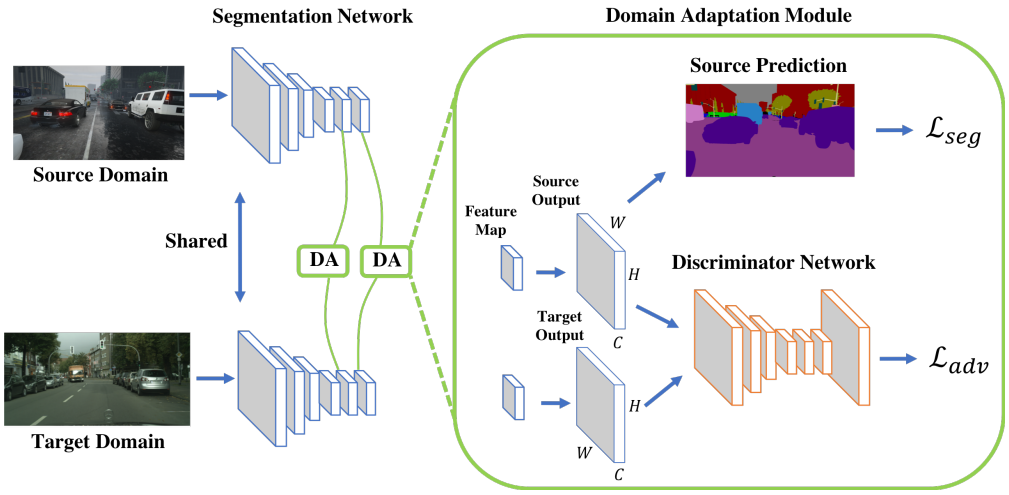


Figure 4.8: AdaptSegNet model overview [34]

by minimizing a segmentation loss for the source domain.
More formally given I_s the set of source images and I_t the set of target images, the objective function to solve will be:

$$\mathcal{L}(I_s, I_t) = \mathcal{L}_{seg}(I_s) + \lambda_{adv}\mathcal{L}_{adv}(I_t). \tag{4.8}$$

We can define $\mathcal{L}_{seg}$ as the cross-entropy loss for the source images,

$$\mathcal{L}_{seg}(I_s) = -\sum_{h,w}\sum_{c\in C} Y_s^{(h,w,c)} \log(P_s^{(h,w,c)}), \tag{4.9}$$

where Y_s is the ground truth for source images and $P_s = \mathbf{G}(I_s)$ is the segmentation output.

$\mathcal{L}_{adv}$ instead is the adversarial loss that adapts predicted segmentations of target images to the distribution of source predictions.
The target images are given in input to G that output the prediction $P_t = \mathbf{G}(I_t)$. By maximizing the probability of the target prediction being considered as source by the discriminator, it is possible to adapt the two domains.

$$\mathcal{L}_{adv}(I_t) = -\sum_{h,w} \log(\mathbf{D}(P_t)^{(h,w,1)}). \tag{4.10}$$

In order to further adapt the domains can also be possible to add additional adversarial modules that produce better results by enhancing adaptation in a lower-level feature space.

Based on (4.8), we need to optimize the following min-max criterion:

$$\max_{\mathbf{D}} \min_{\mathbf{G}} \mathcal{L}(I_s, I_t), \tag{4.11}$$

where the final goal is to minimize segmentation loss for source domain while maximizing the probability of making target predictions recognized as source.

4.3.2 CLAN

One of big disadvantage in using adversarial loss for domain adaptation is that:

> "When the generator network can perfectly fool the discriminator, it merely aligns the global marginal distribution of the features in the two domains (*i.e.*, $P(F_s) \approx P(F_t)$, where F_s and F_t denote the features of source and target domain in latent space) while ignores the local joint distribution shift, which is closely related to the semantic consistency of each category (*i.e.*, $P(F_s, Y_s) \neq P(F_t, Y_t)$, where Y_s and Y_t denote the categories of the features)."

This as the effect that can provide a negative transfer if the features are already aligned before the adaptation.
Category-Level Adversarial Network (CLAN) [8] uses a weighted-adversarial loss which is used to weight more the adversarial loss if the features of a specific class are distributed differently between source and target domains, and weight less the loss if the features are already well aligned between the two domains, leading to a category-level joint distribution alignment.
In Figure 4.9 we can see in more detail this problem of domain adaptation based on adversarial training.

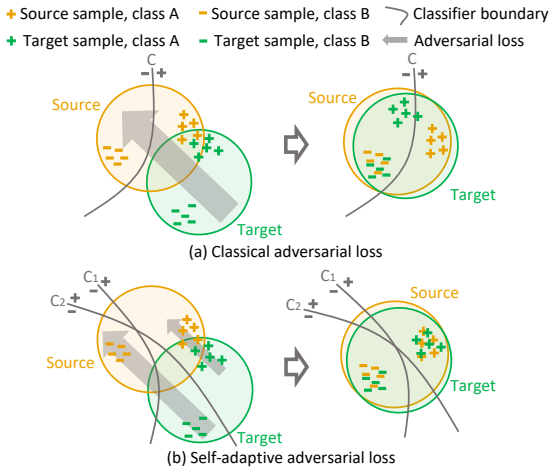


Figure 4.9: Self-adaptive adversarial loss [8]

This is achieved by using the co-training approach [35], that consist in using two classifiers to predict how well each feature is semantically aligned between source

and target domains.
More formally, the semantic segmentation network that represents the generator G of the adversarial game, is divided into a feature extractor E and two classifiers C_1 and C_2.
The task of E is to extract features from the input images, while the tasks of C_1 and C_2 are to predict the features extracted by E into one of the defined classes.
In order to provide different views of the classifiers we need to enforce the weights of C_1 and C_2 to be consistent but diverse through a cosine similarity distance.
The final prediction will be an ensemble prediction by the sum of the predictions of each classifier.

CLAN model follows the work of AdaptSegNet [34] by adapting output predictions, adding the contribute of a weighted adversarial loss that allow a category-level feature alignment between source and target domain.
For this reason, given source domain images with ground truth labels X_s, Y_s and a set of target data X_t without labels, the goal is to allow the segmentation network to correctly predict semantic segmentations for target images.
The network training is based on three loss functions, segmentation loss, weight discrepancy loss and self-adaptive adversarial loss.

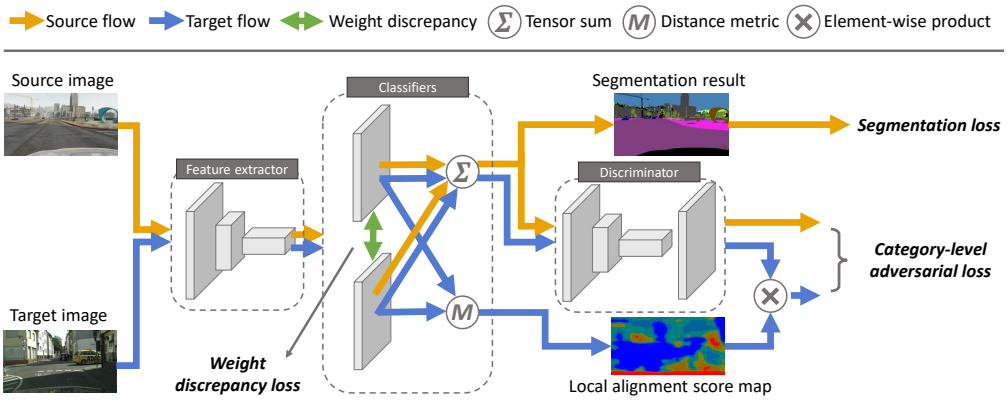


Figure 4.10: Category-Level Adversarial Network architecture [8]

On the source images is calculated a segmentation loss as in equation 4.9, that represents a multi-class cross-entropy loss between the sum of the predictions of C_1 and C_2, with the ground truth Y_s.
In order to allow the two classifier C_1 and C_2 to provide different views on a feature we need to minimize the cosine similarity of the weights of the convolutional filters of each classifier. The weight discrepancy loss is defined as:

$$\mathcal{L}_{weight}(G) = \frac{\vec{w_1}\vec{w_2}}{\|\vec{w_1}\|\|\vec{w_2}\|} \tag{4.12}$$

where $\vec{w_1}$ and $\vec{w_2}$ are weights of the convolution filters of C_1 and C_2 flattened and concatenated.
Finally, in order to provide a weighted adversarial loss we can use the discrepancy between the prediction of C_1 and C_2 as reference $p^{(1)}$ and $p^{(2)}$.
For each target images x_t is calculated a discrepancy map $\mathcal{M}(p^{(1)}, p^{(2)})$ with the cosine distance element-wise out of $p^{(1)}$ and $p^{(2)}$.
The final adversarial adapted loss function will be:

$$\begin{aligned}\mathcal{L}_{adv}(G, D) = -E[\log(D(G(X_S)))]- \\ E[(\lambda_{local}\mathcal{M}(p^{(1)}, p^{(2)}) + \epsilon)\log(1 - D(G(X_T)))],\end{aligned} \tag{4.13}$$

When $\mathcal{M}(p^{(1)}, p^{(2)})$ is small it means that there is an overlap across the joint distribution of the domains, this means that we don't need to weight much the loss. Otherwise if $\mathcal{M}(p^{(1)}, p^{(2)})$ is large, it means that the feature maps of a class are not aligned, so there are difference in the joint distribution of the two domains and we need to give a higher weight to the loss.
The local alignment score map will be multiplied element-wise with the adversarial loss map for target samples in order to provide an adaptive loss.

The final objective function that needs to converge by optimizing G and D will be:

$$\begin{aligned}\mathcal{L}_{CLAN}(G, D) =&\mathcal{L}_{seg}(G) + \lambda_{weight}\mathcal{L}_{weight}(G) + \\ &\lambda_{adv}\mathcal{L}_{adv}(G, D)\ ,\end{aligned} \tag{4.14}$$

where λ_{weight} and λ_{adv} are the hyperparameters that control the weight of each loss.

The network and the code of CLAN, as said before, are based on AdaptSegNet, for this reason they used the same model and parameters.
In table 4.1 it's possible to see in more detail some results of this method compared to the AdaptSegNet and a model trained only on source data without domain adaptation. Results presented is the mean IoU obtained using DeepLab v2 (ResNet101) as a backbone trained on the dataset of GTA 5 as source and tested on Cityscapes dataset as target one.

Table 4.1: CLAN and AdaptSegNet results

GTA5 → Cityscapes		
	mIoU	**gain**
Source only	36.6	—
AdaptSegNet [34]	41.4	4.8
CLAN [8]	**43.2**	**6.6**

4.3.3 Self-Supervision tasks

I explained earlier that computer vision tasks require to associate to raw data some labels in order to train models that perform task like classification or semantic segmentation. Considering in more detail the task of semantic segmentation, in order to train a CNN to perform this task will require a complex labeling task, due to the fact that for each class that we want to consider (*e.g.* cars, road, peoples, ...) a human needs to delineate the border pixels of each of them in the image.
This process is extremely tedious and labour intensive and research topics that try to reduce or completely remove this human labeling process are fully active and hot nowadays.

Self-supervised learning consist in using simple auxiliary tasks also knows as *pretext task* in order to avoid the needs of a manual labeling process by learning some visual models [36].
This process could be particularly interesting when applied to semantic segmentation, where the labeling task required is extremely expensive.
Some Convolutional Neural Networks are trained in order to learn specific transformation applied to original data such as image rotations or jig-saw puzzle [37]. By leveraging on the features extracted by a task specific model, a pretext CNN can produce new features that will be useful for the generalization of the main task. [36]
Even if the labels of the main task are very low, by using a Self-Supervised task it's possible to use all the labels that we want, for this reason could be possible to use extract all the information contained in the original data.

In the paper [10] has been explored how Self-Supervision techniques may be applied to Domain adaptation, of particular interest for this work is its application to Semantic Segmentation.
They designed a framework in order to jointly train a pretext and main task in order to perform Unsupervised Domain Adaptation.

The main task will be trained only with the labels of the source domain and the pretext task with self target labels will learn domain invariant features that will make the main task able to generalize on the target domain.

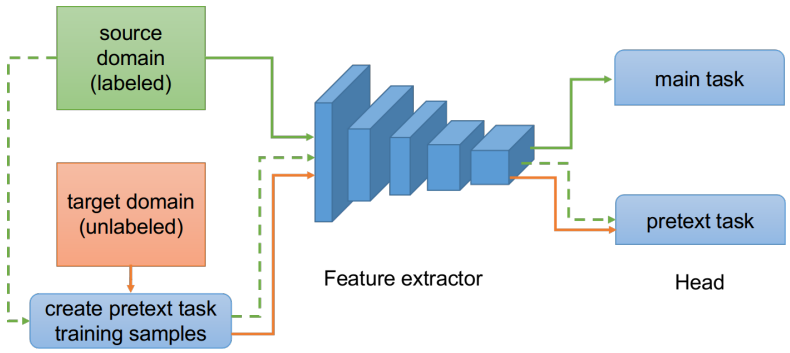


Figure 4.11: Framework for self-supervised domain adaptation [10]

Considering semantic segmentation as main task and a general pretext task let's see in more detail how Self-supervision may help the generalization of the main task.
Given a set of training data from the source domain X_s, Y_s, we can train a CNN composed of a feature extractor E and a decoder network D, in order to predict segmentation outcomes. The encoder network E is shared with a network P added in order to solve the pretext task, which is trained with X_t target data and Y_t self labels.
Both source and target domain samples are forwarded through E, then source extracted features are used in order to perform a Segmentation loss $\mathcal{L}_{seg}$ with ground truth Y_s, while target extracted features are passed to P with self labels in order to calculate the task loss $\mathcal{L}_p$. By minimizing both losses the weights of the shared encoder E are updated so that it can learn domain invariant feature representations from both domains.
During test phase the target domain images are forwarded to E and D in order to exploit invariant features learned and obtain a semantic segmentation outcome, hopefully similar to its hypothetical real label.

A pretext task that has been studied in [10] is a rotation algorithm inspired by [36].
Given a set of images from target domain X_t and a set of geometric transformations, in particular image rotations of [0, 90, 180, 270] degrees. Samples of X_t are forwarded to the encoder E, and then the feature maps extracted are given in input to P after having been rotated into one of the possible outcome rotations.
The task of P is to predict the rotation that has been applied to the feature map in input, by giving in output a probability distribution over all the possible

transformations. The loss function that needs to be minimized is:

$$\mathcal{L}_p = -\frac{1}{4}\sum_{r=0}^{3}\log(P(E(g(\mathbf{x}^t, r), \boldsymbol{\theta_e}), \boldsymbol{\theta_p})). \tag{4.15}$$

where $E(g(x^t, r), \theta_e)$ is the feature map extracted by E of parameters θ_e from input image x^t and rotated by function g by r*90 degrees, where $r \in [0, 3]$.
The CNN by learning image orientation is able to localize salient objects in the image and their orientations. This can provide a semantic information to the encoder about the target images, improving its cross-domain features representations power and enhancing domain adaptation.

Self-supervised learning can even be used together with adversarial based domain adaptation as done in AdaptSegNet [34].
This adds complexity to the full network because there will be three networks to train: a semantic-segmentation network as the main task, a discriminator in order to distinguish real from synthetic feature maps that will allow to align the domains, and finally a pretext network to solve a self-supervision task.
Authors of [10] provide several results based on different experiments done on this context.
Results of the paper shows that using adversarial loss together with rotation task and batch normalization provide the best results of 43.3 of mIoU by using as source domain GTA 5 dataset and Cityscapes as target domain.

Chapter 5

Proposed Algorithm

In this section, I will explain the algorithm proposed, the design decisions made and the different possible experiments that could be interesting to investigate in the future to further improve domain adaptation results between synthetic and real data.

5.1 Introduction

As we seen in the previous chapter, collecting and manually annotating large datasets with dense pixel-level labels is extremely costly for companies due to the large amount of human effort that is required. One possible solution is the usage of synthetic datasets where data and labels comes relatively from free. However, due to the big difference that may exist between real and virtual domain, a model trained only on synthetic data may not generalize on the real domain.
Domain adaptation tries to address this domain shift, initially by placing itself in the worst case, where target labels are not available during training. Problem that refers as Unsupervised Domain Adaptation.
After a deep study of the literature, I decided to come up with a solution that provides a mix of the techniques that I consider really promising and interesting, in order to try to further reduce the gap between synthetic and real data for semantic segmentation, so trying to achieve competitive results with the relative papers in terms of mean intersection-over-union on target data.
In the literature the most part of the benchmarks is based on the dataset of GTA 5, that is used as synthetic dataset (*source domain*) [6] and Cityscapes dataset used as real dataset (*target domain*) [3], both were deeply explained Section 3.5. Therefore I decided to use the same datasets as baseline for my experiments.

The proposed method is based on a two-steps Domain adaptation process, likewise to what was done in [38].
The first step is a pixel-level adaptation method based on a style-transfer. The idea is based on the assumption that synthetic dataset and real dataset are very different at first, on colors, textures and brightness.
For this reason, in order to train a segmentation network on synthetic data that will be able to generalize on target data providing a feature level adaptation, a good idea is to make firstly synthetic images look like to real images, such as a preprocessing step, by applying a sort of filter that makes the distribution of the pixels of the synthetic domain close to the distribution of the pixels of the target domain.
The style transfer is made by using an unsupervised image-to-image translation model, in order to map from the domain of synthetic data to the domain of real data, producing a new adapted dataset similar to the real one, that will be used for the next step of domain adaptation with the original synthetic label.
Using such adapted dataset together with unlabeled target images, it could be possible to further reduce the domain shift by applying a feature-level adaptation technique by training the segmentation network such as [34], [8] or [10], which by applying a discriminator or an auxiliary task of the top of the features extracted may help the generalization.
Different experiments have been produced, both in an Unsupervised Domain Adaptation scenario and in a Semi-supervised scenario. In a first moment, it was decided to use a subset of GTA 5 dataset, for example 2000 or 10000 images. However, by sampling randomly such subset, it is possible to introduce bias in the data, because the class distributions inside the dataset may change.

5.2 Pixel-level Adaptation

The first step of the proposed method is based on a pixel-level adaptation technique. Looking into more detail to synthetic and real images, it is possible to notice that the two domains are very different in principle.
Images of GTA 5 datasets [6] are bright and have light colors. Even if the game has a high level of realism, textures of the objects make it easily recognizable to human eye that they are from the synthetic domain. Real images of Cityscapes instead are darker and contain cold colors, with different textures from the synthetic dataset.
Therefore, before training a semantic segmentation network with a domain adaptation module at a feature-level that will tries to adapt the features of source and target, in order to reduce the domain shift, it is possible to create an adapted source dataset, based on the synthetic domain but similar to the real one, which

has already the advantage of having the domain shift highly reduced.
In this way, it is possible to exploit in a second step, a feature-level adaptation technique to further enhance the adaptation, producing at the end better results.
The pixel level adaptation technique is based on an unpaired Image-to-Image translation model (Chapter 4.2).
The idea is to learn a mapping from the domain of synthetic data to the domain of real data, in order to produce a new adapted dataset in which the basic structure of the single image does not change, but brightness and colors change to look like real data style.
In order to achieve this task, it is possible to use a Style transfer model based on Generative Adversarial Networks, such as cycleGAN [7], considering that in recent years it has been demonstrated that these methods allow to get the best results.
A discriminator is trained in order to distinguish if the image that is given in input to it, it is from source or target domain, while a generator is trained in order to fool the discriminator by giving it in input an image of the source domain marking it as a target one. In this way, after a training phase, the generator will be able to generate starting from a synthetic image, an adapted image indistinguishable from a real one for the discriminator.
If the model used is a cycleGAN, there is an opposite GAN that maps in the other side, or rather from target to real, in order to further enhance the results, as we saw in the previous chapter.
A trained generator will then be used in order to create an adapted dataset from GTA to Cityscapes.
In order to train a cycleGAN model, there is the need to resize the datasets, because such networks need images of squared resolution, while GTA V and Cityscapes are not. In [38] authors propose a solution that is similar to the one proposed here, but they used a joint training of both pixel and features adaptation phases, in the paper they chose to take a random crop of 452x452 from original datasets in order to train GAN's models. The network is trained for 20 epochs and the parameters are the ones proposed in the original paper of cycleGAN [7]. Where for the generator, that is a U-Net with 9 blocks, and the discriminator, which is the standard 70x70 PatchGAN of the cycleGAN, a learning rate of 0.0002 is used with Adam optimizer and a batch size always set to 1.
Moreover, authors decide to enhance the performance of the translation model by adding a perceptual loss, as done in [33].
During my experiments, I decided to use the adapted dataset that was provided on the *github* link of [38], because they achieved visibly better results than me.
In figure 5.1 it is possible to understand better the idea behind this pixel-level adaptation process.

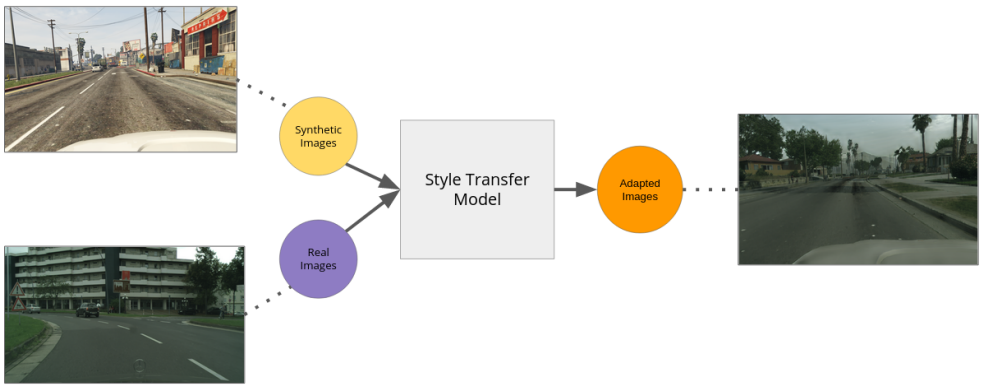


Figure 5.1: Proposed Pixel-level Adaptation

5.3 Feature-level Adaptation

In order to transfer knowledge from synthetic to real data, there is the need to train a semantic segmentation network, that without any domain adaptation techniques, does not allow a good generalization on real data due to domain shift. In literature it has been demonstrated that, training segmentation networks together with a Discriminator or an Auxiliary task, provide benefits in terms of domain adaptation [34] [10], allowing a features generalization from source to target domain.
After a deep study of the literature, I decided to use a Category-Level Adversarial Network (CLAN) as base model for my different experiments. I chose to use this network because it is an evolution of AdaptSegNet, which is one of the first network proposing feature-level adaptation and on which the majority of the new techniques of domain adaptation for semantic segmentation are based.
This model, as we saw in Section 4.3.2, is based on co-training approach, that exploiting the usage of two classifiers at the end of the encoder network together with a discriminator, can obtain a class level joint distribution alignment between source and target domain. Summing the two predictions, it is possible to get an ensemble prediction, that at least in theory could be more accurate than the case of using only a single classifier. Otherwise calculating a distance metric between the two predictions, a local alignment scope map is obtained, that will be used in order to weight the adversarial loss.
Moreover, also self-supervised learning as we saw in the previous chapter, can help features generalization, by adding an auxiliary net on the top of the output of the segmentation network with the objective of solving some specific task, such as guess an impressed rotation [10], leading to a spatial invariant features representation on the encoder network. Since the work proposed from authors of the paper is

based on AdaptSegNet, it could be interesting to see the effect of a self-supervised learning task on top of a model based on co-training approach.
Both CLAN and Self-supervised task for Semantic Segmentation provide their experiments using standard GTA V datasets as source domain. However, it has not been tried to use such networks using an adapted source dataset, which may further enhance the domain adaptation performance.
For these reasons, I decided to implement a model based on CLAN on which an auxiliary net for self-supervised task has been added, calling it Self-CLAN. The final objective is to understand in more detail the effect of each module and the advantage that comes from using a source domain that has been styled with unsupervised target images.
In figure 5.2, it is possible to see a scheme of the model proposed. Orange arrow points the flow of source data, violet stands for target and green for both sources and targets.

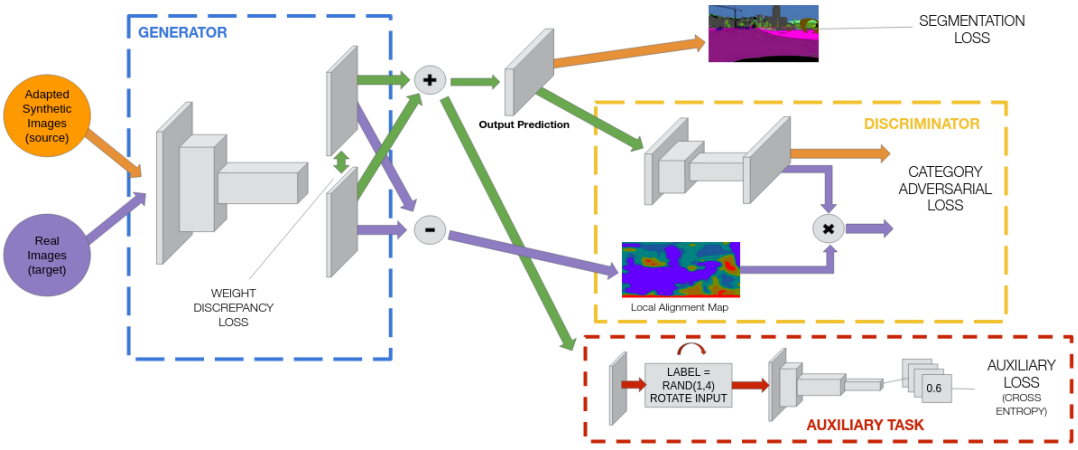


Figure 5.2: Proposed Network Self-CLAN based on CLAN [8] and [10]

In case of using standard CLAN in an unsupervised setting, recapping the objective function, given source domain images with corresponding pixel annotated labels, and given target domain unlabeled images:

$$\begin{aligned}\mathcal{L}_{CLAN}(G,D) =& \mathcal{L}_{seg}(G) + \lambda_{weight}\mathcal{L}_{weight}(G) + \\ & \lambda_{adv}\mathcal{L}_{adv}(G,D)\ ,\end{aligned} \tag{5.1}$$

where G is the generator (DeepLab network), D the discriminator, $\mathcal{L}_{seg}$ is the segmentation loss on source data, $\mathcal{L}_{weight}$ equation 4.12 and $\mathcal{L}_{adv}$ the weighted adversarial loss presented at 4.13.

In case of using the auxiliary task alone, the objective function will change in:

$$\begin{aligned}\mathcal{L}_{SELF}(G, A) =& \mathcal{L}_{seg}(G) + \lambda_{weight}\mathcal{L}_{weight}(G) + \\ & \lambda_{rot}\mathcal{L}_{rot}(A) \ ,\end{aligned} \tag{5.2}$$

where in this case A is the auxiliary network and $\mathcal{L}_{rot}$ is a multi-class cross entropy loss referred to 4.15 for the self-supervised task of rotation on the output target prediction of the segmentation network. During different experiments has been tried to perform the self-supervised task of rotation not only on target predictions as presented in [10] but also on source one. The results obtained are substantially identical.

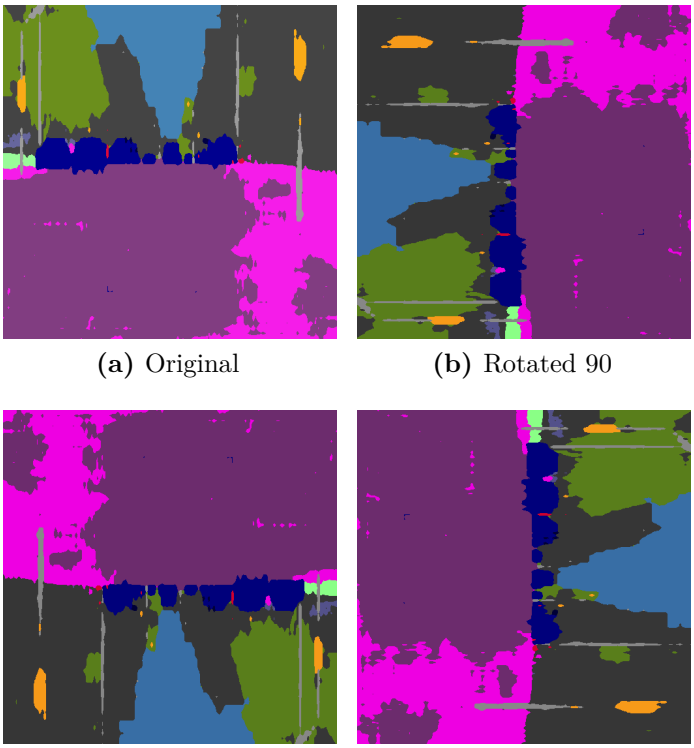
Finally in the case of using both adversarial losses and self-supervised tasks, the final objective function will be:

$$\begin{aligned}\mathcal{L}_{SELF-CLAN}(G, D, A) =& \mathcal{L}_{seg}(G) + \lambda_{weight}\mathcal{L}_{weight}(G) + \\ & \lambda_{adv}\mathcal{L}_{adv}(G, D) + \lambda_{rot}\mathcal{L}_{rot}(A) \ .\end{aligned} \tag{5.3}$$

Experiments have also been proposed in a semi-supervised scenario, in which a segmentation loss is calculated also for a fraction of target data. In such case, the objective function changes by adding another segmentation loss $L_{segT}(G)$, but referred to target data. In this case, a lambda multiplier for the different segmentations loss has not been used (both equal to 1), even if probably using a higher multiplier for target segmentation loss than the source one, may lead to better results.

Looking in more detail the network architectures, the segmentation model (G) is based on DeepLab v2, which in turn is based on a ResNet101. During all the experiments, a version of DeepLab pre-trained on ImageNet has been used.
The discriminator (D) is a binary classifier CNN-based with a fully-convolutional output, composed of five convolutional layers (filter size: 4, stride: 2, padding: 1), each of one is followed by a leaky-ReLu activation function with 0.2 slope, except for the last layer.
The auxiliary network (A) instead is based on a standard ResNet18, but experiments have been tried also with AlexNet. Input tensors to the auxiliary network are resized to a resolution of 256x256.
Optimizer used by the networks in order to update weights during training are: Stochastic Gradient Descent for generator (learning rate: 2.5e-4, weight decay: 0.0005, momentum: 0.9) and auxiliary network (learning rate: 0.001, weight decay: 0.0005, momentum: 0.9), while for the discriminator Adam optimizer was used (learning rate: 0.0001, weight decay: 0.0005). Learning rates are polynomially decreased during training.

Standard values for lambda loss multiplier, as proposed in relative papers, have been set to: $\lambda_{weight} = 0.01$, $\lambda_{adv} = 0.001$, $\lambda_{rot} = 1$, λ_{seg}=$\lambda_{segT} = 1$. Training iterations have been set to 48'000 and batch size equal to 1 due to computational requirements.
Resolution of images of the GTA V dataset is 1280x720 while for Cityscapes 1024x512.
Different experiments have been performed in order to establish the relevance of each module, based on different settings and parameters.



(a) Original **(b)** Rotated 90

(c) Rotated 180 **(d)** Rotated 270

Figure 5.3: Examples of prediction maps rotated, input for the Self-supervised network

Chapter 6

Results

In this section I will present the results obtained from the different experiments with the proposed algorithm.
All the different results proposed are based on the best value of mean intersection-over-union performed on the validation set of Cityscapes (target domain) calculated every 2000 or 4000 iterations of training. Two different settings have been defined, firstly an unsupervised domain adaptation approach, in which the objective is to obtain the maximum value of mIoU on the target validation set of the real data without using any real label. Finally, some experiments have been performed in a Semi-supervised setting, in which a fraction of real data has been used. The aim was to understand if could be possible to achieve competitive results using a less amount of real data annotations and synthetic data, with respect to a model trained only on real data.
Due to computational requirements and the huge amount of different parameters setting, every experiment has been validated only one single time, although for more truthful results there is the need to perform more runs proving mean and standard deviation on the results obtained.
The deep learning framework used was pytorch, using python as programming language.

6.1 Unsupervised Domain Adaptation

In order to provide comparison metrics, are required some baseline values. The first baseline is the best value of mIoU calculated on the validation set of Cityscapes by training the base segmentation model (DeepLab v2) with standard parameters, on the source dataset without any techniques of domain adaptation. During training has been minimized the multi-class cross entropy loss for the segmentation loss on GTA V images and the weight discrepancy loss in order to exploit the co-training

approach. Has been obtained a value of **37.96** mIoU.
The second baseline is obtained by training the segmentation model only on target data, so on the training set of Cityscapes. Has been reached a value of **64,78** mIoU.

Baselines	
	mIoU
Source	37.96
Target	64,78

From those two base results it's possible to easily understand the problem of domain shift. If we train the same model on two different but related dataset get very different results when then it is tested on the validation set of the real domain. This happens because the two domains are different, contains different object in different locations and the colours and light of the datasets is different.
For this reason domain adaptation techniques try to reduce this gap, with the aim to get as close as possible to the target result, using the less amount as possible of target labels, due to their expensive labeling cost.

The first relevant experiment is about using as source training data, a dataset of GTA 5 adapted with a CycleGAN trained in order to apply style of Cityscapes, as proposed in Pixel-level adaptation paragraph.
Training DeepLab v2 with such adapted dataset, get very interisting and competitive results, achieving a mean intersection-over-union of **45,57**. Adapting images with a style transfer technique lead to obtain a gain of 20% in performance with respect to results on the original dataset of GTA. This result shows how relevant are the pixels of the images in input to the model. Even if the labels used are the same, a great result is obtained by only changing colors and brightness of the images so that they could look like to the real world data style.
Has also been proposed an additional experiment using a limited set of Cityscapes images in order to create a new adapted dataset. A cycleGAN was trained using as source dataset the GTA 5 original dataset and as target dataset only 150 samples (5%) of Cityscapes. The subset of 150 images has been sampled randomly, however I realized later that this is not the best way to do that. Inside of the 150 images in fact is present many vegetation, especially in the upper parts of the images, however this is not the same for GTA samples. So, what happens is that by mapping from the real to the virtual domain, trees are created in the upper part of the image where there should be only sky. Furthermore, the images do not have a good quality compared to those that were used in other experiments.
A better idea that was recommended later to me from my company tutor, was to

sample these 150 images uniformly from the various clusters that can be formed by reducing the dimensionality of the dataset using a non-linear algorithm, so as to try to obtain a subset of samples which is quite representative of the entire Cityscapes distribution.
An other problem may be that the cycleGAN is unable to converge with such unbalanced domains (25000 vs 150) or that there is a need for better hyperparameters tuning.
The cycleGAN training took about 4 days so was difficult to carry out further experiments due to deadlines.
Surprisingly, by training DeepLab v2 on this so adapted dataset without further target images, has been obtained relatively good results when then the model was tested on Cityscapes, although the images were visibly problematic. It is possible to see some failure cases in Figure 6.10. The maximum value of mIoU reached was **42.6**. Showing that with even a limited set of target images is possible to increase the level of generalization, demonstrating effectiveness of pixel level adaptation methods.

Moving to feature-level adaptation is followed the architecture proposed in CLAN [8]. Several experiments have been performed by adding a discriminator on top of the predictions of the segmentation model, exploiting the category level adversarial training in order to make filters of the encoder network able to generalize on target data.
First of all, was used the original dataset of GTA V in order to achieve comparative results to those published in the reference paper [8]. Using parameters proposed by the authors has been reached a value of mIoU of 41,82. This value is less than the one proposed in the paper (43.2) because the training time has been reduced to 48'000 steps instead of 100'000 as done in the paper, due to computational requirement, as told before.
In others test has been tried different parameters, for example changing the λ_{adv} multiplier to 0.01 or using a higher learning rate, but results obtained were too low and the training were stopped in order to give space to more significant trials.
Training the Category-level Adversarial Network with the adapted dataset of GTA instead leads to much more interesting results. In table 6.1 are reported the values of mIoU reached by trying different values of λ_{adv}. Best score is obtained with λ_{adv}=0.001, leading to **46.17** of mIoU. λ_{adv} controls the relevance of the adversarial loss in the total objective function.

Let's now dive in Self-supervised experiments. In the paper that proposes using a rotation algorithm on the top of the prediction of the encoder as auxiliary task, authors reached a mIoU value of 41.2 [36]. During my trials instead, using the original GTA V dataset I reached a value of **42.63**, leading to a 12.30% gain with

Table 6.1: Different results by changing λ_{adv} on CLAN trained with adapted GTA as source

Adversarial loss multiplier	
	mIoU
$\lambda_{adv} = 0.001$	**46.17**
$\lambda_{adv} = 0.01$	45.13
$\lambda_{adv} = 0.1$	45.53
$\lambda_{adv} = 1$	45.68

respect to train the semantic segmentation network without domain adaptation. This difference in the results may exist due to the difference base model used. In fact in the paper is used a standard DeepLab v2, while in my trial is used a version of the same model based on co-training approach, so using two prediction of two classifier as an ensemble prediction. Moreover, authors did not provide code for the self-supervised task for semantic segmentation so the implementation of the algorithm may be different from the one used here.
One of the problem of this solution is that during training the auxiliary model, it learns very quickly to distinguish the rotation impressed to the prediction map. For this reason the self-supervision loss provide meaningful gradients only in the early iteration steps, converging to zero after a short period of time.
Some possible solutions to this problem may be to make auxiliary task more difficult or using an easier auxiliary model. In order to make a more difficult task for the auxiliary network, has been tried to change the size of the feature map in input. Standard values were (256,256,19) where the first two values are width and height of the map and the last value indicates the classes present in the prediction. So have been tried to resize the feature maps to a size of 400x400 and 512x512, but results did not change too much.
Using on the contrary a different network, an AlexNet instead of a ResNet18, performance degraded not leading to better results, obtaining a mIoU of 40.86 .
Also in this case has been tried to change network parameters such as learning rate and lambda loss multiplier, but performance were to low and also in this case training were stopped.
Training the self-supervised model with the cycleGAN adapted dataset as before, leads to better results and boost in performance. Using standard parameters with such source dataset, as proposed in the previous chapter, is obtained a value of **45.49** of mIoU. In this case, employing a resize on the input feature map of the auxiliary network to 400x400 resolution, is reached a score of 45.51 mIoU.
Using both an AlexNet and a higher input resolution however results are lower,

with only a score of 44.72. This shows that using a ResNet18 as auxiliary network is a better option than using an AlexNet for this self-supervised trial.

Finally has been tried experiments using both discriminator and auxiliary network. In the paper of Self-supervision task for semantic segmentation [10] has been tried also this combination. They obtained a boost of performance leading to a score of 42.3, however I did not obtained the same results.
In fact by training the segmentation network with both weighted adversarial loss and self-supervision loss I reached a value of 39.64 mIoU, using standard GTA 5 dataset.
By using the adapted dataset results did not change too much, obtaining a score of 45.32 .
This discrepancy in the results could be due to different implementation of the algorithm or different hyperparameters used during training. Training 3 different networks for the same model is a very heavy computational task, for this reason the number of iterations that needs to be done may be higher.
Looking in more detail the graphics below is possible to see a recap of the just exposed results. In figure 6.1 we can see the different reference results obtained without feature-level domain adaptation.

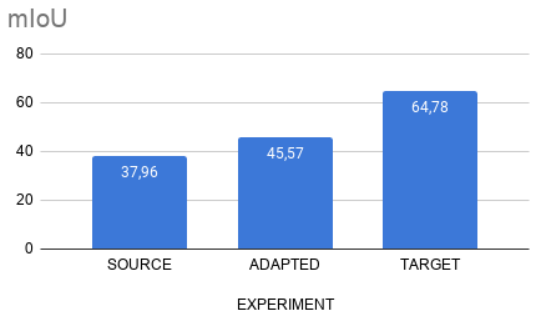


Figure 6.1: DeepLab v2 without Feature-level Adaptation

In figure 6.2 instead we can see better the different results obtained with feature-level domain adaptation.

Some conclusions can be drawn from an analysis of the results obtained.
Feature-level domain adaptation performs very when the initial distribution of source and target are far. Adversarial learning with CLAN performs a +10% gain and Self-supervision with the rotation algorithm a +12% with respect to a model trained only on source data without any technique of domain adaptation. While the combination of both methods did not produce improvements during my trials

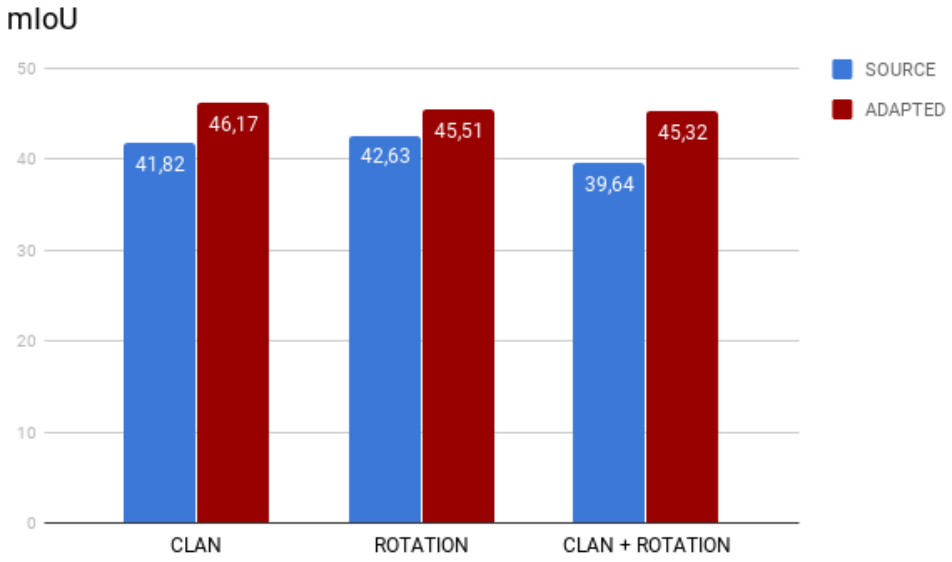
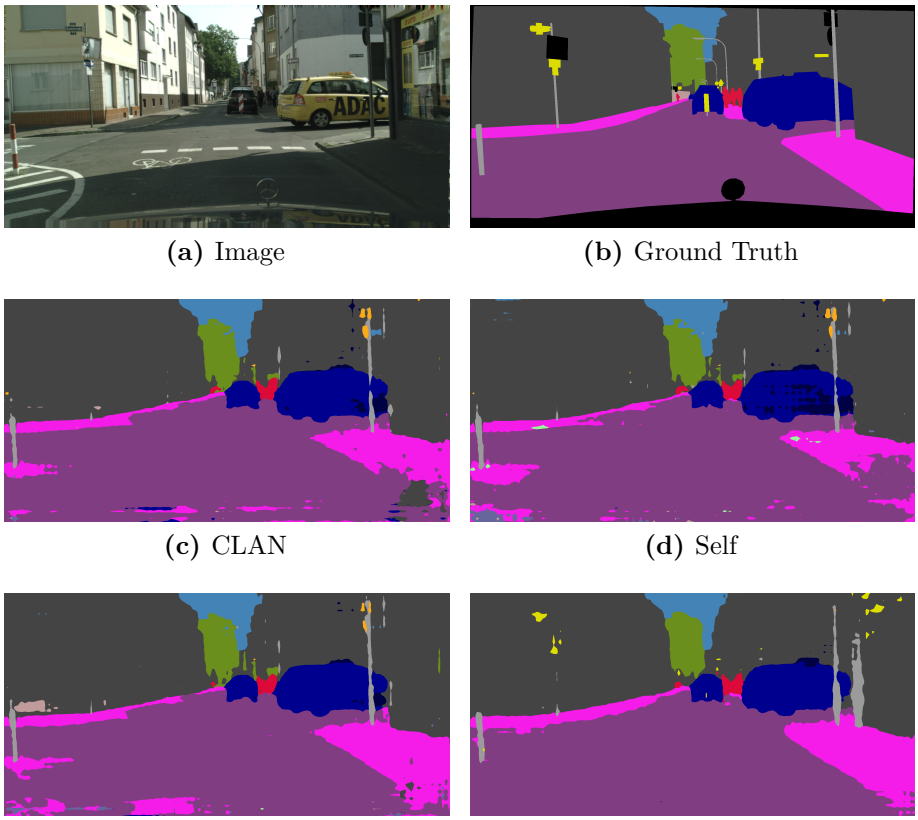


Figure 6.2: Unsupervised Domain Adaptation

(only 4% gained).
However when source and target domain are already very similar, the effect of feature-level adaptation is less evident.
Training the semantic segmentation model using as source domain an adapted dataset, with source images very similar in terms of pixel statistics to the target ones, by adapting them with image-to-image translation model, provide a huge improvement with respect to the source baseline, leading to a gain of 20% in performance.
However trying to further enhance domain adaptation by using the adapted dataset as source and a technique of feature level adaptation does not provide competitive results as before, only +1.2% with CLAN and zero gain with self-supervision with respect to adapted baseline.
For this reason is possible to conclude that pixel-level domain adaptation is more powerful than feature-level domain adaptation to align source to target domain.
A combination of both techniques may lead to more competitive and interesting results, however there is the needs of datasets more similar a priori in terms of class distribution, for example applying a data cleaning process in order to reduce the big amount of bias present in GTA V dataset and Cityscapes, explained better in the limitation section.



(a) Image **(b)** Ground Truth

(c) CLAN **(d)** Self

(e) Self+CLAN **(f)** Target

Figure 6.3: Some samples of the predictions on the validation set of the proposed techniques

6.2 Semi-supervised Domain Adaptation

In a real case scenario could be possible that a company that would like to create a perception algorithm for a possible self-driving car based on semantic segmentation solution, has available a small amount of labeled target data, due to their expensive labeling cost. In this scenario the relevance of using synthetic data is more evident. The outcome that we would like to obtain is to get further knowledge from synthetic data in addition to the one already present in real data.
For this reason I decided to perform some experiments in semi-supervised setting scenario, in which during training over synthetic data are used even labeled and unlabeled target data.
In literature is difficult to find papers that provide experiments using also target labeled data, as the authors prefer to concentrate on unsupervised domain adaptation, but in the industrial sector it can be more interesting.

I provide different experiments with the proposed algorithm, using three different settings, 5% - 20% - 50% of target labeled data in addition to source labeled data and target unlabeled data.
Practically has been added a segmentation loss also on target data every a fixed number of iterations.
For each of the different setting has been performed the same experiment as in the unsupervised domain adaptation case. Networks and parameters are the same used as before, so DeepLab v2 for Semantic Segmentation network trained for 48'000 iterations using a batch size of 1.

In case of using only 5% of target labels, so calculating a segmentation loss only for about 150 real images, is obtained a best value of mIoU over the validation set of Cityscapes of 54.7. Using data augmentation and so performing learning also on the synthetic datasets results change.
When is calculated a segmentation loss also for the original dataset of GTA V is obtained a value of 54.63 mIoU. Instead when it's used as source dataset the one adapted with image-to-image translation, the result obtained is very promising. In fact is obtained a score of **55.25**, that outperform the case of not using any synthetic data.
This result prove that synthetic data help in generalizing a model that will be tested on real data when the amount of real label available is limited (Figure 6.4).
When applying feature-level adaptation using the adapted dataset as source and

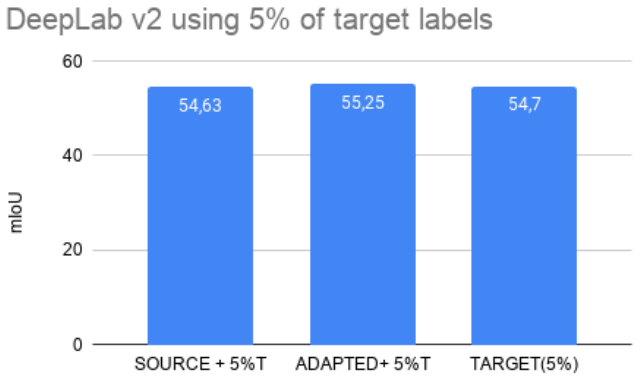


Figure 6.4: Semi-supervised DeepLab v2 using 5% target labels

5% of target annotations is possible to further enhance the results obtained.
Self-supervision helps to further generalize the encoder of the segmentation network, obtaining a score of **55.52** mIoU. However, this is not the case when it's applied adversarial loss with CLAN, leading to a result of 54.69 mIoU and to a best score of 54.54 when combining rotation algorithm and adversarial loss.

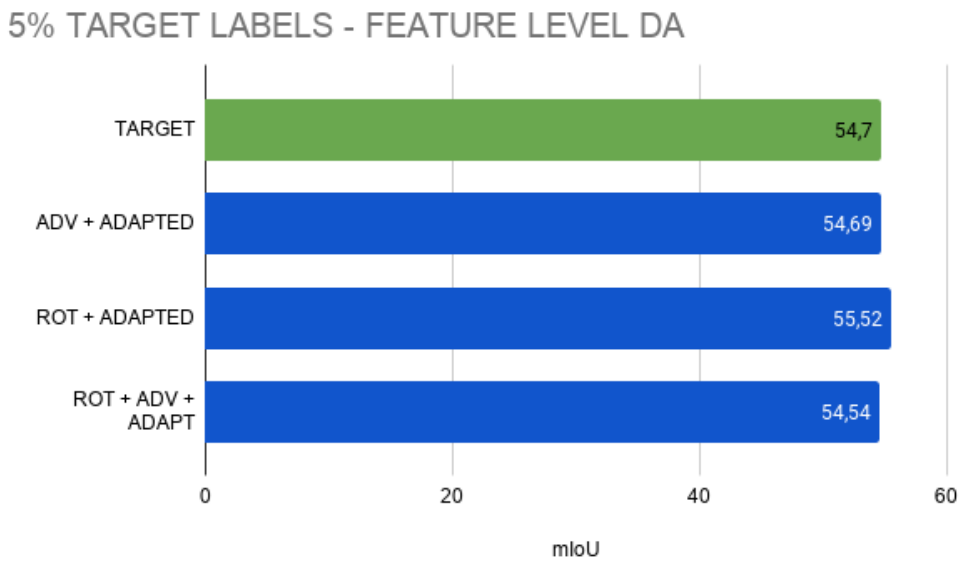


Figure 6.5: Feature-level Domain Adaptation using 5% target labels

In a subsequent trial was trained a DeebLab v2 network using only 20% of target labels, obtaining a base value of 62.04 of mIoU.
Unfortunately in this case the usage of synthetic data did not improve the results obtained considering only real data. Performance degraded both in the case of using original or adapted GTA V images with 20% of target, figure 6.6.
Trying to further enhance domain adaptation with the feature level techniques

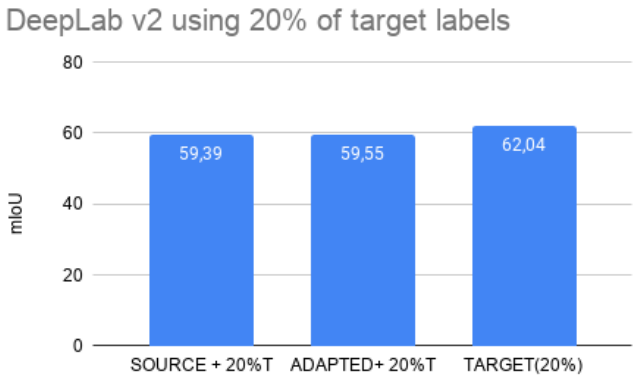


Figure 6.6: Semi-supervised DeepLab v2 using 20% target labels

did not provide any further benefit. Results are shown in figure 6.7.
Finally by performing a training of the segmentation network on the 50% of the

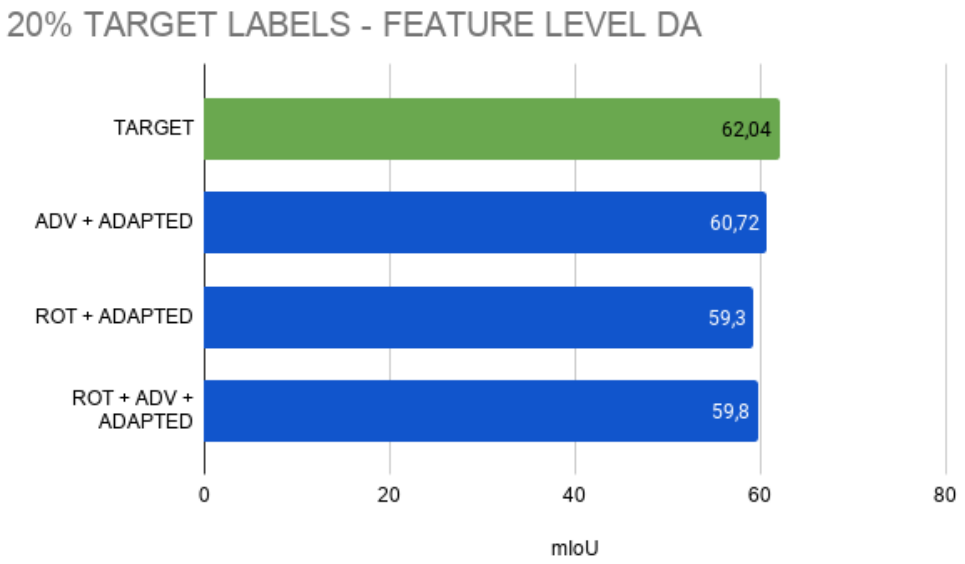


Figure 6.7: Feature-level Domain Adaptation using 20% target labels

Cityscapes label available is obtained a score of 63.67 mIoU.
However also in this case the usage of synthetic data did not provide any benefit to generalization of the semantic segmentation model over real data (Figure 6.8).
Unfortunately feature level adaptation as we might expect also in this case did not

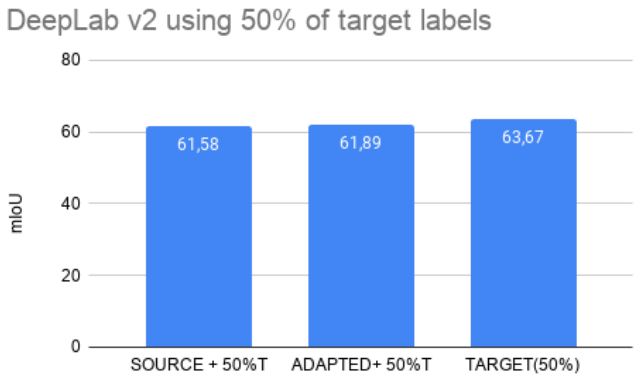


Figure 6.8: Semi-supervised DeepLab v2 using 50% target labels

improve the results obtained using only real data. Showing that synthetic data

did not help when the amount of target label is relatively high to allow a good generalization of the network.

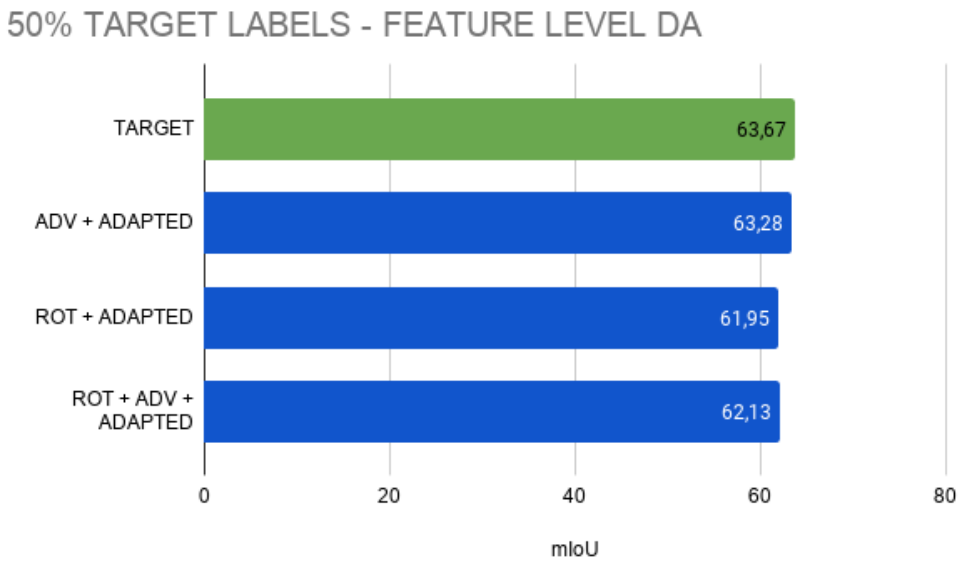


Figure 6.9: Feature-level Domain Adaptation using 50% target labels

However a better hyperparameters optimization is needed in order to have total confirmation of the results proposed here.

6.3 Limitations

One of the biggest problem in adapting GTA to Cityscapes is the very different topography environment of the two domain. Class distributions are not distributed in the same way, how it should be. This problem is mainly due the different setting of the two datasets. GTA is set in Los Angeles, for this reason, leaving aside the realism of the simulation, which anyhow is still satisfactory, there will be many skyscrapers in the background, the traffic lights will be beyond street intersections, buildings are not to high and so on with the details of a typical American city. Contrariwise Cityscapes is collected in German cities, in which there are many bicycles, vegetation and different road signs.
These big difference between the two domains are not solvable with domain adaptation, it is an intrinsic problem in the data, which automatically leads to the introduction of bias inside the models.
For example when training a cycleGAN to map from Cityscapes to GTA, it will learn that due to big amount of trees in the German dataset, in order to change the GTA sample to look like a Cityscapes one, it needs to insert a lot of vegetation in the reconstructed image, making trees appear in the sky where they shouldn't be. Some failure case images can be seen in the following figures (fig 6.10).
A different amount of class specific objects in the two domains may also lead to overfitting when training. For example a model trained on GTA may understand from the different seen images that in the high regions of the images there are always buildings, but that is not the case on Cityscapes, so leading to wrong predictions.

(a) Failure 1 **(b)** Failure 2

Figure 6.10: Failure images when translating from synthetic to real or vice-versa. On left original images on right reconstructed ones.

For these reason it is necessary to use domains that though different in pixels

colors, brightness, textures, they come from domains with a not too different class distribution, for examples different streets of cities of the same countries or regions.

6.4 Future developments

Despite as described above, there is the need of better datasets, there are different improvements that could be done in order to further enhance domain adaptation results in this scenario.
First of all a better hyperparameters optimization for the proposed method may lead to different and more competitive results.
New self-supervision algorithm may further help generalization of the segmentation network. Self-supervision has been deepened by researchers mainly for standard classification, but application to semantic segmentation are relatively new, [10] was released in December of 2019. New ideas related to self-supervision may born in the next years, in particular referring to autonomous driving and applications of semantic segmentation.
As we seen in the chapter of Generative Adversarial Networks have different problems in order to achieve a stable training, one of the proposed solution was the usage of a different loss function that provide better gradients to the generator during training. In literature has not yet been investigate the usage of different loss except for LSGAN for techniques of domain adaptation. For this reason it could be interesting to try to apply them also for domain adaptation.
In the same way, self-attention mechanism and progressing growing training lead to very competitive and stunning results when applied to image generation, in the same way their usage for domain adaptation is not yen been studied, but their application should be considered.
One possible future development should be the application of the proposed methods or similar by using a multi-source dataset. For example in addition to the synthetic dataset of GTA may be used a different dataset such as SYNTHIA, in order to increase the amount of source data available trying to enhance the domain adaptation.
In this work have not been proposed solutions to domain generalization, however in autonomous driving this context is extremely important because it's very easy to imagine a situation in which a model trained for scene understanding needs to work in a completely different domain, unseen during the training process. So in order to propose robust systems such problems need to be considered.
One experiment that due to restricted time I was unable to try was the possibility to join discriminator and auxiliary network in one single network. The task for such joined network would have become much more difficult and the network parameters

would probably have changed, for example the number of iterations might be higher in order to allow a such complex network the possibility to converge.
Finally could be interesting to consider new loss functions during training of both pixel and feature level domain adaptation that take in consideration spatial and geometrical layout of the images. For example in a standard image of Cityscapes used for train semantic segmentation algorithm oriented to autonomous driving, in the bottom of the image there is always the street, in center high probably there is the sky and so on. Objects also have different shapes, pedestrians are tall and thin objects and building are very different from trees. These informations should be insert as priors inside models, and could probably generate models that outperform today's state-of-the-art.
Research in this field is fully active and the most part of the networks proposed in this work are relatively new. For this reason a lot of work could and should be done in order to obtain robust and reliable systems of semantic segmentation in an urban scenario.

Chapter 7

Conclusions

After a deep study of the literature and from the different experiments performed with the proposed algorithm, we can come to some conclusions.
Today, in order to obtain a detailed scene understanding of a driving environment that could be used later for algorithms of autonomous driving, we can use deep convolutional neural networks, that try to solve the task of semantic segmentation. In such task, the aim is to assign a prediction for each pixel of a frame image to a specific class. However, the biggest problem of this approach is the high cost of the per-pixel annotations needed.
Synthetic dataset can be used in order to help generalize such networks thanks to the main advantage that they come with free annotations from design. However, due to the big difference that may exist between real and synthetic data, networks may not be able to generalize easily from such different data and for this reason, domain adaptation techniques are required.
In recent years applications of Generative Adversarial Networks increased, nowadays a great part of algorithms of domain adaptation is based on adversarial training, due to the ability of GANs of reducing divergence in probability distributions.
From different experiments performed by training a semantic segmentation network adversarially with a discriminator network or jointly with an auxiliary task performing a self-supervised algorithm, has been shown that is possible to reduce the domain shift of the two domains leading to a feature-level adaptation.
Image to image translation models instead, allow a direct transfer of the style of real data on the synthetic data. By using style transfer models based on GANs, it is possible to create fine adapted datasets, that can be used as source domain in order to train semantic segmentation networks with or without further adaptation techniques. Such approach has been shown to lead to very competitive results when tested on real data, achieving a generalization of the segmentation model.
From the different experiments conducted, it is possible to conclude that feature-level domain adaptation techniques perform very well when source and target

distribution are far. However, when the two domains are already similar and close, the effect of adaptation is lower.
Pixel-level domain adaptation instead has been shown to be more powerful and leads to a better performance and results in terms of adaptation.
By a joint combination of both techniques, it is possible to achieve state-of-the-art results for the task of unsupervised domain adaptation for semantic segmentation [38].
Results obtained also underline that, when the amount of real annotation is limited, by training a semantic segmentation network in a semi-supervised setting using the algorithms proposed, it is possible to transfer knowledge from synthetic data, obtaning a better result than the case of not using any domain adaptation techniques.
For this reason, it is possible to ascertain that synthetic data help to generalize vision understanding models on the real world when the amount of labels available is limited.
However there is the need to do more studies and this topic will be increasingly at the center of the research in the coming years.

Finally, to conclude, this work has the purpose of trying to exploit as mush as possible the information contained inside data generated from a computer, in order to enhance performance of models that will be tested with real world data. However, the real world is much more complex, random and weird, that is impossible to simulate it totally. For this reason, it is really important to have the knowledge about the type of data on which the model is trained, in order to not introduce any kind of bias, since these models will then be used in a real scenario with real lives.
Making algorithms that can allow autonomous vehicles, it means that the responsibility on the lives that depends on such algorithms, falls on the engineers and developers that such systems have developed.
Being fully aware of this and the ethics that it represents, it is essential for future developments of artificial intelligence.

Bibliography

[1] Jonathan Long, Evan Shelhamer, and Trevor Darrell. «Fully Convolutional Networks for Semantic Segmentation». In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015 (cit. on pp. 1, 21–23).

[2] Xiang Li, Wei Zhang, Qian Ding, and Jian-Qiao Sun. «Multi-layer domain adaptation method for rolling bearing fault diagnosis». In: *Signal Processing* 157 (2019), pp. 180–197 (cit. on pp. 2, 31).

[3] 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: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223 (cit. on pp. 3, 4, 22, 27, 28, 47).

[4] Mei Wang and Weihong Deng. «Deep visual domain adaptation: A survey». In: *Neurocomputing* 312 (2018), pp. 135–153 (cit. on pp. 3, 5, 32, 33).

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. «Generative adversarial nets». In: *Advances in neural information processing systems*. 2014, pp. 2672–2680 (cit. on pp. 3, 5, 33).

[6] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. «Playing for Data: Ground Truth from Computer Games». In: *European Conference on Computer Vision (ECCV)*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Vol. 9906. LNCS. Springer International Publishing, 2016, pp. 102–118 (cit. on pp. 4, 29, 47, 48).

[7] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. «Unpaired image-to-image translation using cycle-consistent adversarial networks». In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232 (cit. on pp. 4, 36–38, 49).

[8] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. «Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2507–2516 (cit. on pp. 4, 41, 42, 44, 48, 51, 57).

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. «Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs». In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848 (cit. on pp. 4, 24, 25).

[10] Jiaolong Xu, Liang Xiao, and Antonio M. Lopez. «Self-Supervised Domain Adaptation for Computer Vision Tasks». In: *IEEE Access* 7 (2019), pp. 156694–156706. ISSN: 2169-3536. DOI: 10.1109/access.2019.2949697. URL: http://dx.doi.org/10.1109/ACCESS.2019.2949697 (cit. on pp. 4, 44–46, 48, 50–52, 59, 67).

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. «Image-to-image translation with conditional adversarial networks». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134 (cit. on pp. 5, 34–37).

[12] Zhengwei Wang, Qi She, and Tomas E Ward. «Generative adversarial networks: A survey and taxonomy». In: *arXiv preprint arXiv:1906.01529* (2019) (cit. on p. 6).

[13] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. «Unrolled generative adversarial networks». In: *arXiv preprint arXiv:1611.02163* (2016) (cit. on p. 9).

[14] Martin Arjovsky, Soumith Chintala, and Léon Bottou. «Wasserstein gan». In: *arXiv preprint arXiv:1701.07875* (2017) (cit. on p. 11).

[15] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. «Least squares generative adversarial networks». In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2794–2802 (cit. on p. 12).

[16] Alec Radford, Luke Metz, and Soumith Chintala. «Unsupervised representation learning with deep convolutional generative adversarial networks». In: *arXiv preprint arXiv:1511.06434* (2015) (cit. on p. 13).

[17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. «Progressive growing of gans for improved quality, stability, and variation». In: *arXiv preprint arXiv:1710.10196* (2017) (cit. on p. 15).

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV] (cit. on p. 16).

[19] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. «Are gans created equal? a large-scale study». In: *Advances in neural information processing systems*. 2018, pp. 700–709 (cit. on p. 18).

[20] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. «Image Segmentation Using Deep Learning: A Survey». In: *arXiv preprint arXiv:2001.05566* (2020) (cit. on p. 21).

[21] Nils Plath, Marc Toussaint, and Shinichi Nakajima. «Multi-Class Image Segmentation Using Conditional Random Fields and Global Classification». In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 817–824. DOI: 10.1145/1553374.1553479 (cit. on p. 21).

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Advances in neural information processing systems*. 2012, pp. 1097–1105 (cit. on p. 22).

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep residual learning for image recognition». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on p. 24).

[24] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. «The Pascal Visual Object Classes (VOC) Challenge». In: *International Journal of Computer Vision* 88 (2009), pp. 303–338 (cit. on p. 26).

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. «Microsoft coco: Common objects in context». In: *European conference on computer vision*. Springer. 2014, pp. 740–755 (cit. on p. 26).

[26] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. «Vision meets robotics: The kitti dataset». In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237 (cit. on p. 27).

[27] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. «Semantic object classes in video: A high-definition ground truth database». In: *Pattern Recognition Letters* 30.2 (2009), pp. 88–97 (cit. on p. 27).

[28] 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: 2016 (cit. on p. 29).

[29] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. «CARLA: An open urban driving simulator». In: *arXiv preprint arXiv:1711.03938* (2017) (cit. on p. 30).

[30] G E Hinton and R R Salakhutdinov. «Reducing the dimensionality of data with neural networks». In: 313 (2006). URL: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google (cit. on p. 35).

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. «U-net: Convolutional networks for biomedical image segmentation». In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241 (cit. on p. 35).

[32] Chuan Li and Michael Wand. «Precomputed real-time texture synthesis with markovian generative adversarial networks». In: *European conference on computer vision*. Springer. 2016, pp. 702–716 (cit. on p. 35).

[33] Chaoyue Wang, Chang Xu, Chaohui Wang, and Dacheng Tao. «Perceptual Adversarial Networks for Image-to-Image Transformation». In: *IEEE Transactions on Image Processing* 27.8 (Aug. 2018), pp. 4066–4079. ISSN: 1941-0042. DOI: 10.1109/tip.2018.2836316. URL: http://dx.doi.org/10.1109/TIP.2018.2836316 (cit. on pp. 38, 49).

[34] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. «Learning from synthetic data: Addressing domain shift for semantic segmentation». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3752–3761 (cit. on pp. 39, 40, 42, 44, 46, 48, 50).

[35] Zhi-Hua Zhou and Ming Li. «Tri-training: Exploiting unlabeled data using three classifiers». In: *Knowledge and Data Engineering, IEEE Transactions on* (2005). DOI: 10.1109/TKDE.2005.186 (cit. on p. 41).

[36] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. «Unsupervised representation learning by predicting image rotations». In: *arXiv preprint arXiv:1803.07728* (2018) (cit. on pp. 44, 45, 57).

[37] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. «Domain generalization by solving jigsaw puzzles». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2229–2238 (cit. on p. 44).

[38] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. «Bidirectional learning for domain adaptation of semantic segmentation». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6936–6945 (cit. on pp. 48, 49, 70).