Università degli Studi di Padova

Department of Information Engineering

Master Thesis in ICT for Internet and Multimedia

Generative Adversarial Models for Unsupervised Domain Adaptation in Semantic Segmentation

Supervisor
Pietro Zanuttigh
Università di Padova

Master Candidate
Marco Toldo

Academic Year 2018/2019
Padova, 09/09/2019

Università degli Studi di Padova

Department of Information Engineering

Master Thesis in ICT for Internet and Multimedia

Generative Adversarial Models for Unsupervised Domain Adaptation in Semantic Segmentation

Supervisor
Pietro Zanuttigh
Università di Padova

Master Candidate
Marco Toldo

Academic Year 2018/2019

Padova, 09/09/2019

Abstract

Semantic segmentation is one of the most challenging computer vision problems, which involves a pixel-wise classification to partition the input image into semantically consistent regions. Its relevance comes from the significant role it plays in many real world applications, such as autonomous driving or medical imaging.
Recently, the development of fully convolutional neural networks has proven to be crucial to successfully address the semantic segmentation task. However, while FCNs have shown remarkable performance when employed in dense image classification, their success is strongly dependent on the availability of huge amounts of training data, which has to be annotated through a laborious work.
Moreover, convolutional networks usually are not capable of generalizing what they learn in a specific context to new environments, which may result into a noticeable accuracy drop when data statistical distribution is subject to a shift between train and test phases. This shortcoming, for instance, prevents the usage of synthetic training datasets with easily accessible computed-generated annotations, given that, despite the recent advances in computer graphics allowing photo-realistic image generation, the inherent domain discrepancy with real data causes a decrease of model performance when final predictions are computed on real images.
Our goal is to address these limitations by implementing an unsupervised domain adaptation framework specifically designed for the semantic segmentation task and tested in a synthetic-to-real adaptation scenario. First, we propose a pixel-level adaptation strategy based on a cross-domain image mapping to transfer visual attributes from an unlabelled target dataset to a fully annotated source one. In this way we are able to exert a form of target supervision by resorting to adapted source examples still carrying useful annotations. We also add a semantic consistency constraint to force generated images to share the same semantic content of their original versions, so that source label maps can be safely transferred to the corresponding adapted data. In a second moment, we introduce an additional feature-level adaptation to enforce feature distribution alignment between source and target domains. For both the image and feature level adaptation tasks we resort to an adversarial learning strategy based on the architecture of generative adversarial networks. Finally, we perform a single stage model optimization where all the components of our domain adaptation framework are simultaneously trained.

Contents

Listing of figures

Listing of tables

1 Introduction

The development of deep learning architectures has been the enabling factor for the solution of complex tasks in the machine learning and computer vision fields. Many advanced real-world systems depend on some level of comprehension of the surrounding environment, usually based on the analysis of visual data. On this regard, visual scene understanding approaches aim at extracting the semantic content of a scene, and this can be done at different abstraction levels. Image classification, for example, lean on the discovery of global scene properties, while more advanced tasks, such as object detection or semantic segmentation, are more focused on relations between local image components at a pixel or instance levels.

Until not so many years ago, visual understanding problems were addressed resorting to complex algorithms with multiple stages. A feature extractor module was employed to detect visual descriptors (e.g. SIFT or Haar wavelets), which were then provided to a feature classifier (e.g. SVM or Random Forest) for the final output classification. Recently, the adoption of deep convolutional neural networks (DCNNs) has brought a huge performance boost in most computer vision tasks. These deep network models consist of a large number of elementary computing units organized in a layered structure, to form a highly non-linear and end-to-end trainable framework able to effectively deal with the whole feature extraction and classification process in a single shot.

The significant amount of complexity in terms of millions of learnable parameters, despite playing a fundamental role in the solution of advanced tasks, is a critical issue for deep neural networks, especially when compared with previous methods based on hand-crafted descriptors. Not surprisingly, in fact, a great contribution to their success has to be attributed to the availability of large volumes of annotated data from public datasets, which turned out crucial to avoid overfitting phenomena and enable their full potential. Unluckily, building such datasets could be tremendously expensive and time consuming, since it usually requires a huge human effort during data generation and labelling processes. This is especially true when dealing with dense prediction problems, such as semantic segmentation, which entails a pixel-wise labelling action.

A viable solution to the training data shortage could be to take advantage of fully labelled (and possibly more easily accessible) samples from a source domain different from the target one, on which we would like to perform our final predictions. Particularly appealing is, for instance, the use of synthetic data for training purposes. Not only computer generated imagery has reached in the last few years a high level of realism (thanks to non-stop progress in computer graphics), but its exploitation also allows a scalable and almost effortless dataset generation and labelling, as well as a total control over training data properties.
A few examples of real-world applications that could benefit from the availability of multi-domain data are autonomous driving, for which several synthetic datasets have been created in order to address semantic understanding of urban scenes, and video-surveillance systems, since it is possible to harness multiple internet sources to collect security cameras footage. Medical applications could make use of additional training information as well, given that sample annotation is really expensive due to medical expertise requirements.

Unfortunately, deep neural networks tend to fail to generalize learned knowledge to new domains and even a minor change in the statistic distribution between the source and target data could strongly degrade their prediction effectiveness. So, for example, if we train our predictor only on synthetic data (source domain) and test it in a real-world environment (target domain), it will probably have low performance due to an input distribution gap between the two phases. Therefore a domain adaptation

process is needed, where domain adaptation can be defined as the task of learning a predictor that performs optimally even in case of domain discrepancy between source and target samples. In addition, an adaptation based only on source ground-truth information (i.e. using no labels for the target domain) is called unsupervised domain adaptation.

Our work is focused on unsupervised domain adaptation specifically applied to semantic segmentation. As source domain we consider synthetic images from a computer generated dataset, which can reasonably assumed to be fully annotated. Then, our objective is to learn a segmentator with good prediction performance on real-world test pictures that carry the same semantic content of synthetic ones. Moreover, the training procedure is based only on source supervision, which means that we completely ignore target label maps (even if available) until the final testing phase.
The approach we adopt combines a generative process to perform a cross-domain image translation and an additional feature alignment operation, both carried out by means of adversarial learning [2].
First, following the framework proposed by J. Zhu et al. [3], we make use of an image-to-image translation network, with the purpose of discovering a mapping from the source domain to the target one. Domain adaptation is then performed by supervisedly training the semantic segmentator on labelled source examples that have been previously translated into the target domain.
Secondly, we extend the work of [3] with the introduction of a semantic consistency loss [4] to preserve the semantic content of images during the cross-domain translation. In particular, a semantic segmentator is inserted inside the original framework and used to measure the semantic distance between original and transformed examples.
Finally, we introduce an adversarial feature framework consisting of two additional feature discriminators. They work as their image-level counterparts, while being fed with semantic features rather than elements from the image space. The objective is to further improve the degree of adaptation reached by adopting just a generative approach.

The thesis is organized as follows: in Chapter 2 we describe convolutional neural networks and existing CNN-based approaches to solve the image classification and

semantic segmentation tasks, in Chapter 3 we introduce the concept of adversarial learning and its application to solve both generative and domain adaptation problems with a specific focus on semantic segmentation, in Chapter 4 we propose in detail our model together with the optimization strategies we adopt to train it and finally in Chapter 5 and 6 we show some experimental results and draw conclusions.

2

Semantic Segmentation

Semantic segmentation is one of the most challenging problems in the computer vision field. It combines image segmentation with object detection in order to produce a partitioning of the input image into semantically uniform regions. In practice, the task is to assign a class label to every pixel inside the image, according to the content which is represented. The result of the labelling operation is a dense and structured output, since it involves multiple pixel-level predictions and requires the recognition of spatial relations between image components.

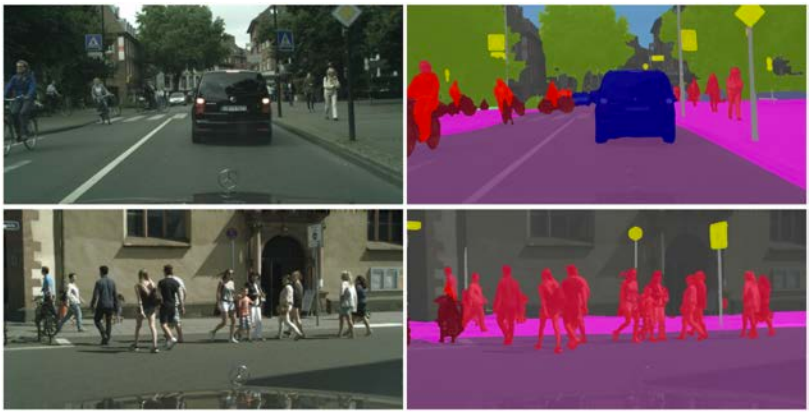


Figure 2.1: Examples of semantic segmentation of urban scenes.

Segmentation networks are often employed in complex computer vision systems for real-world applications. The availability of high quality images, which can be acquired at low cost from camera systems with essentially no location constraints, naturally comes with the need of tools able to interpret and understand their semantic content. One popular application is autonomous vehicles, which strongly rely on extremely accurate vision-based algorithms for tasks such as road segmentation, pedestrian detection and traffic sign recognition in order to safely navigate the urban environment. Other examples are video-surveillance and robotic systems, which require the understanding of the surrounding environment and the precise identification of scene components. Those applications could also benefit from scene geometry information in the form of depth maps, to be added to the color images as input of the segmentation model.

Learning problems based on visual information have traditionally been addressed as two-staged processes by performing feature extraction and classification separately. However, it's only with the spreading of deep learning frameworks in the past decade that many computer vision problems have been successfully solved. Nowadays, state-of-the-art semantic segmentation models are built upon convolutional neural networks (CNNs), which have been successfully applied to the image classification task, and more specifically on their extension into a fully convolutional form (FCNs). In the next sections we will briefly introduce deep neural networks, and in particular their convolutional reinterpretation. We will also review some convnet-based approaches in the context of image classification and semantic segmentation.

2.1 Convolutional Neural Networks

2.1.1 Artificial Neural Networks

Artificial neural networks (ANNs) have been developed to simulate the learning process in the human brain. They are based on a simple computing unit called neuron, which takes multiple input values and performs their weighted sum. In addition, a non linear activation function is applied to get the final output. More than one elemental unit connected together form a neural network, by which in principle it's

possible to model a wide range of complex non-linear functions relating the net input with its output. In figure 2.2 we report a graphical representation of an artificial neuron.

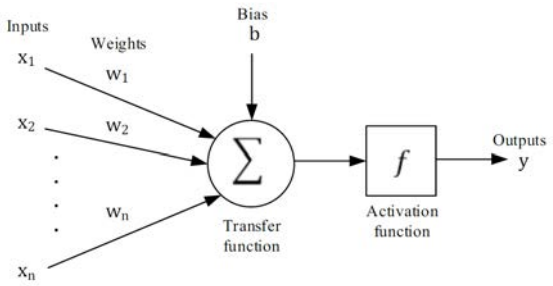


Figure 2.2: Scheme of an artificial neuron.

The most common network structure is the multi-layer feed-forward framework: neurons are arranged in layers and each node (except for the input ones) is fed with the outputs from the previous stage. A key feature of ANNs is the ability to learn a specific function given a sufficient amount of input-output observations. The learning process consists in finding the best values of the network weights to match those observations and it is usually performed by means of the back-propagation algorithm (section 2.1.3) [5].

2.1.2 CNN Architecture

Convolutional neural networks are a special class of ANNs, particularly suitable to process image data thanks to their capability of performing spatial filtering. Essential CNN features are local connectivity and weight sharing: each neuron is connected to a relatively small number of other neurons from the previous layer and weights are shared by different elemental units of the same layer. Both properties combined allow the network to apply multiple convolutional operations with learnable kernels and generate feature maps as intermediate outputs. Usually, inside a single convolutional layer several feature maps are jointly produced as a result of multiple filtering operations applied simultaneously.

More formally, each layer takes in input a 3D tensor of size $H \times W \times D$ (which corresponds to the input image for the first layer or alternatively to a collection of

intermediate feature maps) and produces an output of size $H' \times W' \times D'$, by convolving the input with D' filters of size $K_1 \times K_2 \times D$. The amount of input padding and the stride parameter, defined as the interval at which filters slide through the input volume, determine the actual output spatial dimensions H' and W'. Similar to ANNs, activation functions are applied on the convolution output to introduce non linearities. A popular solution involves the ReLU activation, which behaves like an identity function in correspondence of positive input values, while blocking the negative ones.

An additional dimensionality reduction is achieved by a sub-sampling action carried out by pooling layers. The result is an increased robustness against noise and, at the same time, a drop of the computational complexity.

The feature maps extracted by a concatenation of multiple convolutional and pooling stages are fed to one or more fully connected layers (with the inevitable loss of spatial information) and a problem-dependent classifier (a softmax function for example) is employed to compute the final prediction. The typical CNN architecture is shown in figure 2.3.

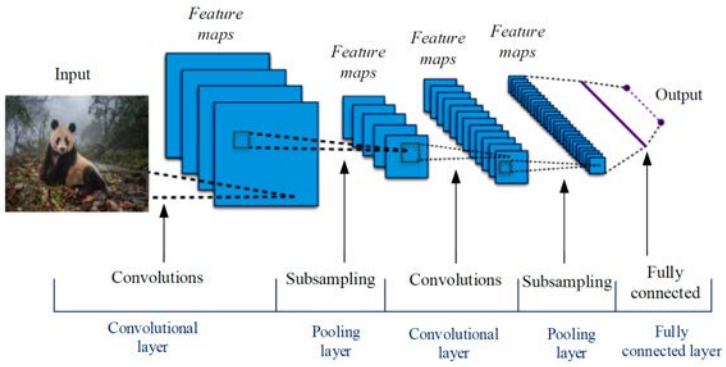


Figure 2.3: An example of CNN architecture.

Convolutional neural networks are capable of extracting hierarchies of features. Shallow layers have access to local information, therefore are focused on image details at low-level and are able to capture simple abstractions. Going deeper inside the network, features loose spatial information in exchange for a more complex understanding and a better invariance to input transformations. The ability of effectively and automatically extract meaningful features is fundamental for CNNs to successfully learn models in many computer vision applications.

2.1.3 Training Procedure

Training a CNN means optimizing a task-dependent loss function $L(\boldsymbol{\theta})$ with respect to the network parameters $\boldsymbol{\theta} \in \mathbb{R}^d$. The most popular optimization strategies are gradient-based and rely on the back-propagation algorithm. First, input samples are fed to the network and predictions are computed. Then, the error information, which measures the mismatch between the expected and actual outputs and is measured by the loss function, is back-propagated through the network in the form of loss gradients with respect to weights and biases $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. Finally, network parameters are updated moving in the opposite direction of the the computed gradient.
The update step can be executed following different variations of the standard gradient descend approach:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{2.1}$$

where the learning rate η controls the size of the update and the gradient is calculated over the entire training set. A popular variant, called stochastic gradient descent (SGD), speeds up the training procedure by executing an update for each training example at the price of higher instability. A trade-off between balance and speed is achieved by the mini-batch gradient descent strategy, for which the update is performed every time the network sees a subset of the training data.
To reduce oscillations typical of SGD, a momentum method has been introduced. The update step can be written as:

$$\boldsymbol{v_t} = \gamma \cdot \boldsymbol{v_{t-1}} + \eta \cdot \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{2.2}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{v_t} \tag{2.3}$$

where the exponential moving average of the gradient $\boldsymbol{v_t}$ is used in place of the gradient itself. Another widely used optimization algorithm is the Adaptive Moment Estimation (Adam) [6], which is a further extension of SGD with momentum with the following update rule:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \frac{\eta}{\sqrt{\hat{\boldsymbol{m}}_{\boldsymbol{t}}} + \epsilon} \cdot \hat{\boldsymbol{v}}_{\boldsymbol{t}} \tag{2.4}$$

where $\hat{\boldsymbol{v}}_{\boldsymbol{t}}$ and $\hat{\boldsymbol{m}}_{\boldsymbol{t}}$ are bias-corrected estimates of respectively the first and second moment of the gradient. An exponential moving average and a rescaling operation are used to compute $\hat{\boldsymbol{v}}_{\boldsymbol{t}}$ and $\hat{\boldsymbol{m}}_{\boldsymbol{t}}$. Moreover, β_1 and β_2 are the hyper-parameters

controlling the exponential rate decay for respectively the first and second moment estimate.

2.2 From Classification to Semantic Segmentation

2.2.1 Image Classification with CNNs

The first work to show the potentialities of convolutional neural networks is the one proposed by LeCun et al. [7] (1989), who exploit CNNs to perform digit classification. Even though the original LeNet model [7] was composed of very few layers with a relatively small number of parameters, major performance constraints were imposed by hardware limitations.

The first popular CNN implementation is the AlexNet [8], proposed by Krizhevesky et al. in 2012. It won with impressive results the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [9] in that year, performing significantly better than traditional machine learning and computer vision approaches for visual recognition and classification tasks. The AlexNet architecture is similar to the LeNet one, with an increased number of convolutional layers and multiple filters for each of them.

Showing that depth in CNNs is a critical factor, the VGG network [10] has been proposed by Simonyan et al. in 2014. The main structure resembles the one of the original AlexNet, but only 3×3 convolutions are employed and the total amount of filters is largely increased. The VGG model is commonly used as a backbone feature extractor for many advanced applications, such as semantic segmentation.

The GoogleNet [11] architecture designed by Szegedy et al. is the winner of the ILSVRC in 2014. Driven by the idea of capturing sparse correlation patterns in convolutional structures, a new inception module with convolutions of different kernel sizes is introduced. An improved and more efficient version of the module also incorporates 1x1 convolutions for dimensionality reduction purposes. Hence, the resulting GoogleNet model is a stack of inception layers with only a total of 7M parameters, far less than the AlexNet (60M) and VGG-19 (138M) solutions.

The concept of residual connection is introduced in the CNN framework by He et al. with the ResNet architecture [12] in 2015. A generic residual block receives the output of the previous layer, performs a series of operations (which depend on the specific architecture) and returns the sum of the intermediate result with the original input. By extension, a residual network is a composition of several residual blocks. The diagram of a basic residual module is reported in figure 2.4.

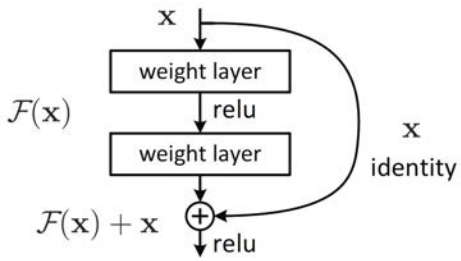


Figure 2.4: Residual block.

The key property of residual connections is that they provide a better gradient preservation during the back-propagation step, which is crucial for an effective network training. This allows to successfully design and implement ultra-deep frameworks, while keeping the complexity under control.

2.2.2 Semantic Segmentation with CNNs

Convolutional neural networks, as defined in the most popular architectures, output a single vector containing the discrete probability distribution for a set of pre-defined classes. For this reason, they are suitable to perform non spatial predictions, such as whole-image classification. On the contrary, semantic segmentation can be seen as an extension of the image classification problem, where global semantic information must be fused with local spatial information to perform a pixel-wise dense prediction. Actually, it would be possible to modify the CNN structure to get a pixel-wise prediction (by changing the configuration of the final fully connected layer for example), but this would increase a lot the model complexity, thus making the optimization computationally infeasible.

A solution that has proven to be effective to address the semantic segmentation task is the conversion of CNNs into a fully convolutional form, capable of recovering spatial knowledge from coarse features extracted by the convolutional layers with the help of a decoder module.
The work of Long et al. [13] is the first one to come up with an end-to-end trainable fully convolutional network (FCN) for the segmentation task. Their idea is to exploit representations extracted by a pre-trained classification network (the VGG, for instance), which is adapted in order to accept an input of arbitrary size and produce a prediction map of the same dimension of the input. First of all, fully connected layers are reinterpreted as convolutions with kernels the size of the input activation map and with as many filters as the number of neurons in the original layer. This essentially removes any constraint on the input dimension, while making the output no longer a single vector but a tensor with coarse spatial information. Those features are then upsampled by a deconvolution layer, which performs a learnable interpolation to obtain a dense segmentation. Additionally, skip connections are introduced to provide the final dense classifier with information of low-level fine details from shallower feature maps. A simple FCN scheme is shown in figure 2.5.

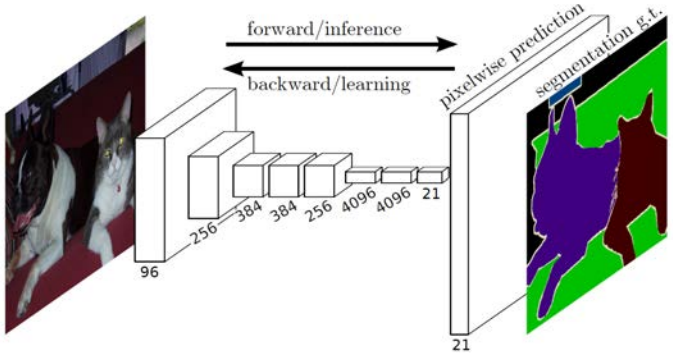


Figure 2.5: An example of fully convolutional network (FCN).

Fully convolutional networks, as defined in the original form [13], tend to overlook global information: even though deep layers in FCNs usually have large receptive fields filling the entire input image, their practical coverage could be much smaller and thereby not able to fully capture global context [14]. To overcome this shortcoming, Liu et al. [14] suggest a simple extension of the FCN model (called ParseNet) based on the addition of scene-level context to feature maps. In particular, global average pooling is applied on a generic layer output to extract a context vector, which

is then unpooled and fused to the associated activation map before being fed to the next layer. Furthermore, L2 regularization and scale factors are employed to successfully combine global features at different scales.
Following the same line of work, Zhao et al. [15] propose the pyramid scene parsing network (PSPNet). The main feature of the PSPNet is the presence of a pyramid pooling module before the classification layer: since global descriptors may not hold enough representational power when highly detailed input images are provided, additional multi-scale information is collected from sub-regions of different sizes by multiple pooling operations. The resulting pooled maps are then fused together for the final prediction. A graphical representation of the PSPNet is shown in figure 2.6.

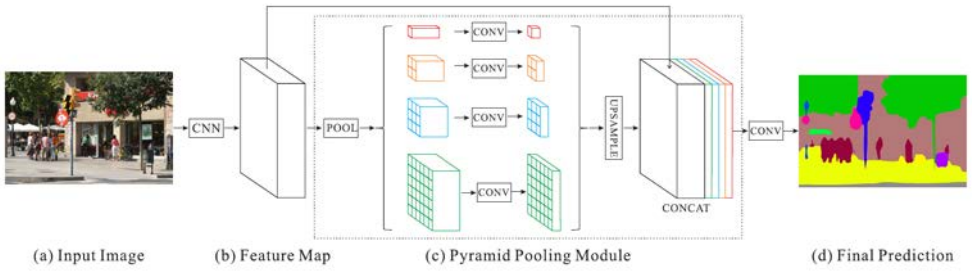


Figure 2.6: PSPNet with pyramid pooling module.

An alternative approach to convert CNNs into a fully convolutional form relies on a modified version the standard convolution, called atrous convolution [16, 17, 18]. The idea is to increase the size of convolutional filters by creating holes inside kernels and filling them with zeros. This solution effectively enlarges the receptive field of neurons (with no additional complexity), and partially removes the subsampling requirements, which are essential in CNNs for learning at multiple abstraction levels, but are also responsible of harmful reductions of signal resolution.
In this direction, Chen et al. develop the DeepLab architecture [16], based on the repurposing of the VGG-16 classification network in a fully convolutional fashion, with the introduction of the atrous convolution and a fixed bilinear upsampling in place of the original transposed convolution. On top of that, a fully connected Conditional Random Field (CRF) is appended to the FCN output to improve the dense prediction accuracy on low-level details (e.g. object boundaries).
A second version of the DeepLab [17] is provided with an atrous spatial pyramid pool-

ing (ASPP) module for multi-scale context capturing (following the same concept of [15]), equipped with multiple parallel atrous convolutions at different rates. Finally, in a third variant (DeepLabV3) [19] the ASSP module is revisited with the addition of a 1×1 convolution and a global pooling stage to extract image-level features and the CRF is removed. In figure 2.7 we report the DeepLabV3 architecture.

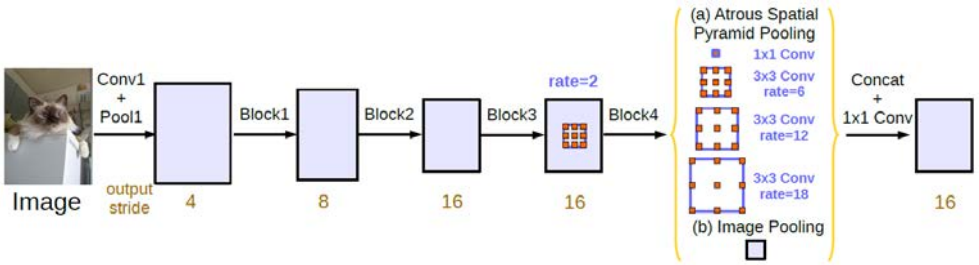


Figure 2.7: DeepLabV3 with ASPP module.

A major weakness of FCNs is the heavy computational requirements for both training and inference stages due to the large amount of learnable parameters. This is a strong limitation for mobile and embedded vision applications affected by strict hardware constraints or for complex frameworks which are built upon more than one network connected together (such as the increasingly popular adversarial architecture)
In this regard, the MobileNetV2 recently proposed by Sandler et al. [1] achieves state-of-the-art performance of mobile networks in multiple computer vision tasks, including semantic segmentation. Its design is based on the previous version of the MobileNet [20]. The primary component is the depthwise separable convolution, which is responsible for both the efficiency and the reduced weight of the architecture. It is split into two phases: first a depthwise convolution performs several individual single-channel lightweight filterings, then a pointwise 1×1 convolution linearly combines input channels to construct the output features.
Let h_i, w_i be the input height and width, d_i, d_j be the input and output number of channels and k be the size of the filter. Then, while the standard convolution has a computational cost proportional to $h_i \cdot w_i \cdot d_i \cdot d_j \cdot k \cdot k$, where we assume that spatial dimensions h_i, w_i are kept constant and d_j square filters of size $k \times k \times d_i$ are applied, the depthwise and pointwise operations summed together result in a cost of $h_i \cdot w_i \cdot d_i \cdot (k^2 + d_j)$. This leads to a reduction factor of $(h_i \cdot w_i \cdot d_i \cdot d_j \cdot k \cdot k)/(h_i \cdot w_i \cdot d_i \cdot (k^2 + d_j)) = k^2 d_j / (k^2 + d_j) \approx k^2$ (assuming $k^2 \ll d_j$)

in terms of number of parameters with respect to an identical solution based on the basic convolution.

The MobileNetV2 also introduces the inverted residual blocks with linear bottleneck (figure 2.8): a low-dimensional input representation is first expanded with a pointwise convolution into a high-dimensional space and then re-projected back after that a depthwise convolution based filtering has been applied. As described in section 2.2.1, a residual connection is built between input and output maps.
To address the semantic segmentation task, the MobileNetV2 has been inserted as the backbone feature extractor inside the DeepLabV3+ model [21]. The MobileNet combined with an ASPP module forms the encoder, while a light decoder with a few convolutional layers fuses features from the encoder output with intermediate representations for the final prediction.
In our work we resort to the MobilenetV2 network embedded in the DeeplabV3+ framework [21], as it offers a good trade-off between segmentation performance and memory occupation.

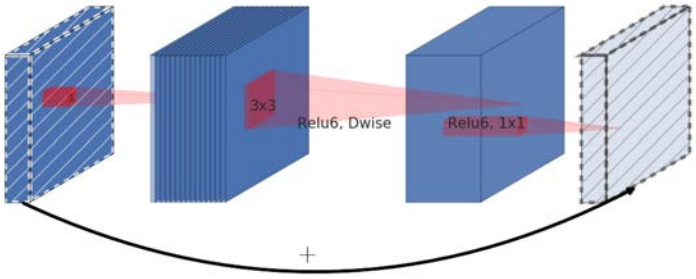


Figure 2.8: MobileNetV2 inverted residual.

3

Domain Adaptation

As extensively discussed in Chapter 2, deep convolutional networks have brought significant improvements in a wide range of computer vision tasks, such as image classification and semantic segmentation. Nonetheless, their success is strictly related to the availability of massive amounts of labelled data. Unfortunately, building such datasets can be quite cumbersome, especially when the collection of highly detailed dense manual annotations is required (e.g. semantic segmentation). At the same time, deep models suffer from poor generalization performance, making so that it's not possible to directly transfer acquired knowledge from a source domain where labels are plentiful or easily accessible to a target domain lacking of sufficient annotations.

Domain adaptation presents itself as a possible solution to the annotated data shortage. Let X be the input space and Y be the label space, and assume the existence of a source and a target joint distributions (that we denote respectively as $S(x, y)$ and $T(x, y)$) from which we can drawn source and target labelled samples (x_S, y_S) and (x_T, y_T). Thus, the domain adaptation problem can be stated as the task of learning a predictor $h : X \rightarrow Y$ that performs well on target data when target labels are available in a small quantity or even missing, by resorting to source annotations.
In our work we address the specific case of unsupervised domain adaptation, therefore we assume complete lack of target supervision.

Unsupervised domain adaptation in deep learning has been traditionally addressed by identifying a measure of domain dissimilarity between source and target distributions. Some works refer to the Maximum Mean Discrepancy (MMD), such as [22, 23, 24]. Tzeng et al. [22] introduce an adaptation layer and a domain confusion loss in the standard CNN architecture [8] to learn domain invariant representations. In [23, 24] the degradation in transferability of application-specific hidden representations is tackled by matching mean domain embeddings in a new space. A different approach relies on correlation alignment, taking into account second order statistic dataset properties [25, 26]. A key factor for the majority of these works is that deep adaptation is performed end-to-end by assisting the task loss with a supplementary adaptation objective. Furthermore, no restriction to a specific deep architecture is assumed.

Game-changer is the introduction of adversarial learning in the context of domain adaptation. The adversarial approach, in fact, has proven to be suitable to successfully perform representation alignment both at feature and pixel levels. In the next sections we will discuss the Generative Adversarial Network (GAN) model [2] (the first to propose the adversarial framework), while briefly introducing the more general case of generative networks. We will also describe the image-to-image translation task, and its connection to domain adaptation. Finally, we will illustrate some main contributions in literature to adversarial domain adaptation, and in particular we will focus on adaptation techniques applied to semantic segmentation.

3.1 Generative Networks

Generative models are based on the evaluation of complex probability distributions. In other words, given samples drawn according to an unknown probability distribution p_{data}, the objective is to learn a representation of that specific p_{data}.
Deep generative frameworks built upon explicit probability estimation usually are very difficult to be trained with the standard back-propagation algorithm due to problematic gradient computations. Nonetheless, explicit density models that keep computational tractability have been successfully developed, and they usually rely on

particular structures that provide that tractability or depend on manageable approximations of the likelihood function [27, 28].
Another solution that has been pursued by a different line of work is to avoid explicit estimation of probability distributions by adopting an implicit density computation. On this direction, we report the well known work of Goodfellow et al. [2]. The key contribution of [2] is the introduction of the concept of adversarial learning, which is employed in the original paper to address the generative task, but has been successfully applied to solve a wide range of different problems, such as semi-supervised learning and unsupervised domain adaptation.
In the next sections we will describe the Generative Adversarial Network (GAN) model from [2]. We will also discuss a particular case of generative task, that is image-to-image translation, which involves input transformation in place of image synthesis from scratch. It constitutes, together with the adversarial technique, a fundamental component in our domain adaptation framework.

3.1.1 Generative Adversarial Networks

The generative adversarial framework can be thought as a game between two players, a generator and a discriminator. The generator has to learn to produce data with the same statistical distribution of samples from a training dataset. To do so, it is paired with a discriminator, which has the goal of understanding when input data comes from the original set and when instead it has been generated. At the same time, the generator is trained to fool the discriminator by creating samples that look authentic.
The cooperation (or competition) between the two components is essential to successfully solve the generative task: the generator is helped by the discriminator in order to improve the realism of synthesized samples, while the discriminator enhances its prediction performance with the support of an accurate generative action. Balance during the learning process is also fundamental: a discriminator or a generator becoming too strong prevents the other player (and in turn itself) from learning useful information and provokes model collapse. However, when the training procedure is successfully carried out, the generator should be able to create samples statistically very similar from the training ones.
More in details, the generator corresponds to a differentiable function $G(\mathbf{z}; \boldsymbol{\theta}_G)$, where

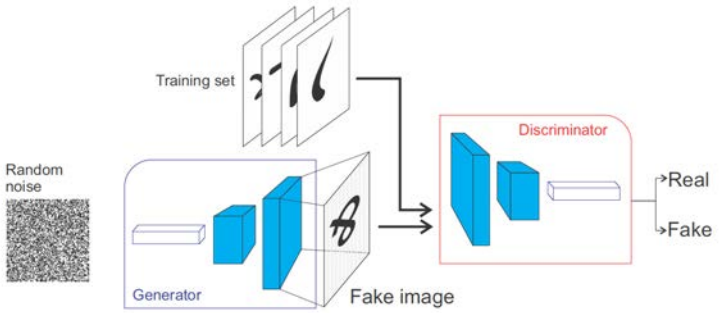


Figure 3.1: Architecture of the Generative Adversarial Network.

we indicate $\boldsymbol{\theta}_G$ as its learnable parameters. In the original GAN, when G is provided with noise vectors $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$, it returns generated samples $\mathbf{x} \sim p_G(\mathbf{x})$ from a distribution p_G. Extensions of the adversarial model have also been proposed, where G is supplied with different types of input information, such as directly with data from a domain correlated with the output space.
The objective of G is to learn an unknown probability distribution p_{data} over $\mathbf{x}$ when only individual samples $\mathbf{x} \sim p_{data}(\mathbf{x})$ from a training dataset are available, in order to make p_G coincide with p_{data}. The discriminator $D(\mathbf{x}; \boldsymbol{\theta}_D)$ is defined as a differentiable function as well, parametrized by $\boldsymbol{\theta}_D$. When it is fed with a sample $\mathbf{x}$, its output represents the probability that $\mathbf{x}$ comes from the training set ($\mathbf{x} \sim p_{data}(\mathbf{x})$) rather than the set of generated data ($\mathbf{x} \sim p_G(\mathbf{x})$). Both G and D functions are commonly implemented as multilayer perceptrons, but it's possible to extend the adversarial framework to deep convolutional neural networks (DCGANs) [29].

The training procedure requires the definition of the two cost functions $J^{(G)}(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D)$ and $J^{(D)}(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D)$, one for each player, which are directly related to the objectives of G (fool D) and D (distinguish real from fake samples). Since each player's objective involves the other player's parameters, the cost functions depend on both G and D. Additionally, the simultaneous minimization of $J^{(G)}(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D)$ and $J^{(D)}(\boldsymbol{\theta}_G, \boldsymbol{\theta}_D)$ corresponds to a game. Hence, the solution is given by a Nash equilibrium, which is a tuple $(\boldsymbol{\theta}_G^*, \boldsymbol{\theta}_D^*)$ identifying a local minimum of $J^{(G)}$ with respect to $\boldsymbol{\theta}_G$ and a local minimum of $J^{(D)}$ with respect to $\boldsymbol{\theta}_D$.
In the original paper a zero-sum game is proposed to model the adversarial framework. In particular, it is assumed that the sum of the costs of G and D is always

zero, so that we have $J^{(G)} = -J^{(D)}$. A single value function $V(G, D)$ is then defined and a min-max game is played by the generator and discriminator:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}} \left[\log \left(D(\mathbf{x})\right)\right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} \left[\log(1 - D(G(\mathbf{z})))\right] \tag{3.1}$$

where a standard binary cross-entropy cost is employed. By maximizing V, the discriminator seeks to bring $D(\mathbf{x})$ close to 1 and $D(G(\mathbf{z}))$ close to 0, that corresponds to perfect sample recognition. On the other side, a minimization of V with respect to G entails a value of $D(G(\mathbf{z}))$ that tends to 1, which implicates a completely deceived discriminator. Provided that G and D have enough capacity (i.e. a sufficient network complexity), the solution of the min-max game leads to $D(\mathbf{x}) = 1/2$ for every sample $\mathbf{x}$. In this scenario $p_G = p_{data}$, so that the generative task is solved.
The min-max cross-entropy cost in practice is not working well. This is due to saturation effects that appear during the initial training stages when G is not able yet to produce convincing samples and D is always providing correct predictions with high confidence, which as a result induces vanishing generator gradients (the $\log(1 - D(G(\mathbf{z})))$ term is near 0). As a solution to this problem, the authors of [2] propose to split the min-max game into two alternating optimization processes with separate cost functions for G and D, in order to generate stronger gradients.

The GAN model is capable of learning a structured loss in the form of a learnable discriminator, which guides the generative network in its optimization procedure. For this reason, the objective function can be thought as automatically adapting to the specific context, removing, in fact, the necessity to manually design complex losses. Therefore, the adversarial learning introduced in [2] can be extended and adjusted to address multiple tasks that would normally require different types of application-specific objectives.

3.1.2 Image-to-Image Translation

Image-to-image translation is a class of generative techniques where the objective is to learn a function that maps images across domains. The idea is to extract characteristics peculiar to a specific set of images and transfer those properties to a different data collection. In other words, the same scene is converted from one representation

to another.
In principle, many computer vision tasks could fall under the category of image-to-image translation problems. For instance, semantic segmentation could be modelled as a domain mapping application, where the goal is to learn a function converting images from their original shape to label maps containing semantic information. Similarly, vision tasks such as edge detection, super-resolution and image colorization can be seen as image-to-image translations, since they involve pixel-level predictions.
In our domain adaptation framework we employ an image-to-image translation block with a style transfer purpose: both domains among which the mapping is performed comprise realistic urban scenes. Our goal, in fact, is to transfer visual attributes from the target domain to the source one, while preserving source semantic information.

In a more formal definition, the image-to-image translation task aims at discovering a joint distribution of images from different domains. In a supervised setting pairs of corresponding images are available, whereas in the unsupervised scenario only individual images can be accessed from independent sets. The second case is more challenging, but at the same time more practical, since it doesn't require the generation of expensive training datasets. Obtaining couples of matching samples, in fact, can be quite problematic (e.g. for graphic tasks as artistic stylization) or even doesn't constitute a well-defined assignment (e.g. for object transfiguration). Unsupervised image-to-image translation is then defined as the task of learning a joint image distribution when only marginal distributions from single domains are accessible. In our work we resort to unsupervised domain mapping, as no matching pairs from source and target domains are at our disposal.

Gatys et al. [30, 31] are the first to apply convolutional neural networks to address image style transfer. They propose the *Neural Algorithm of Artistic Style* for artistic stylization (2015). Activations from high-level layers of a pre-trained VGG network [10] are used to capture the content of an input image, whereas feature statistic at multiple representation levels provide texture information. The generative process is then guided by a content loss, which measures the distance between features extracted from the input image and a content-target image, and a style loss, which forces statistical alignment in terms of feature correlation between the input image and a style-target image. The synthesized picture should possess the content and the

style of the two target examples.
To overcome the excessive computational burden of the neural style transfer algorithm [30, 31], Johnson et al. [32] split the framework into a trainable generative block and a pre-trained convolutional network for image classification to achieve real-time image translations. A perceptual loss is also accompanied by a per-pixel loss to achieve better stylization performance.
Despite reaching impressive results in artistic stylization, the methods introduced above suffer from the presence of unpleasant artifacts when the synthesized images must appear as real photos. To solve this problem, photorealistic stylization algorithms have been developed [33, 34], usually based on additional refinement steps to improve the results of an initial style transfer stage. For instance, the method suggested by [34] relies on a smoothing action that effectively removes spacial inconsistencies, while causing a considerable increase of the running time.

Image-to-image translation represents a more general problem compared to image style transfer: instead of focusing on individual images carrying content and style information, image-to-image translation involves a mapping between entire image collections. Thus, the attributes extracted and transferred pertain the whole set rather than single examples, resulting in more robust transformations.
The adversarial approach has been widely employed to address the image-to-image translation task [3, 35, 36, 37, 38, 39, 40, 41, 42]. One of the first works following this direction is the one proposed by Isola et al. [35] (2016), who develop a generative framework based on conditional generative adversarial networks [43]. Both the generator and discriminator are provided with additional information to impose a stricter control on the generative process: the discriminative action is performed on matching couples $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}, G_{X \to Y}(\mathbf{x}))$, as opposed to the standard scenario in which only the true $\mathbf{y}$ and fake $G(\mathbf{x})$ samples are available to D. The major drawback of [35] lies in the requirement of supervision during the training process, since corresponding samples from different domains must be accessible.

Unsupervised image-to-image translation removes the need for paired data. In turn, the problem becomes ill-posed, as an infinite set of joint distributions can be inferred from the marginal ones. For this reasons, appropriate constraints must be applied to obtain acceptable solutions.

On this regard, Taigman et al. [36] employ a multiclass GAN for unsupervised cross-domain image generation. An external function is used with regularization purposes to force consistency between input and output of the generative unit. Moreover, the generator is required to be an identity mapping when applied on the output domain. A different line of work is based on a shared latent space assumption [37, 38], which infers the existence of a latent space where corresponding elements from different domains share the same representation. For instance, the CoGAN network [37] is built upon multiple GANs tied by a weight sharing constraint to learn a joint distribution of multi-domain images in absence of supervision. The work in [38] extents the CoGAN model with the insertion of variational autoencoders (VAE) [28] combined with pre-existent generative adversarial modules to achieve high quality image translation. Another popular constraint is the cycle-consistency [3, 39, 40], introduced by Zhu et al. with the CycleGAN [3]. Two mappings $G_{X \to Y}$ and $G_{Y \to X}$ are concurrently learnt by means of the adversarial technique and forced to be one the inverse of the other (i.e. $G_{X \to Y}(G_{Y \to X}(X)) \approx X$ and $G_{Y \to X}(G_{X \to Y}(Y)) \approx Y$). The reconstruction requirement provided by the cycle-consistency loss [3] should encourage the preservation of structural properties during the translation, for a more convincing image generation. As mentioned before, our work is based on the CycleGAN framework.

Recently, one-sided translation models have been proposed to avoid the concurrent optimization of multiple generative networks required by the cycle-consistency [41, 42]. The DistanceGAN [41] is based on a distance constraint, which forces examples from the same domain to preserve their distance during the translation. To this end, a predefined function is used to measure the gap between images.
Similarly, Fu et al. introduce a geometry-consistency constraint [42] to reduce the space of possible solutions to the unsupervised domain mapping problem. Behind their assumption lies the observation that geometric transformations don't alter semantic content. Thus, a predefined geometric transformation is included inside a reconstruction loss, in order to prevent the emerging of generative artifacts.

3.2 Adversarial Techniques for Domain Adaptation

The adversarial technique illustrated in section 3.1.1 can be effectively employed to solve the unsupervised deep domain adaptation problem for visual recognition tasks. The key idea is to bridge the domain gap between source and target distributions by means of a domain classifier, where the discriminative action is applied on data samples (or their feature representations) from different domains. In other words, a measure of domain discrepancy is simultaneously learnt and minimized during the adversarial competition.

The adversarial strategy can be applied directly on the input space (pixel-level) or on an intermediate latent space (feature-level). In the first case, the generator still holds a generative role, as it performs cross-domain mapping of images from the source domain to the target one. The generator-discriminator purpose is therefore to produce target-like images, while preserving their ground truth information coming from source annotations. The availability of the new adapted data, in principle, should allow us to train the predictor directly with target supervision.
There exist several works that have explored the unsupervised domain adaptation problem with a generative perspective [3, 37, 44, 45]. Some of them resort to the development of an image-to-image translation network, such as the already mentioned CycleGAN [3] and the CoGAN [37], which is then exploited to perform source-to-target image mapping. In a second stage, an application specific predictor is learnt on top of the generative model.
Other works focus specifically on synthetic data adaptation [44, 45]. Bousmalis et al. [44] introduce a GAN-based adaptation model, where a task-specific predictor is trained in conjunction with a generative adversarial module. In addition, a content-similarity loss is proposed to prevent an excessive input alteration in the translation process. Shrivastava et al. [45] develop a framework to improve the realism of synthetic images produced by a simulator. In particular, they stabilize the adversarial learning with a local adversarial loss, for which the discriminator is applied on several overlapping patches instead of directly over the entire image. Moreover, they use a history of refined images rather that just the current batch to update the discriminator parameters.

A different branch of research has been focusing on feature-space adaptation [46, 47, 48, 49]. Differently from a generative approach, the generator inside the original adversarial framework is replaced by a neural network performing feature extraction. Thus, the discriminator is now provided with feature embeddings that must be classified as originating from source or target inputs. The adversarial objective, then, becomes the distribution alignment of activations relative to different domains. As a result of the adversarial competition, the feature extractor should span a shared feature space, where samples projected from source and target domains no longer show discrepancy in terms of probability distribution. At this point, a single classifier for both domains can be trained over the common space to carry out the adaptation process.

One of the first contributions to the adversarial feature adaptation problem is brought by Ganin et al. [46, 47] (2015). They propose a solution based on a gradient reversal layer to effectively backpropagate the error information during the adversarial min-max game and correctly update the feature extractor and domain discriminator parameters. A domain classification loss is also joined by a label prediction loss to promote feature discriminativeness in addition to domain invariance.

Following a similar approach, the ADDA model [48] is built to address unsupervised domain adaptation. However, unlike previous adaptation models such as [46, 47], the feature extractor is no longer shared by the source and target domains. Instead, two distinct networks are employed for separate representation learning. This induces an asymmetric feature mapping, while improving adaptation flexibility.

In order to learn more transferable representations, Ren et al. [49] explore adversarial feature adaptation in the context of multi-task learning. The core module is still composed of a feature extractor and a domain discriminator, while the task-specific classifier is replaced with multiple predictors addressing more related objectives. In addition, they test their framework on synthetic data and prove that it produces better visual representations than single-objective competitors.

3.3 Domain Adaptation in Semantic Segmentation

In the previous section we discussed multiple solutions to unsupervised domain adaptation of deep convolutional neural networks based on adversarial learning. The majority of those works, despite being devised as application-independent, are mainly focused on the image classification problem. However, unlike the image classification task, semantic segmentation involves complex high-dimensional representations to capture spatial affinity of local semantics. In addition, semantic segmentation predictions belong to an exponentially large label space due to their dense pixel-level highly-structured nature. For these reasons, domain adaptation in the context of semantic segmentation requires an extra effort in model design, as well as supplementary techniques to carefully deal with the inherent additional complexity.

The first work to propose an adaptive framework for semantic segmentation is [50] (2016). Similarly to previous image classification methods, an unsupervised adversarial approach is employed to perform global domain alignment at feature level. In particular, a domain classifier is trained to distinguish source from target features, while the fully convolutional [13] feature extractor learns to produce domain invariant representations. A local alignment constraint-based technique is also introduced to acquire category specific knowledge from the target domain and achieve a more robust adaptation.

A different path is followed by Zhang et al. [51] (2017), who adopt a curriculum-style adaptive method. First, some high level label properties are inferred on target data by resorting to traditional machine learning algorithms (e.g SVMs). In particular, they opt for an estimate of both global statistic of target semantic maps and local superpixel label distributions to gather information about spatial structures. These are called *easy* tasks, since they don't require complex dense predictions and, in turn, are less sensitive to domain shift. Then, the semantic segmentation *hard* task is learnt with a standard source-based supervised optimization, assisted by additional losses to enforce the inferred properties on target predictions lacking of supervision.

Recently, unsupervised domain adaptation has acquired huge interest in the computer vision community. Many researches have tried to improve the state-of-the-art

of domain adaptation in the semantic segmentation field, especially focusing on adaptation from synthetic data to real world scenes. Some of those contributions resort to style transfer algorithms to perform cross-domain translation of source samples before using them for a target-based supervised training of the segmentation network [52, 53].
Another line of works falls in the category of adversarial adaptation methods [4, 54, 55, 56, 57, 58, 59].
On this regard, Sankaranarayanan et al. [54, 55] use a generative adversarial module to learn an encoder-decoder couple capable of projecting images into a feature space and reconstructing them back for both source and target domains. At the same time the entire network is trained to produce target-like images when features are extracted from input source samples.
With the purpose of learning domain invariant semantic representations, Tsai et al. [56] investigate adversarial learning in the output space (i.e. segmentation maps). They also extend their model with a multi-level adaptation framework to adapt features at different scales.
Similarly, Murez et al. [57] design an adaptation framework based on a backbone encoder supported by multiple auxiliary networks and losses with regularization purposes to achieve domain agnostic feature extraction. The discovered latent feature space is then exploited to learn a domain invariant predictor.

Following a different self-training strategy, Vu et al. [58] address unsupervised domain adaptation in the context of semantic segmentation with the introduction of a loss based on the entropy of output probability maps. Their goal is to improve the low-confident (high entropy) predictions on target samples by mimicking the over-confident (low entropy) behaviour of source predictions. Moreover, they explore a direct approach with a hand-crafted loss, as well as an indirect adversarial method in which the objective is expressed by a learnable discriminator.
Additionally, other works [60, 61] resort to a self-training technique by generating pseudo-labels from high-scoring pixel-wise predictions selected by applying a threshold on probability maps computed on unlabelled data.

In a popular work, Hoffman et al. [4] introduce the CyCADA framework, which relies on the CycleGAN [3] image-to-image translator for pixel-level adaptation. First of

all, the CycleGAN-based generative module, which is augmented with a semantic-consistency constraint to avoid semantic alterations of input data, is trained to perform cross-domain image translation. Then, the generator responsible for the source-to-target mapping is applied on source images before they are provided to the predictor in a supervised training phase. Finally, a separate adversarial feature adaptation step is proposed, where the goal is to align the distribution of features extracted by the semantic segmentator from the source-adapted domain with those from the target domain to further improve adaptation results.
Unfortunately, the authors were not able to fully exploit the proposed method for the semantic segmentation task, due to hardware constraints hindering the insertion of a fully convolutional predictor inside an already memory-demanding generative module comprising four neural networks.
In figure 3.2 we report a graphical representation of the CyCADA model (the final feature adaptation step is not included).

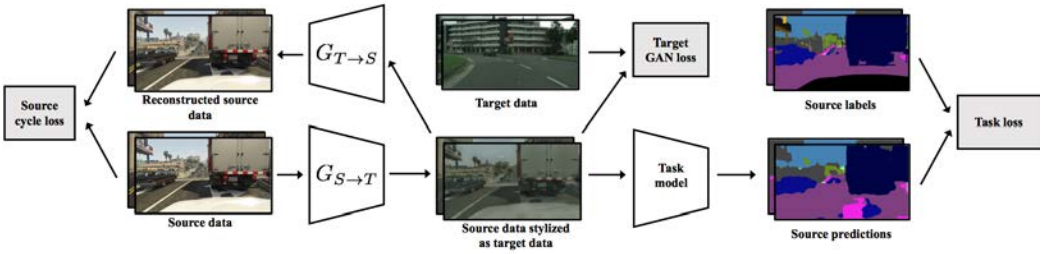


Figure 3.2: Architecture of the CyCADA framework (feature adaptation step not included).

Similarly to [4], Chen et al. [59] propose an extension of the CycleGAN framework by introducing a pair of feature domain discriminators and a couple of semantic segmentation networks (one per domain), which are trained in conjunction with the original image-to-image translation module (two generators and two image discriminators). In addition, a cross-domain consistency loss based on the bi-directional KL divergence is used to encourage similar predictions on original and transformed images in the target domain.
We base our work on [4], since it shows quite good results in terms of adaptation performance, while leaving room for further investigations.

4

Proposed Framework

As already mentioned, we base our work on the frameworks proposed by Zhu et al. [3] for the adversarial generative process and by Hoffman et al. [4] for the more general domain adaptation task, which have been both briefly described in Chapter 3. We insert a lightweight semantic segmentator inside the main CycleGAN framework, pushed by the target of building an unified model in presence of hardware limitations. For this reason, we choose the MobileNetV2 embedded in the DeepLabv3+ frame, which has proven to represent a good trade-off between prediction accuracy and network size, as illustrated in section 2.2.2. We further implement an additional adversarial framework working at feature-level to improve adaptation performance.

Our goal is to perform unsupervised domain adaptation in the context of semantic segmentation. The core idea is to transfer semantic attributes from target to source images by means of a generative process, and then use the target-like source samples in a supervised training. For this reason, we assume to have access to a consistent amount of fully annotated source data, along with a sufficient number of unlabelled images from a target set (we also make use of a small group of labelled target images, but only for testing purposes). More specifically, we resort to a computer-generated dataset of urban scenes as source domain (with in principle no limitations to the availability of labelled data) and we test our final framework on real-world images from an urban environment as well.

As initial step, we adopt a generative approach to solve the unsupervised domain adaptation task. In particular, we implement a framework built upon five different deep neural networks jointly operating. Two fully convolutional generators work together with other two discriminators to learn two cross-domain image translation functions in both source-to-target and target-to-source directions, while a semantic segmentator is included to enforce semantic consistency in the generative processes.
In a second moment, an additional feature-level adaptation is included: we introduce a couple of new feature discriminators that work as their image-level counterparts from the original CycleGAN model [3], but with the job of discriminating between features extracted by the segmentator rather than images.
Finally, we unify the entire framework by changing the training policy from a multi-stage optimization to a single step in which both the generative and segmentation tasks are learnt at once.
In the next sections we will illustrate in detail the architecture that we employ to address the adaptation task and we will discuss the optimization strategies that we adopt to train the model.

4.1 Architecture

4.1.1 Cross-Domain Image Generation

The generative module is based on the CycleGAN framework from [3]. Its original purpose is to solve the unpaired image-to-image translation task, with the underlying objective of learning a joint probability distribution starting from the marginal ones from single domains. The "unpaired" term refers to the fact that no supervision in the form of aligned couples of images from both domains must be provided during the training process. This generative framework has proven to be suitable for color and texture changes, but not for more radical geometrical transformations. This is positive for us, since we are looking for a translation which allow us to safely transfer the ground truth information from the original image to the translated one.

The model is made of two generative adversarial modules (in the standard generator-discriminator form), tied together by a cycle-consistency constraint to enforce image structure preservation. The generator is a fully convolutional network with a concatenation of one stride-1 and two stride-2 convolutions, nine residual blocks, two stride-1/2 fractional convolutions to recover the input dimensionality and a final stride-1 convolution. We remove the final tanh activation layer, which is likely to harm the signal propagation due to saturation effects. The discriminator consists of a series of five stride-2 convolutions with the leaky-ReLU activation. Its role, following the original concept of GANs [2], is to discriminate between images in their original form and images that have been translated from the other domain. We report the generator and discriminator structures in figure 4.1.

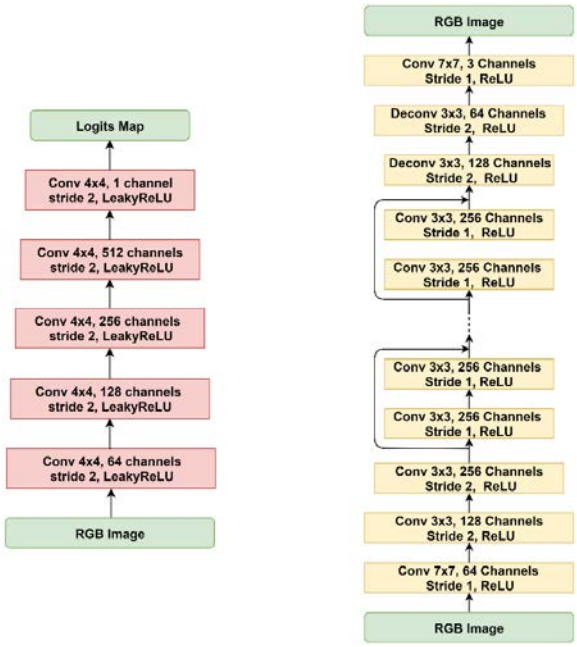


Figure 4.1: Discriminator (left) and generator (right) structures.

4.1.2 Task Specific Adaptation

We embed the generative module in a task-specific domain adaptation framework. In particular, following the work of Hoffman et al. [4], the CycleGAN architecture is augmented with an additional deep neural network performing semantic segmentation on data from both domains. This model add-on is exploited to enforce semantic

consistency between images before and after the generative action. Basically, the segmentator is supplied with original and adapted versions of the same sample and in return it provides information about the semantic changes that the generator has produced. This error information is then propagated back through the network up to the generator, which is optimized in order to minimize semantic alteration (among the other objectives).

The proposed framework, in principle, can utilize any deep network capable of performing semantic segmentation. Nevertheless, on the practical side we have a limited amount of computational and memory resources at our disposal (we work with a single gpu of a maximum of 11GB of memory). Moreover, the generative module in its original form is already quite complex, since it's built upon 4 different neural networks with millions of parameters each.
To overcome this issue, we decide to employ a mobile segmentation network. In particular we resort to the MobileNetV2 [1] embedded in the DeepLabV3+ [21] framework, a state-of-the-art solution for mobile applications. As discussed in section 2.2.2, the key idea behind that architecture is to employ lightweight structures such as separable convolutions and inverted residual blocks for an effective network size reduction, while still keeping a good prediction accuracy. In figure 4.2 we show a graphical representation of the encoder-decoder segmentator architecture.

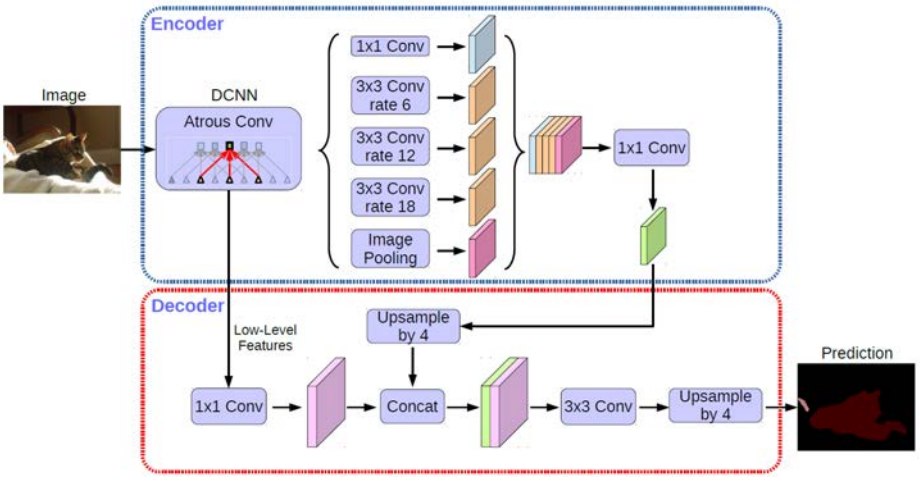


Figure 4.2: DeepLabV3+ architecture. We use the MobileNetV2 [1] as DCNN backbone of the encoder module.

4.1.3 Adversarial Feature Alignment

Aiming at further improving the generative domain adaptation model, we decide to introduce an additional adversarial module to perform feature-level adaptation. The core idea is to replicate the adversarial strategy adopted for the image-to-image translation task, where the goal is a pixel-level distribution alignment to make images from one domain (e.g. source) look as if originating from the other domain (e.g. target) after being translated. The new adversarial objective is instead the adaptation of intermediate feature representations. The goal now is to make generated images from one domain appear statistically identical to original images from the other domain when looking at their semantic representations extracted by the segmentation network.

More specifically, we add a couple of feature discriminators and provide them with activations from the output of the MobileNet feature extractor, before the ASPP and decoder modules. The adversarial game, then, takes place between the two feature discriminators and the two original generators, and eventually also the segmentator if it is made trainable. As a result of the adversarial game, we should have access to a pair of image-to-image mappings capable of translating images across domains, while, at the same time, making generated samples statistically indistinguishable from true ones when projected into the feature space defined by the segmentator network.

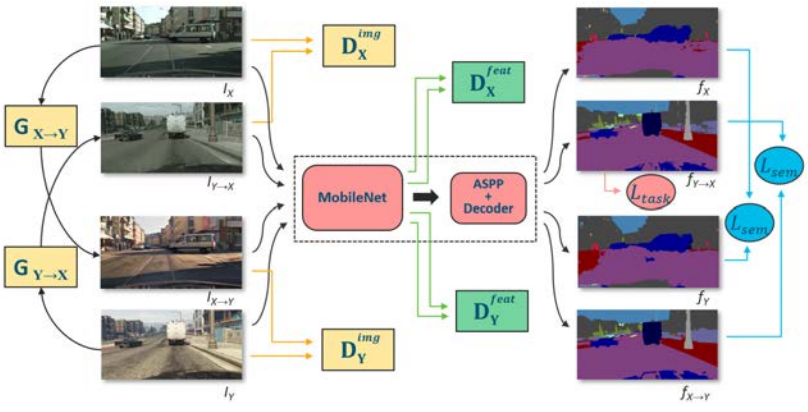


Figure 4.3: Proposed framework.

As concerns the implementation side, the pair of feature discriminators is constructed following the PatchGAN [35] structure described in section 4.1.1. A key adjustment

we make is to decrease the stride value from 2 to 1 in all convolutional layers. The reason for that lies in the fact that the discriminators are not fed with full-size images, but with their feature maps, so that the input spatial dimensionality is reduced. In figure 4.3 a scheme of the proposed framework is reported.

4.2 Optimization Strategies

In this section we describe the methods that we adopt to train the proposed domain adaptation framework. In particular, we report the loss functions that we use for parameter optimization, as well as the strategy we follow to effectively train all the different components.

We define X_S, Y_S as the image and semantic label spaces in the source domain, from which we can access annotated samples in the form of $(x_S, y_S) \in X_S \times Y_S$ pairs, generated according to a known source joint distribution over $X_S \times Y_S$. Analogously, we denote X_T, Y_T as the target data and label spaces, and $(x_T, y_T) \in X_T \times Y_T$ as a generic labelled target image. Since we are operating in the unsupervised domain adaptation scenario, no target supervision can be exploited during the training process. Therefore, we are allowed to use target data samples $x_T \in X_T$ only by themselves, without their respective ground-truth label maps $y_T \in Y_T$ (the joint distribution over $X_T \times Y_T$ is unknown).
In addition, we denote F as a predictor that associates data samples to the corresponding semantic maps. In our case, F is a fully convolutional neural network performing semantic segmentation: given an input RGB image of size $H \times W \times 3$, a dense classification map of size $H \times W \times C$ is computed, where C is the number of classes. Our goal is to learn a segmentator F that performs well on target images, while resorting exclusively to source ground-truth information.

We start by training F in a supervised manner on source data. We use the standard cross-entropy loss:

$$\mathcal{L}_{task}\left(F, X_S, Y_S\right) = \mathbb{E}_{(x_S, y_S)\sim(X_S, Y_S)}\left[-\sum_{h,w,c} y_S^{(h,w,c)} \cdot \log\left(f_S^{(h,w,c)}\right)\right] \tag{4.1}$$

which encourages the predictor to output a probability map $f_S = F(x_S)$ that is as close as possible to the one-hot encoded semantic label y_S (both of size $H \times W \times C$). This first step should give us a predictor $F_S = \arg\min_F \mathcal{L}_{task}(F, X_S, Y_S)$ quite effective on the source domain, but typically not capable of generalize the acquired knowledge to new domains, such as the target one. This causes a performance drop when predictions are computed on target images.

To address the effect of domain discrepancy, we resort to a generative approach. In particular, we employ a generative adversarial framework to learn a mapping to project images from the source to the target space. The objective is to produce adapted source images that resemble target ones, while preserving the ground-truth information at our disposal. In this way, we should be able to introduce a sort of target supervision by exploiting target-like annotated source images to train the semantic segmentator. This process should lead to better results when evaluating the predictor on the target domain.

We define the cross-domain source-to-target mapping as $G_{S\to T} : X_S \to X_T$, and we pair it with a discriminator D_T. The generator $G_{S\to T}$ must fool the discriminator by making source images look like target ones, while the discriminator must be able to discern between real-target and fake-target images. We implement both modules as fully convolutional networks.
The standard adversarial objective from [2] is of the form:

$$\begin{aligned}\mathcal{L}_{GAN}(G_{S\to T}, D_T, X_S, X_T) = &\mathbb{E}_{x_T\sim X_T}\left[\log\left(D_T(x_T)\right)\right] \\ &+ \mathbb{E}_{x_S\sim X_S}\left[\log(1 - D_T(G_{S\to T}(x_S)))\right]\end{aligned} \tag{4.2}$$

The optimization is performed in multiple stages, during which $G_{S\to T}$ aims to minimize the objective, whereas D_T concurrently seeks to maximize it. The optimization rule is the next:

$$\min_{G_{S\to T}} \max_{D_T} \mathcal{L}_{GAN}(G_{S\to T}, D_T, X_S, X_T) \tag{4.3}$$

Following the work of [62], we split the adversarial objective in two different losses (one for the discriminator, and one for the generator) in order to have a more stable learning process. The logarithm is replaced by a power-2 operator to avoid the saturation effects and vanishing gradients that appear when the original sigmoid cross-entropy

cost (equation 4.2) is used. The new objective functions (both to be minimized) now become:

$$\begin{aligned}\mathcal{L}_{GAN}^{dis}\left(G_{S\to T}, D_T, X_S, X_T\right) = \mathbb{E}_{x_S\sim X_S}\left[\left(D_T(G_{S\to T}(x_S))\right)^2\right] \\ + \mathbb{E}_{x_T\sim X_T}\left[\left(D_T(x_T)-1\right)^2\right]\end{aligned} \tag{4.4}$$

$$\mathcal{L}_{GAN}^{gen}\left(G_{S\to T}, D_T, X_S\right) = \mathbb{E}_{x_S\sim X_S}\left[\left(D_T(G_{S\to T}(x_S))-1\right)^2\right] \tag{4.5}$$

We group the two objectives in a single term that we denote as $\mathcal{L}_{adv}^{img}\left(G_{S\to T}, D_T, X_S, X_T\right)$. Adversarial frameworks are notoriously hard to train, but if the optimization is correctly carried out, then the generative module should produce convincing target-like samples.

With the just described generative process, we are able, in principle, to learn a joint source-target distribution starting from the marginal ones, which means that we can produce couples of images representing the same sample in both source and target styles. Unfortunately, since there is an infinite number of joint distributions that match the available marginal ones, we are not guaranteed that the mapping function we've found is preserving content structure and semantics. In other words, without additional constraints our translation network could completely disrupt the starting source image, still producing a new sample with target properties, but far from its original version.

We combine the adversarial losses 4.4 and 4.5 with supplementary objectives for a better safeguard of the input image appearance. Following the work of Zhu et al. [3], we introduce a cycle-consistency constraint: a target-to-source mapping $G_{T\to S}: X_T \to X_S$ is learnt through an additional adversarial generator-discriminator framework. The two generators $G_{S\to T}$ and $G_{T\to S}$ are then tied by the following loss term:

$$\begin{aligned}\mathcal{L}_{cycle}\left(G_{S\to T}, G_{T\to S}, X_S, X_T\right) = \mathbb{E}_{x_S\sim X_S}\left[\left\|G_{T\to S}\left(G_{S\to T}\left(x_S\right)\right) - x_S\right\|_1\right] \\ + \mathbb{E}_{x_T\sim X_T}\left[\left\|G_{S\to T}\left(G_{T\to S}\left(x_T\right)\right) - x_T\right\|_1\right]\end{aligned} \tag{4.6}$$

which basically guarantees structural content preservation by enforcing a perfect recovery of the input image with the joint application of both translations, i.e. we want $G_{S\to T}\left(G_{T\to S}\left(x_S\right)\right) \approx x_S$ and $G_{T\to S}\left(G_{S\to T}\left(x_T\right)\right) \approx x_T$. In figure 4.4 we report

(a) original (X_S) **(b)** adapted ($X_{S\to T}$) **(c)** reconstructed ($X_{S\to T\to S}$)

Figure 4.4: Example of the cycle-consistency constraint.

a visual example of the effect of the cycle-consistency constraint.
In addition, inspired by the work of Hoffman et al. [4], we impose semantic uniformity on images before and after the translation, by means of a semantic consistency loss:

$$\begin{aligned}\mathcal{L}_{sem}\left(G_{S\to T}, G_{T\to S}, X_S, X_T, F_S\right) = \mathcal{L}_{task}\left(F_S, G_{S\to T}(X_S), \rho(F_S, X_S)\right) \\ + \mathcal{L}_{task}\left(F_S, G_{T\to S}(X_T), \rho(F_S, X_T)\right)\end{aligned} \tag{4.7}$$

The intent is to exploit the segmentation network in its pre-trained version F_S from the first step to produce a semantic label of a generic image from both source or target domain by computing the $argmax$ of the probability map $F_S(X)$:

$$\rho\left(F_S, X\right) = \arg\max\left(F_S\left(X\right)\right) \tag{4.8}$$

The cross-entropy task loss $\mathcal{L}_{task}$ (equation 4.1) is then employed, where in place of the ground-truth label we introduce the semantic map $\rho(F_S, X)$ provided by F_S. The loss effect is no longer to promote correct segmentator predictions, but to force the generator to yield transformed images that are semantically identical to the original ones when viewed under the F_S scope. This scheme allow us to compute a measure of semantic distance in both domains, without resorting to ground-truth information. On the other side, F_S is not a perfect predictor (not even a good one in the target domain), therefore an excessive emphasis on this loss could introduced undesired artifacts in the generative process.
As a further objective, we introduce the identity loss [36] with regularization purposes:

$$\begin{aligned}\mathcal{L}_{iden}\left(G_{S\to T}, G_{T\to S}, X_S, X_T\right) = \mathbb{E}_{x_S\sim X_S}\left[\|\left(G_{T\to S}\left(x_S\right)\right) - x_S\|_1\right] \\ + \mathbb{E}_{x_T\sim X_T}\left[\|\left(G_{S\to T}\left(x_T\right)\right) - x_T\|_1\right]\end{aligned} \tag{4.9}$$

It obliges the generators to act like identity mappings when supplied with input images already from the output domain. The underlying goal is to encourage the generative process to preserve colors.

By grouping all the losses that we've introduced above in a single objective to be minimized with respect to all the generative framework parameters, we get:

$$\begin{aligned}
&\mathcal{L}_{tot}\left(G_{S\to T}, G_{T\to S}, D_T, D_S, X_S, X_T\right) \\
&\quad = \lambda_{img}\cdot\mathcal{L}_{adv}^{img}\left(G_{S\to T}, D_T, X_S, X_T\right) + \lambda_{img}\cdot\mathcal{L}_{adv}^{img}\left(G_{T\to S}, D_S, X_T, X_S\right) \\
&\quad + \lambda_{cycle}\cdot\mathcal{L}_{cycle}\left(G_{S\to T}, G_{T\to S}, X_S, X_T\right) + \lambda_{sem}\cdot\mathcal{L}_{sem}\left(G_{S\to T}, G_{T\to S}, X_S, X_T, F_S\right) \\
&\quad + \lambda_{iden}\cdot\mathcal{L}_{iden}\left(G_{S\to T}, G_{T\to S}, X_S, X_T\right)
\end{aligned} \tag{4.10}$$

where the λ terms are included to balance the several individual components. In Chapter 5 we will discuss more in detail how we set the values for the different loss weights.

If the image-to-image translation framework is successfully trained, we should have access to a source-to-target translator $G^*_{S\to T} = \arg\min_{G_{S\to T}} \mathcal{L}_{tot}$ capable of producing target-like samples still provided with annotations from source domain, thanks to semantic consistency constraints. Now we can apply the cross-entropy loss similarly to the first step, but this time using the adapted source samples:

$$F_T = \arg\min_F \mathcal{L}_{task}\left(F, G^*_{S\to T}(X_S), Y_S\right) \tag{4.11}$$

As a result of this third step, the new predictor F_T should have improved performance on the target domain and we should have reduced the gap with a totally supervised target-based training.
As described in section 4.1.3, an additional adversarial framework is proposed to improve the adaptation performance, with the introduction of two new discriminators

D_S^{feat} and D_T^{feat} and the relative adversarial losses similarly to the pixel-level case:

$$\begin{aligned}\mathcal{L}_{GAN}^{dis-feat}(F', G_{S\to T}, D_T^{feat}, X_S, X_T) = \mathbb{E}_{x_S\sim X_S}\left[\left(D_T^{feat}(F'(G_{S\to T}(x_S)))\right)^2\right] \\ + \mathbb{E}_{x_T\sim X_T}\left[\left(D_T^{feat}(F'(x_T)) - 1\right)^2\right]\end{aligned} \tag{4.12}$$

$$\mathcal{L}_{GAN}^{gen-feat}(F', G_{S\to T}, D_T^{feat}, X_S) = \mathbb{E}_{x_S\sim X_S}\left[\left(D_T^{feat}(F'(G_{S\to T}(x_S))) - 1\right)^2\right] \tag{4.13}$$

where we denote with F' the first section of the segmentator (i.e. only the MobileNet backbone encoder without the decoder module) when utilized for feature extraction and not for semantic prediction (we report only the source-to-target loss terms).
Again, we group the pairs of adversarial objectives in single terms that we denote as $\mathcal{L}_{adv}^{feat}(G_{S\to T}, D_T^{feat}, X_S, X_T)$ and $\mathcal{L}_{adv}^{feat}(G_{T\to S}, D_S^{feat}, X_T, X_S)$. D_S^{feat} and D_T^{feat} have to discriminate between feature representations of original and transformed images from the same domain, while generators have to fool them as usual.
With the introduction of the feature adaptation framework, we also try to combine the second and third optimization steps in order to perform the segmentator fine-tuning in conjunction with the training of the generative module. The full objective now becomes:

$$\begin{aligned}&\mathcal{L}'_{tot}(F, G_{S\to T}, G_{T\to S}, D_T, D_S, D_T^{feat}, D_S^{feat}, X_S, X_T) \\ &\quad= \lambda_{img}\cdot\mathcal{L}_{adv}^{img}(G_{S\to T}, D_T, X_S, X_T) + \lambda_{img}\cdot\mathcal{L}_{adv}^{img}(G_{T\to S}, D_S, X_T, X_S) \\ &\quad+ \lambda_{feat}\cdot\mathcal{L}_{adv}^{feat}(F', G_{S\to T}, D_T^{feat}, X_S, X_T) + \lambda_{feat}\cdot\mathcal{L}_{adv}^{feat}(F', G_{T\to S}, D_S^{feat}, X_T, X_S) \\ &\quad+ \lambda_{cycle}\cdot\mathcal{L}_{cycle}(F, G_{S\to T}, G_{T\to S}, X_S, X_T) + \lambda_{sem}\cdot\mathcal{L}_{sem}(G_{S\to T}, G_{T\to S}, X_S, X_T, F_S) \\ &\quad+ \lambda_{iden}\cdot\mathcal{L}_{iden}(G_{S\to T}, G_{T\to S}, X_S, X_T) + \lambda_{task}\cdot\mathcal{L}_{task}(F, G_{S\to T}(X_S), Y_S)\end{aligned} \tag{4.14}$$

The complete framework is more prone to unstable behaviours, given that in early stages F is trained with images produced by generators not yet in their final versions. On the other hand, an improved F inside the generative module should help the semantic loss to promote a more reliable and less noisy semantic consistency, thanks to more accurate predictions.

5 Results

5.1 Implementation Details

The proposed domain adaptation framework that we introduced in section 4 is developed using TensorFlow 1.13. We resort to pre-existent CycleGAN [3] and MobileNetV2 [1] implementations, which are modified and adapted in order to build our final model.
The model hyper-parameters are selected with a preliminary tuning procedure. Note that it is not guaranteed that the values we find are the best ones, since an optimal solution entails a joint optimization over the entire parameters space, which would require much more computational and time resources than what we have at our disposal. For this reason, we base our search on values proposed by the original works, which we then modify in order to adapt them to our scenario.

The semantic segmentation network is trained by means of an Adam optimizer [6] with β_1, the exponential decay rate for the first moment estimates, set to 0.9, and a weight decay of 4×10^{-5}. The learning rate is subjected to a polynomial decay of power 0.9, and it is decreased to 0 from its initial value by the end of the training procedure. In section 5.3.1 we will discuss more in detail how segmentation performance are affected by different starting points for the learning rate.

As described in section 4.1.2, the feature extractor backbone is augmented with ASPP and decoder modules to recover the input dimensionality. On this regard, we select an output stride of 16 for the feature extractor, whereas for the ASPP block we choose atrous rates of 6, 12 and 18 as suggested by the original paper [21]. We speed up the training procedure of the ASPP and decoder modules by boosting their learning rate of a factor 10, since initially we don't have access to pre-trained weights for them as we do for the rest of the network. When the segmentation network is fine-tuned, we no longer apply this multiplicative factor and, at the same time, we decrease the learning rate. Moreover, we start with a version of the MobileNetV2 which has been pre-trained on the Pascal VOC 2012 dataset [63] to improve efficiency and speed up the training process.

The CycleGAN framework optimization is based on the Adam method [6] as well. In particular, we keep a small β_1 term equal to 0.5 as in the original paper. It makes the training process more unstable, but we notice no improvements by changing it. The learning rate is fixed to 2×10^{-4} and kept constant for the entire training time. For additional stability, the image-level discriminators are updated using a history of generated images. Following [3], two buffers with 50 transformed samples each are built (one per domain), and each discriminator is provided with the latest generator output or with a stored image with equal probability.

As concerns the generative objective $\mathcal{L}_{tot}$, we make several tests trying to balance the different components. We notice that in correspondence of $\lambda_{img} = 1$, $\lambda_{cycle} = 20$ and $\lambda_{sem} = 0.1$ we get the best domain adaptation results. The $\mathcal{L}_{iden}$ term seems to be beneficial only to produce better looking images, with no effects on the overall adaptation process.
With the introduction of the feature framework and the full objective $\mathcal{L}'_{tot}$, we set λ_{feat} to small values and we start to observe the regularization effect of $\mathcal{L}_{iden}$ on the adaptation process. In addition, we set $\lambda_{task} = 1$.
Further insights on this subject will be reported and discussed in the next sections.

We train our domain adaptation model with a single GPU, a NVDIA GeForce GTX 1080 Ti, which has a total dedicated memory of 11GB. The large amount of model parameters, combined with the high resolution images from source and target sets,

prevent us from using full resolution data for training purposes. To overcome this issue, we extract random patches from training samples and we use them as model inputs. Moreover, we set a batch size of 5 when training the semantic segmentator alone in the first step. In all the other phases, where multiple deep neural networks have to fit the gpu memory at the same time, we have to reduce the batch size up to 1 to fulfil hardware constraints.

5.2 Datasets

A fundamental ingredient in the developing process of a machine learning algorithm is the availability of training data, that is the source from which the model should be able to extract the necessary information to solve its task. In the context of semantic segmentation, a supervised training procedure requires the availability of pairs of images and corresponding dense labels carrying pixel-level annotations, which usually are extremely expensive to produce. As extensively discussed in previous sections, the combination of annotation shortage and poor generalization skills of deep neural networks, pushes us to undertake the unsupervised domain adaptation task.

To place us in an appropriate framework, we select a source domain with easily accessible labelled samples that can be obtained in large quantities, which is the case of synthetic imagery. Synthetic data not only comes with essentially free annotations and a much lower collection cost than real data thanks to an automatic generation, but also in principle provides a total control over the virtual environment in terms of point of view, illumination, objects inside the scene, etc. This feature allows a careful design of the synthetic dataset properties, to guarantee that it contains a proper amount of useful information.
On the opposite side, we refer to real imagery as the target domain. Nowadays real-world photos are fairly easy to get in any situation, therefore we realistically expect no particular limitations on their availability. In addition, we assume no ground-truth information is accessible in any form for real data (if not for comparison or evaluation purposes), so that we really are in an unsupervised scenario.

As concerns the specific datasets on which we train and validate our framework, we choose the publicly accessible GTA5 [64] and Cityscapes [65] ones. They are built specifically to address semantic segmentation of urban scenes, which is of strategic importance in autonomous driving. The task is quite challenging, since multiple objects of different sizes and semantic categories have to be recognized with high confidence.

The Cityscapes dataset [65] contains high-resolution images of 2048×1024 pixels collected from a vehicle moving across the streets of 50 cities in central Europe. Starting from the hundreds of thousands of frames captured in video sequences during the acquisition process, only a subset of 5000 pictures is manually labelled with dense pixel-level annotations. The annotated batch is further split into groups of 2975, 500 and 1525 images for respectively training, validation and test purposes. We train and test our model on the 3475 samples from the training and validation subsets, which are the only publicly available. The dataset is provided with a total 34 different classes, but we employ only 19 of them in order to match the GTA5 object categories.
Urban scenes captured by Cityscapes images involve the typical dense traffic of European cities, with a strong presence of pedestrians and in general a crowded environment. Additionally, they are characterized by a gray tone due to the peculiar appearance of the urban landscape of the acquisition sites.

Figure 5.1: Examples of Cityscapes images.

The GTA5 [64] is a synthetic dataset composed of 24966 annotated images, with resolution of 1914×1052 pixels and a total of 19 semantic categories compatible with the Cityscapes dataset. Frame acquisition is performed inside the *Grand Theft Auto V* (GTA5) open-world computer game: similarly to the Cityscapes dataset, images are collected from a vehicle point of view, but the scenario is a virtual city resembling Los Angeles in Southern California. The high realism and fidelity of the game en-

vironment in terms of object textures and scene layout ensures the generation of photorealistic images. At the same time it is possible to extrapolate semantic information from scene rendering to efficiently produce dense annotations on synthesized images. This two elements combined have been crucial for the development of the GTA5 large scale semantic segmentation dataset.
In our experiments we use only half of the entire dataset to train the domain adaptation model (12483 images). Furthermore, we select 500 samples from the remaining set for validation purposes.
From a visual analysis of the dataset, images appear to have quite vivid tones (which is typical of video games), while the urban environment they depict is typically suburban with few pedestrians and a wide and diverse landscape.

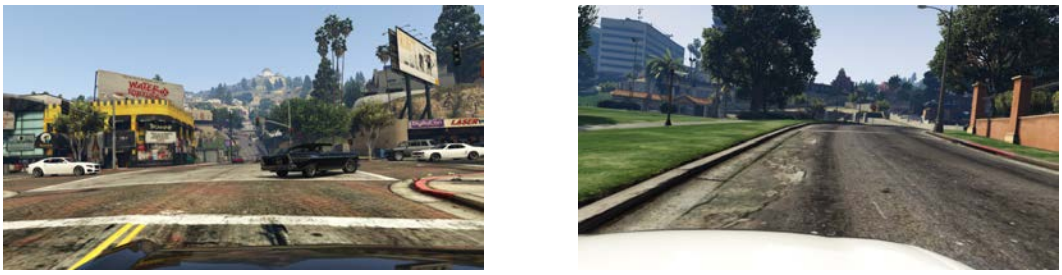


Figure 5.2: Examples of GTA5 images.

5.3 Experiments

The purpose of the following sections is to show the results of each intermediate step of our unsupervised domain adaptation framework and, in the end, to prove its effectiveness. First, we discuss the no adaptation scenario, in which the segmentation network is source-supervisedly trained on the GTA5 dataset before being validated on the Cityscapes dataset with no further adjustments. Next, the generative adaptation strategy is analyzed, in both the visual results from the image-to-image translation stage and the performance boost on semantic predictions provided by domain adaptation. Finally, we will describe the outcomes of our attempts to further improve our framework by introducing feature adaptation and unifying the model optimization in an end-to-end training process.

5.3.1 Training on Synthetic Data

In this section we analyze the optimization process that is done prior to the actual domain adaptation. We train the segmentation network as discussed in section 4.2, with just a standard cross-entropy loss and labelled frames from the GTA5 dataset. We don't use full-size images during the training stage due to memory constraints, and we first perform a random crop to size 800×800 before feeding them to the predictor. For this reason, we are able to utilize a batch of 5 images. To measure the performance achieved by the segmentation network, we test it on the Cityscapes and GTA5 validation sets, both composed of 500 images. During the testing phase, input samples are kept at their original size and prediction maps are upsampled to match the spatial dimension of the corresponding annotations. For comparison purposes, we also perform a target-based optimization, for which the network is directly supplied with annotated target samples from the Cityscapes dataset in both training and testing phases and therefore no adaptation is needed.
The accuracy of the segmentation network in terms of intersection over union (IoU) on the Cityscapes validation set is reported in table 5.1 (we list the highest measured values). We explore 4 different configurations with regard to learning rate and type of supervision. In particular, we consider 3 distinct starting points for the learning rate (namely $10^{-4}, 10^{-5}, 10^{-6}$), whereas we directly report the target upper bound provided by a target-based optimization.

Method	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic Light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	**MeanIoU**
Synth. ($lr = 10^{-6}$)	7.5	13.0	66.1	1.1	0.0	0.5	0.0	0.0	71.8	6.0	50.8	22.3	0.0	62.5	4.5	0.0	0.0	0.0	0.0	16.1
Synth. ($lr = 10^{-5}$)	14.8	13.0	53.6	4.2	10.7	15.9	2.5	0.0	74.1	8.3	54.0	39.6	2.9	56.7	6.8	6.0	0.0	0.1	0.0	19.1
Synth. ($lr = 10^{-4}$)	71.6	20.1	72.7	15.3	18.6	10.0	5.7	4.2	73.5	11.9	42.5	37.7	0.4	71.4	3.4	25.5	0.0	2.3	0.1	25.6
Real ($lr = 10^{-4}$)	96.6	76.3	87.7	43.1	49.2	42.5	48.9	60.5	88.2	55.9	90.2	70.3	47.0	87.5	51.9	71.5	60.1	44.8	66.6	65.19

Table 5.1: Results in terms of IoU on the Cityscapes validation set for different training settings. $Synthetic$ corresponds to a source-based optimization, while $Real$ to a target-based optimization.

Plots of training error, validation error on target data and mean IoU on both target and source domains in correspondence of a source-based network training with different learning rates are shown in figures 5.3, 5.4, 5.5 and 5.6 respectively. We train the

network for a total of 90000 steps, but we limit the curves to an interval comprising the initial 35000 (except for figure 5.6 where we consider all the 90000 steps).
The first thing we notice (table 5.1) is that there is a huge gap in terms of mean IoU (around 40% at least) between the case in which training and validation images belong to separate datasets and when instead a standard supervision with ideally the same train and test data distributions is performed. This is a perfect example of how deep models lack of generalization skills and suffer from a consistent drop of performance in presence of domain shift.

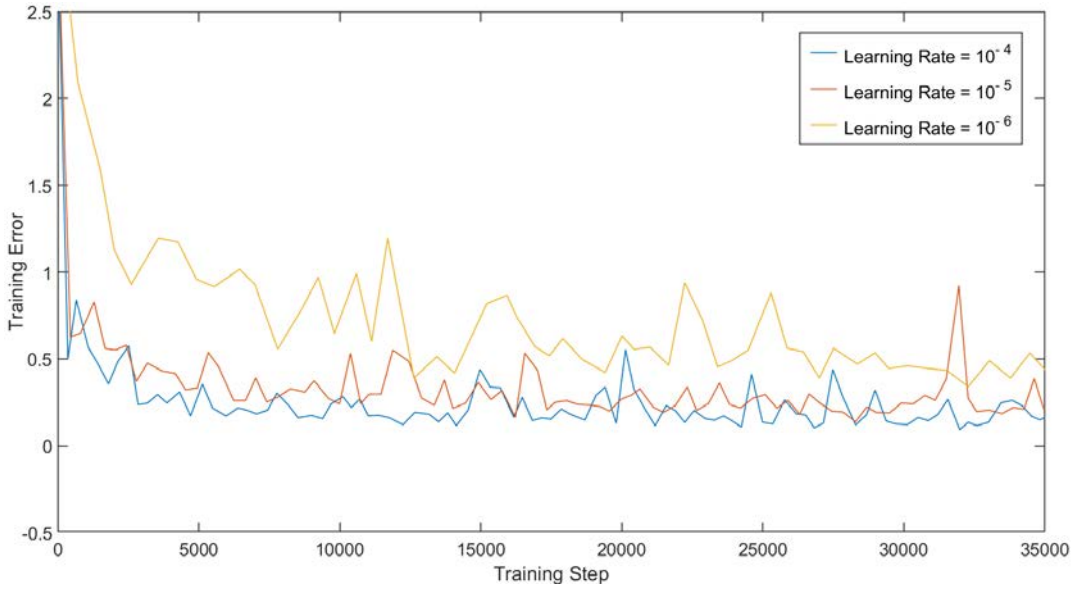


Figure 5.3: Training error on synthetic data (GTA5 dataset). Training is performed on synthetic data (GTA5 dataset).

Moreover, while the training process in case of target supervision is converging to an optimal parameter configuration, in case of source supervision we observe a domain overfitting trend (figure 5.4). In other words, the predictor in the first steps is capable of quickly learning meaningful representations that can be safely shared among the two domains, since they are related to general properties which characterize any urban scene. However, when the segmentator starts to improve its accuracy on the source domain, it does so by focusing on attributes specific to that domain, but it concurrently harms its understanding of samples with similar semantic content of training ones but different appearance. Other than that, the absence of convergence on the target domain forces us to choose a specific point where to stop the training process. For a correct decision, we should look at the mean IoU curve computed over

the Cityscapes dataset in correspondence to a GTA5-based training (figure 5.5). We report the behaviour of the mean IoU with different values of the learning rate, for it has shown to highly influence the network accuracy.

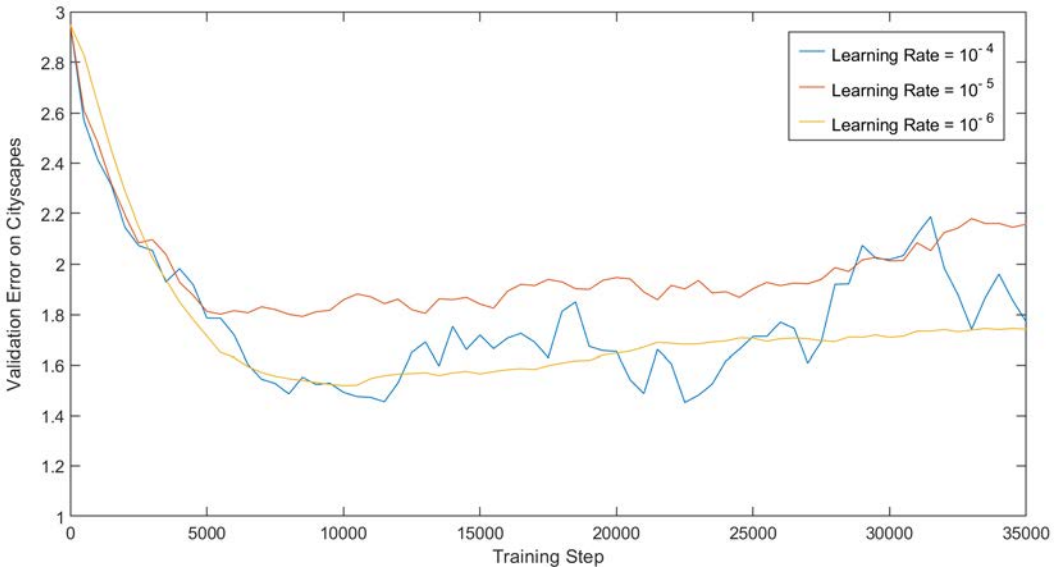


Figure 5.4: Validation error on real data (Cityscapes dataset). Training is performed on synthetic data (GTA5 dataset).

We observe that learning rates of 10^{-4} and 10^{-5} share the best trends, but the first involves a much higher instability. A reasonable choice would be to select the more reliable and stable solution, even though the other one reaches a higher meanIoU (table 5.1). Unfortunately, if we move our attention to the curves representing the training error (figure 5.3) and the meanIoU computed on source data (figure 5.6), we can see that a bigger learning rate entails a noticeable performance gain (around 20% in terms of mIoU between 10^{-5} and 10^{-4} values). With further investigations, we notice that the accuracy drop on the source domain is strongly detrimental to the overall adaptation performance and this is due to the fact that the semantic segmentator inside the generative module operates in both domains. For this reason we choose a learning rate of 10^{-4}. Furthermore, we perform additional tests and measure small variations on the final adaptation results when changing the mean IoU peak (on the Cityscapes validation set) for which we save the network parameters. This is positive, as the unstable pre-training phase doesn't affect the overall domain adaptation performance of the entire framework.
Finally, we pick a subset of 500 images from the Cityscapes training set (which the

network has never seen in a GTA5-based training phase) and use them to test the network prediction accuracy when the checkpoint relative to the 25.62% mean IoU peak is selected. We get an even improved mean IoU of 27.0%, showing that the fluctuations of validation results are due to an unstable optimization rather than a validation set not correctly matching the general properties of the Cityscapes dataset.

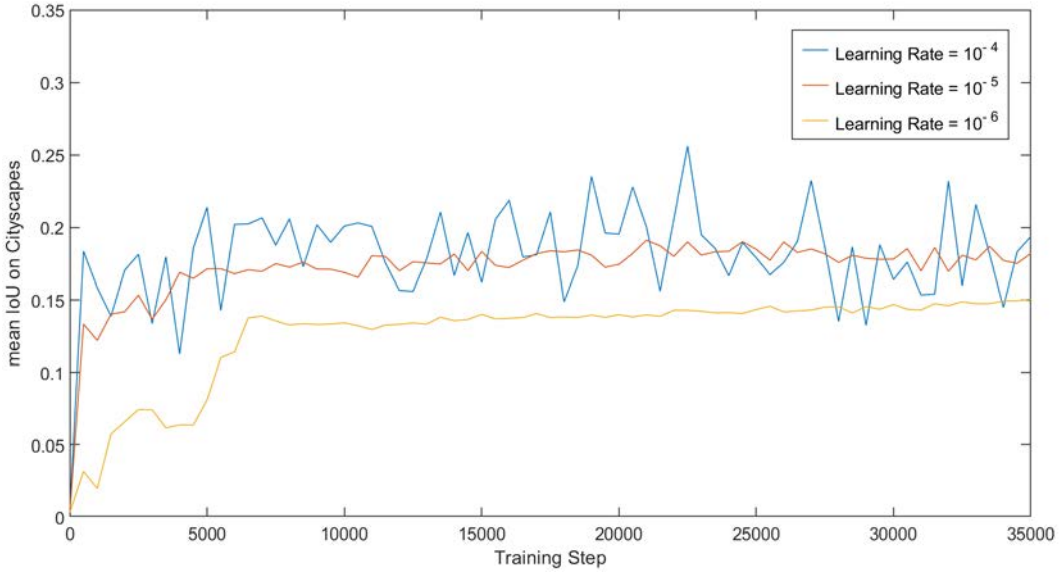


Figure 5.5: Mean IoU on real data (Cityscapes dataset). Training is performed on synthetic data (GTA5 dataset).

In figure 5.7 we show some visual results in terms of semantic prediction maps to relate the optimization in the standard supervised setting to the case in which a domain discrepancy appears between training and test phases due to a change of dataset. The segmentator with source supervision and no adaptation clearly struggles to correctly classify large portions of the target input image relative to some major classes such us the road and the sidewalk. This means that the segmentator is not able to recognize common structures due to textures and color changes across domains. Other than that, details are completely lost and objects from classes with reduced occurrences are easily misclassified. There is no doubt that an adaptation process can be strongly beneficial.

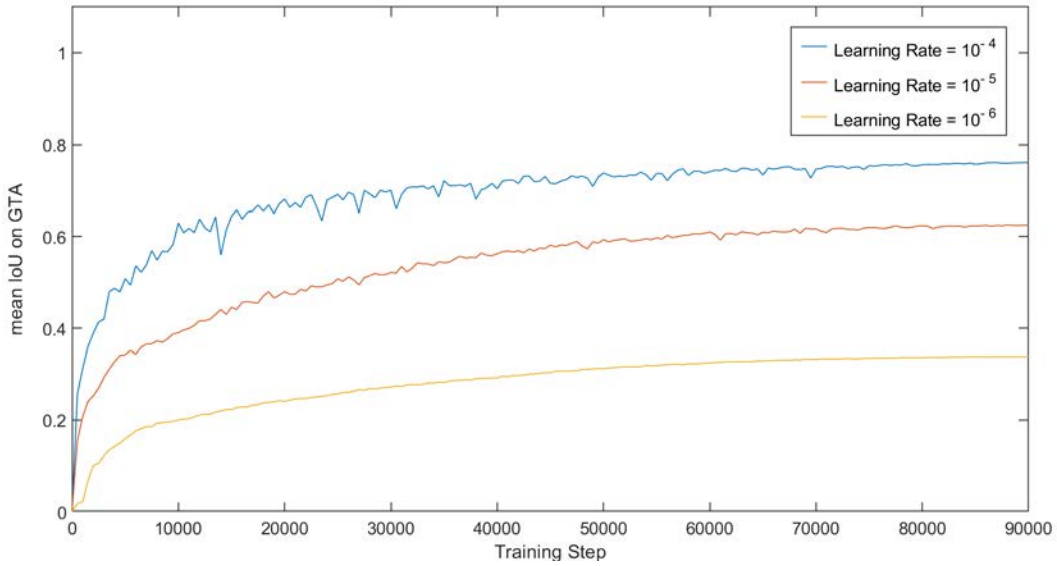
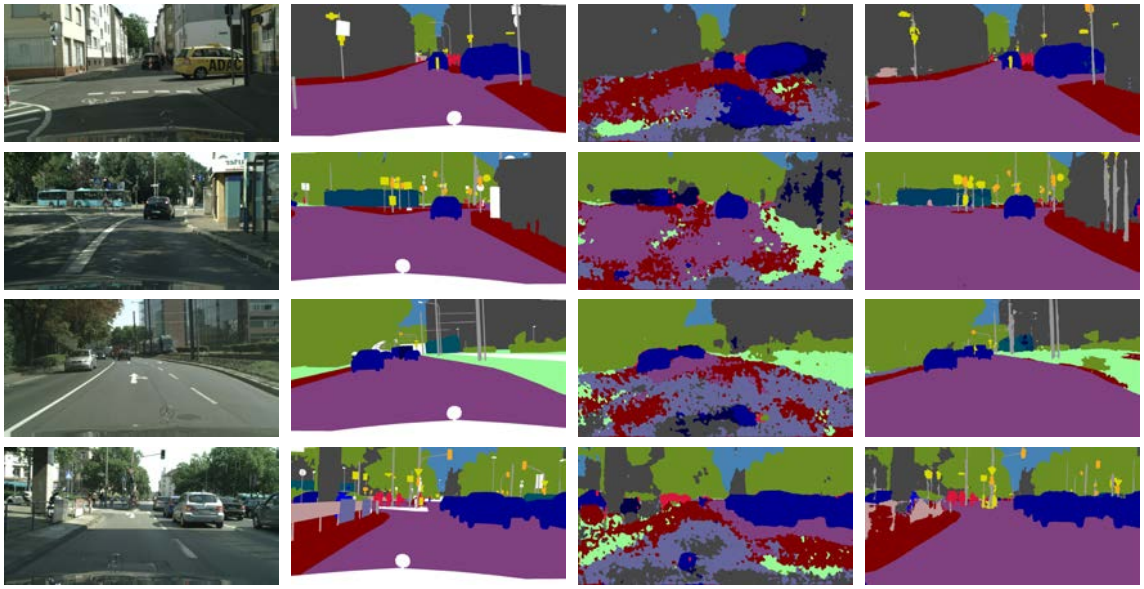


Figure 5.6: Mean IoU on synthetic data (GTA5 dataset). Training is performed on synthetic data (GTA5 dataset). We plot the curve in an interval comprising all the 90000 training steps to better show how the optimization converges when no domain discreapancy is present between training and test phases.



(a) Target image **(b)** Ground-truth **(c)** Source supervision **(d)** Target supervision

Figure 5.7: Segmentation maps of real images from the Cityscapes validation set when source or target supervision is provided (learning rate equal to 10^{-4}).

5.3.2 Generative Framework

The core of our domain adaptation framework is represented by the image-to-image translation module. As extensively discussed in previous sections, it is made of a couple of GANs working in parallel on separated domains and tied by a cycle-consistency constraint. In addition, a pre-trained semantic segmentator is used to enforce semantically consistent translations.
We provide the source generator with synthetic images from the GTA5 dataset and the target generator with real images from the Cityscapes dataset. In this phase, semantic label maps are ignored in both source and target domains. Moreover, the optimization process of the generative module is unsupervised, since we don't have access to pairs of corresponding GTA5 and Cityscapes frames. We don't use full-size images due to memory constraints, and we perform a random crop before supplying them to the generators and discriminators. In particular, we first downsample input images so that they have a width equal to 1024 (while preserving the aspect ratio) and then we randomly select patches of size 400×400 to be fed to the network. This effectively reduces memory occupation allowing us to train the framework with a batch size of 1. The alternative would be to directly downsample input data until hardware constraints are met, but the resolution drop would destroy useful details and, in turn, hurt the generative performance.
We train the generative module for a total of 60000 steps, since it seems to be a sufficiently long interval to reach a stable point.

In figures 5.8, 5.9, 5.10 and 5.11 we report the plots of the adversarial generator and discriminator errors ($\mathcal{L}_{GAN}^{gen}$ and $\mathcal{L}_{GAN}^{dis}$) and of the cycle-consistency and semantic losses ($\mathcal{L}_{cycle}$ and $\mathcal{L}_{sem}$). To remove the strong oscillations in the original curves, we apply a smoothing function that highlights the general trend. The intense variations are probably caused by a batch size of 1 and a small β_1 value (0.5) of the Adam optimizer, which controls the exponential decay rate for the first moment estimates of the gradient.
From figures 5.8 and 5.9 we can see that the adversarial system in both domains is not perfectly balanced, as the discriminator errors are lower than the generator counterparts. Training a single GAN is already a challenging task, training simultaneously two of them to translate high resolution fully detailed images makes the task

even more difficult. On the other hand, the cycle-consistency loss (figure 5.10) seems to effectively tie the two adversarial processes, and that is reflected by a common generator-discriminator behaviour across the source and target domains. Moreover, the cycle-consistency loss appears to require more training time to stabilize with respect to the adversarial losses, justifying our choice to extend the optimization up to 60000 steps.

We report the plot of the semantic loss for both domains and different loss weights (λ_{sem}) to highlight the inherent asymmetry brought by the segmentation network. On this regard, it is possible to notice that the source semantic loss, which forces semantic consistency between source original and generated images, has lower values than the target counterpart. This because, as discussed in the previous section, the semantic segmentator, which is pre-trained on synthetic data, has a poor prediction accuracy on the real domain. Additionally, a higher λ_{sem} leads to a lower semantic error, which, in turns, corresponds to a stronger impact of the semantic segmentator on the generative process.

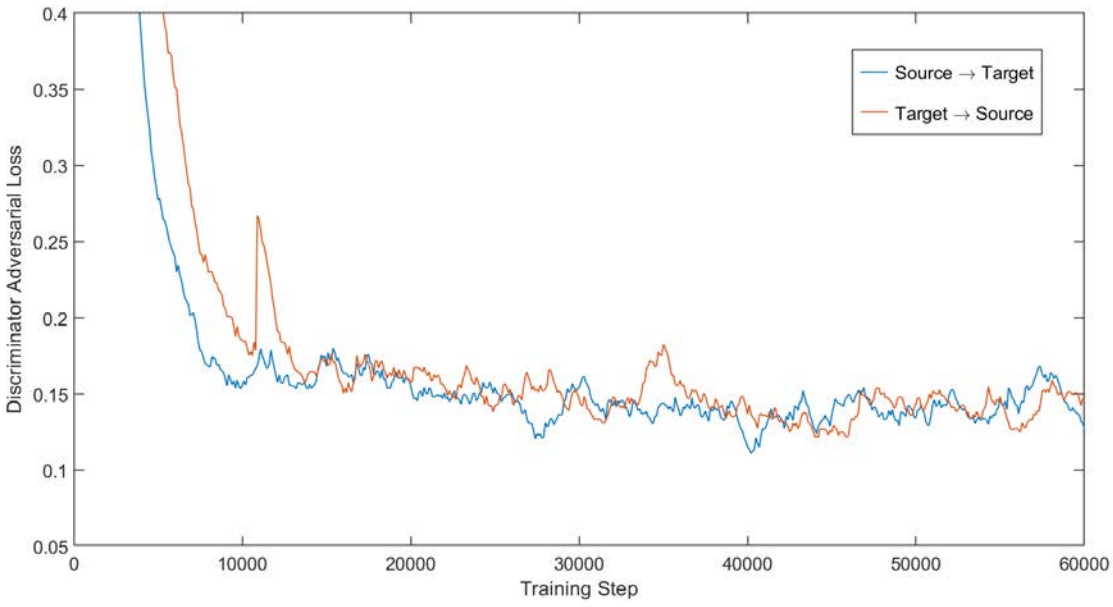


Figure 5.8: Discriminator adversarial loss $\mathcal{L}_{GAN}^{dis}$ for both source-to-target and target-to-source translations.

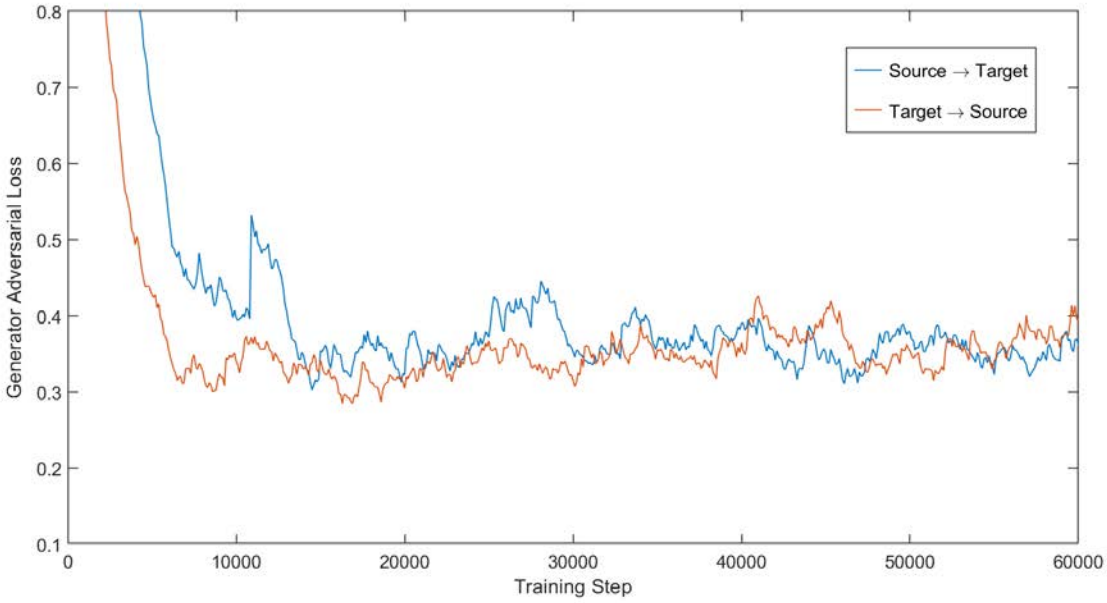


Figure 5.9: Generator adversarial loss $\mathcal{L}_{GAN}^{gen}$ for both source-to-target and target-to-source translations.

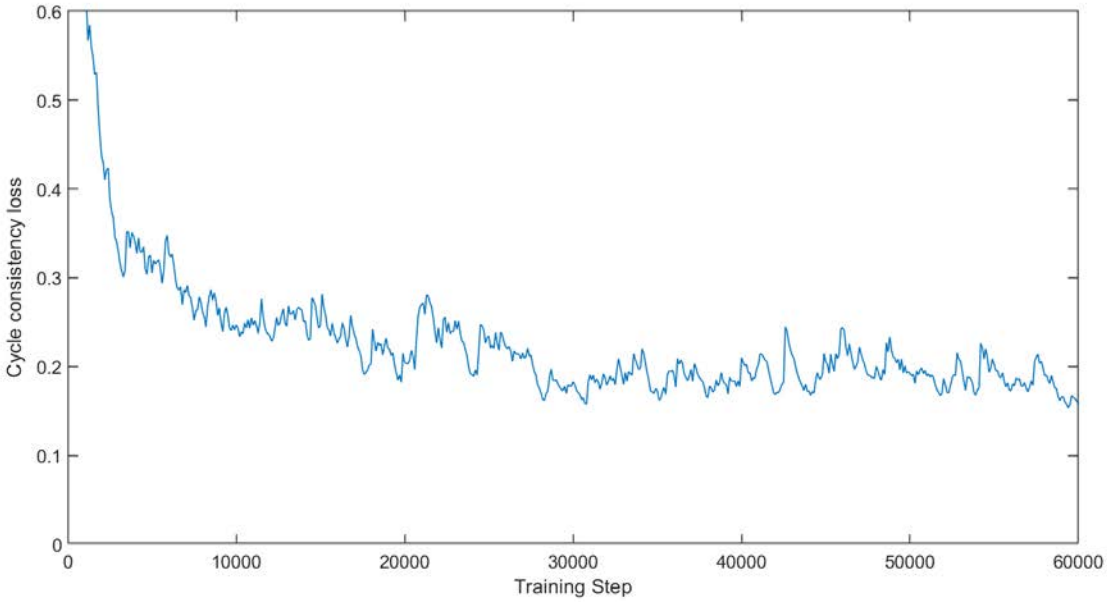


Figure 5.10: Cycle-consistency loss $\mathcal{L}_{cycle}$ computed during the training of the original CycleGAN model ($\lambda_{sem} = 0$).

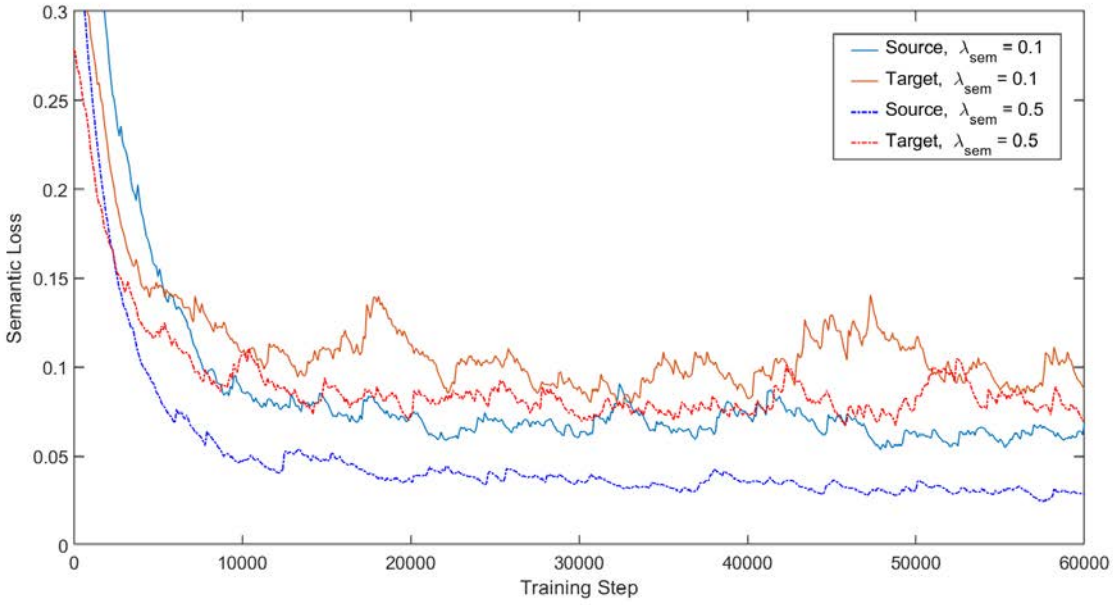


Figure 5.11: Semantic loss $\mathcal{L}_{sem}$ reported for both domains and for $\lambda_{sem} = 0.1, 0.5$.

In figures 5.12 and 5.13 we show some examples of the effect of the cycle-consistency constraint on both the GTA5-to-Cityscapes and Cityscapes-to-GTA5 branches. Images on the left correspond to the originals, whereas the second and third columns illustrate respectively the first cross-domain translation and the second transformation that brings inputs back to their initial state. We can see that color and textures modified by the first stage are almost completely recovered during the reconstruction. For example, the road appearance, which is one of the elements that better differentiate the two datasets, is mostly restored by the reprojection.
In figures 5.14 and 5.15 we report some visual results of the image-to-image translation process across the GTA5 and Cityscapes datasets in both source-to-target and target-to-source directions. In particular, we show the different outcomes when λ_{sem} is set to 0 (standard CycleGAN), 0.1 or 0.5. The translation function applied on the GTA5 dataset is mostly adding a grey shade to the input samples, as Cityscapes images are mainly characterized by grey tonalities. In addition, it is possible to notice some artifacts appearing in the sky introduced by the generative process, due to a not perfectly balanced adversarial competition. On the other side, source-like target images originating from the Cityscapes dataset acquire more vivid and bright tones, in order to match the look of a typical GTA5 urban scene. Moreover, road texture of

input images is always subject to perceptible changes, confirming that it is a distinctive dataset attribute. The purpose of the semantic consistency loss is to preserve the semantic content of images during their translation. Nonetheless, we observe that, as an additional effect, the loss helps also to preserve image appearance. On this regard, we notice that a bigger value of λ_{sem} entails a minor presence of visible alterations and an improved color preservation. This is especially true in the GTA-to-Cityscapes scenario, but can be easily perceived also in the other direction. For example, an unpleasant visual artifact that completely alters the color of some red or blue image components in the $\lambda_{sem} = 0, 0.1$ cases is solved by encouraging a stronger semantic consistency ($\lambda_{sem} = 0.5$).

(a) original (GTA) **(b)** adapted (GTA → CS) **(c)** reconstructed (GTA → CS → GTA)

Figure 5.12: Examples of the cycle-consistency constraint on the synthetic-to-real model branch.

(a) original (CS) **(b)** adapted (GTA → CS) **(c)** reconstructed (CS → GTA → CS)

Figure 5.13: Examples of the cycle-consistency constraint on the real-to-synthetic model branch.

(a) Original **(b)** CycleGAN only **(c)** $\lambda_{sem} = 0.1$ **(d)** $\lambda_{sem} = 0.5$

Figure 5.14: Examples of GTA5 images adapted to the Cityscapes dataset with different generative settings, namely with various semantic loss weights inside the generative objective.

(a) Original **(b)** CycleGAN only **(c)** $\lambda_{sem} = 0.1$ **(d)** $\lambda_{sem} = 0.5$

Figure 5.15: Examples of Cityscapes images adapted to the GTA5 dataset with different generative settings, namely with various semantic loss weights inside the generative objective.

5.3.3 Fine-tuning on Adapted Data

Once the image-to-image translation module is trained, we have access to a generator capable of projecting images from the GTA5 to the Cityscapes dataset, that is from the source to the target domain. We exploit the source-to-target translator by appending it to the input of the pre-trained semantic segmentator, which is then fine-tuned on adapted source data. These synthesized target-like images should still carry accurate semantic information in the form semantic labels coming from their original source versions. This because the semantic loss embedded inside the image-to-image translation model promotes a generative process that preserves the semantic content of input data. Moreover, the CycleGAN framework itself is suited for texture and color changes rather than more radical alterations to the input structure.
We supply the generator-segmentator module with synthetic images from the GTA5 training set. As discussed in the previous section, hardware constraints force us to perform a resize of input data to reduce memory occupation. We again opt for the extraction of random patches of size 400×400 on a resized version of the input.
The segmentator is trained for a total of 50000 steps, while the initial learning rate is decreased to 10^{-5}, leading to a more stable optimization. The generator, instead, is kept fixed and used as a preprocessing function on input data. The segmentation results in terms of per-class and mean IoU evaluated on the Cityscapes validation set are shown in table 5.2. More specifically, we compare the adaptation performance achieved by adopting different values for λ_{sem}, namely 0, 0.1 and 0.5. In this way we can determine the impact of the semantic consistency constraint over the final adaptation results. In addition, we report from section 5.3.1 the IoU values computed in the no adaptation scenario and in case of a target-based supervised training (upper bound).
As first thing, we notice that our generative adaptation technique (with and without semantic loss) introduces a consistent boost to the semantic segmentator prediction accuracy with respect to a simple source-based optimization. When the best value for λ_{sem} is selected (0.1), we find an improvement of 10% of the mean IoU, going from 25.6% to 35.5%. More importantly, the performance increase occurs for almost all the semantic classes. On this regard, the adaptation process is capable to enhance the accuracy on semantic components with small occurrences, which the segmentator particularly struggles to capture in presence of domain shift, but also leads

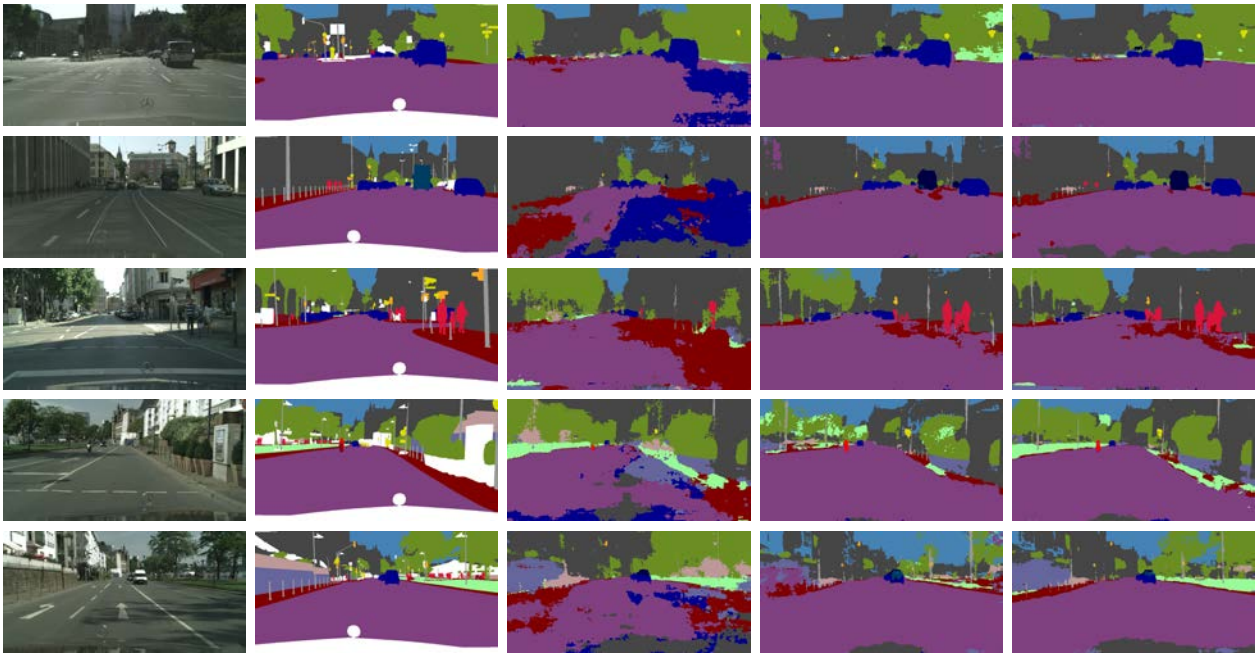
to more accurate predictions on frequent and dominant classes that are recognized with sufficient precision even with no adaptation. For example, classes such as *pole*, *traffic light* and *traffic sign* involving small object detection are affected by the most noticeable prediction enhancement. At the same time, the adapted segmentator performs better on common categories, such as *road*, *car*, *building* and *vegetation*. Another class that benefits from domain adaptation is the *truck* one (with strong semantic similarity to the *car* and *bus* categories), confirming that our framework effectively provides the predictor with additional information to solve its task.

Method	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic Light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	**MeanIoU**
Synth.	71.6	20.1	72.7	15.3	18.6	10.0	5.7	4.2	73.5	11.9	42.5	37.7	0.4	71.4	3.4	25.5	0.0	2.3	0.1	25.6
$\lambda_{sem} = 0$	86.6	37.7	76.2	18.3	11.7	27.2	17.9	20.8	68.2	10.9	53.7	51.0	16.7	76.5	18.8	21.7	9.5	8.7	0.6	33.3
$\lambda_{sem} = 0.5$	84.2	30.1	79.2	18.5	13.8	20.1	14.7	14.3	81.1	19.6	67.3	53.1	15.2	79.2	22.0	28.0	2.5	9.8	0.1	34.4
$\lambda_{sem} = 0.1$	87.1	39.9	79.0	21.3	14.0	22.8	16.0	16.0	80.7	25.4	68.0	52.2	14.3	77.0	19.3	29.8	3.9	8.0	0.6	35.5
Real	96.6	76.3	87.7	43.1	49.2	42.5	48.9	60.5	88.2	55.9	90.2	70.3	47.0	87.6	51.9	71.5	60.2	44.8	66.6	65.2

Table 5.2: Results in terms of IoU on the Cityscapes validation set for different adaptation settings, namely with various semantic loss weights inside the generative objective.

Moving forward, we observe improved adaptation results when the semantic loss is added to the generative objective ($\lambda_{sem} = 0.1, 0.5$). To that end, the increment in terms of mean IoU appearing with the introduction of the semantic consistency constraint (about 2%) shows that the segmentator inserted inside the generative framework truly helps to preserve useful information during image translations. However, the improvement is not well distributed among all the classes. In particular, more confident classes are boosted, while less confident ones are penalized. An imperfect predictor, in fact, could induce some inconsistencies in the generative process that, in turn, leads to worse adaptation results, especially for the most challenging semantic categories. For instance, the *pole*, *traffic light* and *traffic sign* classes appear to be recognized more effectively when no semantic consistency is enforced. That said, the semantic loss is actually beneficial for the majority of the semantic classes.
Finally, we notice that a stronger emphasis on the semantic loss ($\lambda_{sem} = 0.5$), while producing more visually appealing image translations in terms of color preservation and artifact removal (figures 5.14 and 5.15 from section 5.3.2), actually results in

slightly worse adaptation performance with respect to a scenario with a smaller loss weight ($\lambda_{sem} = 0.1$). As just discussed, this is probably due to an inherently flawed segmentation module and to the inaccuracy that it transfers to the generative framework. Figure 5.16 contains some examples of semantic predictions on real images from the Cityscapes validation set. We consider 3 different scenarios: no adaptation with just source supervision (third column), pixel-level adaptation using the original CycleGAN framework (forth column) and pixel-level adaptation with the additional semantic loss ($\lambda_{sem} = 0.1$) (fifth column). We can visually perceive a definite improvement of prediction capabilities when adaptation is performed. For example, the road is identified much more precisely, but also pedestrians and cars are detected with enhanced accuracy. On the other hand, changes introduced by the semantic loss with respect to the standard CycleGAN are less evident. We select some results that better highlight those differences. In particular, it is possible to see some prediction errors in terms of vegetation and wall detection that are corrected when the semantic consistency is enforced (rows one, four and five).



(a) Original image **(b)** Ground truth **(c)** No adaptation **(d)** $\lambda_{sem} = 0$ **(e)** $\lambda_{sem} = 0.1$

Figure 5.16: Segmentation maps of real images from the Cityscapes validation set when only synthetic supervision is provided (no adaptation) and when pixel-level adaptation is performed by the CycleGAN framework, with and without semantic loss.

5.3.4 Feature Framework

In this section we describe the performance of our unsupervised domain adaptation technique when an additional feature alignment is enforced. As discussed in chapter 4, a pair of domain feature-level discriminators are included inside the CycleGAN to improve the original pixel-level adaptation.
We explore two different optimization strategies. To start, we train the new framework exatly how it was done for the pixel-level adaptation, that is with two separate stages in which first the generative and then the segmentator modules are optimized (without counting the initial synthetic-based optimization). Next, we merge the two steps together in such a way that both the updated CycleGAN and the semantic predictor are simultaneously learnt. We set $\lambda_{sem} = 0.1$ in all our experiments.
In the second scenario we train the entire model for a total of 80000 steps. Moreover, the semantic segmentator is kept fixed for the first 20000 steps, that is until the generators start to perform acceptable translations. Finally, the starting learning rate for the segmentation network is set to 10^{-5}.

In table 5.3 we report the segmentation results in terms of per-class and mean IoU evaluated on the Cityscapes validation set. We compare the best performance achieved when only pixel-level adaptation is performed ($\lambda_{sem} = 0.1$), with the case in which the supplementary feature alignment in both a single step or separate stages is included. We set the λ_{feat} parameter, which controls the influnce exerted by the adversarial feature loss inside the full objective, to 10^{-3} and 10^{-4} respectively for the separate and single stage settings. Higher values for λ_{feat} enhance the focus to feature adaptation, but, in turn, seem to hurt the generative process, leading to worse adaptation results. In addition, we notice that an extra regularization action provided by the identity loss is beneficial when training simultaneously the segmentator and the generative modules (we set $\lambda_{iden} = 2$).

We observe that the additional feature framework is actually helpful for the domain adaptation task. The improvement with respect to the pixel-level adaptation is not substantial (less than 2% in the best scenario), but it is distributed quite uniformly on the majority of the classes, showing that it is consistent. Moreover, optimizing the entire framework in a single step seems to be the best choice, with the mean IoU rising

from 36.3% to 37.2%. The performance increase is probably due to a more effective feature-level adversarial learning when the semantic segmentator can contribute to the optimization of the adversarial loss by modifying the image representations it extracts and provides to the feature discriminators. On the other hand, error gradients back-propagating through both a discriminator and a segmentation network are likely to lose their strength before reaching the generator, preventing, in practice, an efficient parameter update to optimize the adversarial objective at feature-level. For this reason, feature adaptation loses effectiveness when the generator is competing alone without the support of a trainable segmentation module. In addition, we notice a more stable convergence in case of a training process concurrently performed for all the model components.

Method	Road	Sidewalk	Building	Wall	Fence	Pole	Traffic Light	Traffic Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorbike	Bicycle	**MeanIoU**
Synthetic	71.6	20.1	72.7	15.3	18.6	10.0	5.7	4.2	73.5	11.9	42.5	37.7	0.4	71.4	3.4	25.5	0.0	2.3	0.1	25.6
Pixel-level adaptation	87.1	39.9	79.0	21.3	14.0	22.8	16.0	16.0	80.7	25.4	68.0	52.2	14.3	77.0	19.3	29.8	3.9	8.0	0.6	35.5
Feature alignment (separate stages)	86.3	35.9	80.0	20.0	17.8	20.3	16.6	23.5	81.9	26.3	66.9	53.7	15.0	81.2	20.5	20.4	16.2	7.4	0.5	36.3
Feature alignment (single stage)	84.6	32.9	80.5	21.7	11.8	29.9	18.4	23.7	83.3	29.5	70.6	55.9	15.5	83.8	22.8	23.6	5.2	13.2	0.3	37.2
Real	96.6	76.3	87.7	43.1	49.2	42.5	48.9	60.5	88.2	55.9	90.2	70.3	47.0	87.5	51.9	71.5	60.1	44.8	66.6	65.2

Table 5.3: Results in terms of IoU on the Cityscapes validation set for different adaptation settings, namely when pixel-level adaptation is performed with and without feature alignment in both single and separate optimization stages.

In figures 5.17 and 5.18 we show some examples of GTA5 and Cityscapes images that have been translated from the original domain to the other. In particular, we refer to the three framework configurations for which we have reported the segmentation results in table 5.3, namely the pixel-level adaptation with semantic loss weight $\lambda_{sem} = 0.1$ and the combined feature and pixel level adaptation for both optimization strategies as described above.

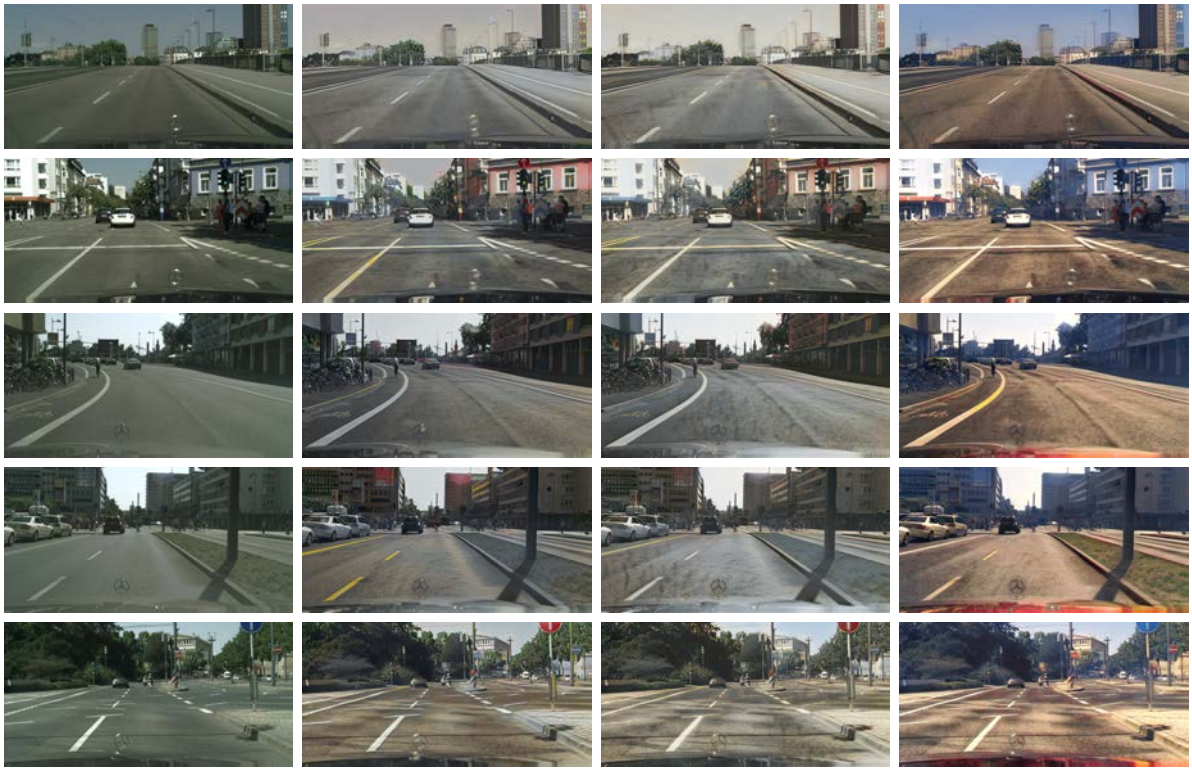
We notice that the feature framework doesn't help to perform more realistic translations. On the contrary, in certain cases it introduces some visual artifacts that normally don't show up when the standard CycleGAN model is employed. This because the feature adversarial loss promotes feature alignment, which entails the generation of images that appear as originating from the other domain under the scope of the semantic feature extractor, rather than samples that possess visual attributes from a

different dataset. For example, GTA5 adapted images tend to show more frequently the generative artifacts that typically affects the sky region (last two rows of figure 5.17).

(a) Original **(b)** Pixel-level **(c)** Separate stages **(d)** Single stage

Figure 5.17: Examples of GTA5 images adapted to the Cityscapes dataset with different generative settings, namely when pixel-level adaptation is performed with and without feature alignment in both single and separate optimization stages.

Furthermore, as already discussed in section 5.3.2, the cross-domain mapping mainly corresponds to a texture change, which is particularly evident in the road becoming smoother in the GTA5-to-Cityscapes adaptation or more irregular when the projection follows the inverse path. Moreover, colors are modified in terms of saturation level and are adjusted from more intense and colorful to less vivid and bright (or viceversa) depending on the direction of the translation.
It is also possible to notice the effect of the identity loss introduced in the single stage optimization scenario, which encourages a better color preservation (last columns of

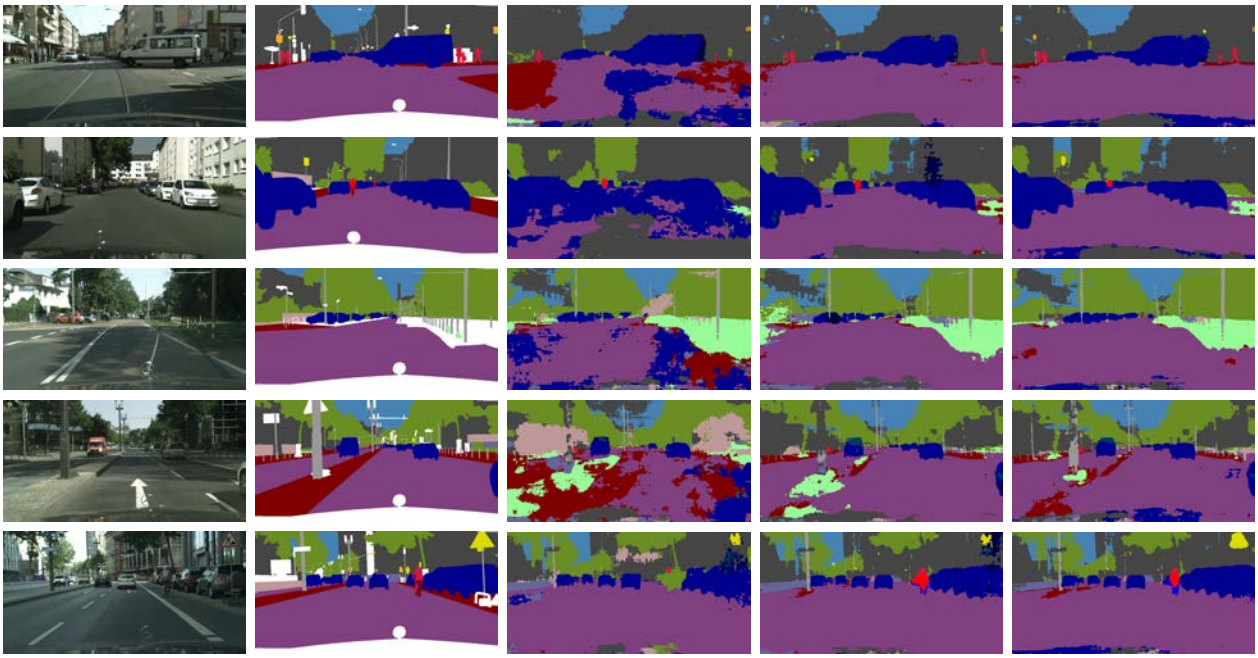


(a) Original **(b)** Pixel-level **(c)** Separate stages **(d)** Single stage

Figure 5.18: Examples of Cityscapes images adapted to the GTA5 dataset with different generative settings, namely when pixel-level adaptation is performed with and without feature alignment in both single and separate optimization stages.

figures 5.17 and 5.18). In particular, it prevents the red-blue color inversion that affects some objects (e.g. cars, buildings and traffic signs) in the other two cases.
Figure 5.19 contains a few examples of prediction maps computed on images from the Cityscapes validation set. We put in comparison the source-only optimization and the pixel-level adaptation with and without the supplementary feature framework to highlight some improvements progressively achieved by the subsequent steps. As previously stated, the accuracy boost brought by the adaptation framework is clearly visible for every image. On the other hand, the improvement introduced by feature alignment is more difficult to be perceived, since it involves the majority of the semantic classes but by a small extent each. Nonetheless, in figure 5.19 we report a few cases which show the better prediction accuracy over some details. For instance, we can see that car edges are generally detected with a slightly more precision. Traffic signs are also recognized with more confidence (last row). Moreover, some more evi-

dent prediction errors that affect semantic maps in the pixel-level adaptation scenario are corrected when feature alignment is performed (second and forth rows).



(a) Original **(b)** Ground truth **(c)** No adaptation **(d)** Pixel-level adaptation **(e)** Final model

Figure 5.19: Segmentation maps of real images from the Cityscapes validation set when only synthetic supervision is provided (no adaptation), when pixel-level adaptation is performed with semantic loss ($\lambda_{sem} = 0.1$) and when the unified domain adaptation framework is applied.

In table 5.4 we compare our final unsupervised domain adaptation model, comprising both pixel and feature level adaptation and trained in a single optimization stage, similarly to CyCADA [4]. In particular, we report the mean IoU computed on the Cityscapes validation set for three different training settings, namely when only synthetic supervision is provided, when pixel and feature level adaptation is performed and when target supervision from annotated real data is directly available.
In [4] the CyCADA model is tested on two different segmentation networks, namely the FCN8s-VGG16 [13] and DRN-26 [18]. Both of those networks have far more parameters than the MobileNetV2 [1] embedded in the DeepLabV3+ [21] that we employ in our framework (134M [13] and 21M [18] compared to 2M [1]), and for this reason they are not suitable to be inserted inside the image-to-image translation module. Furthermore the core architectures are quite different, going from deconvolution layers [13] and dilated convolution [18] to the inverted residual blocks with

Method	Synthetic	Pixel+Feat	Real
Ours (MobileNetV2[1])	25.6	37.2	65.2
CyCADA[4] (FCN8s[13])	17.9	35.4	60.3
CyCADA[4] (DRN-26[18])	21.7	39.5	67.4

Table 5.4: Results in terms of mean IoU on the Cityscapes validation set for different adaptation techniques, i.e. our final model and the CyCADA framework with different semantic segmentation networks. We report the results for multiple training settings, namely when only synthetic supervision is provided, when pixel and feature level adaptation is performed and when target supervision from annotated real data is directly available.

linear bottlenecks [1]. Hence, it is not possible to perform a direct comparison of the domain adaptation performance obtained by the different adaptation frameworks. Nonetheless, we observe that our model with 37.2% of mean IoU places itself right in the middle with respect to the CyCADA final results (35.4% and 39.5%), showing an improved accuracy compared to the FCN8s case. Moreover, the adaptation performance are coherent with the target supervision upper bound: the distance in terms of mean IoU between a source-adapted and a target based supervised training for our model (from 37.2% to 65.2%) is basically identical to the one regarding the DRN-26 adaptation (from 39.5% to 67.4%), revealing similar adaptation performance achieved by our framework and the CyCADA model.
As concerns the source-only optimization lower bound, we notice that the authors of [4] report quite low results compared to ours (table 5.4). Unfortunately, they don't go into details on how they obtain those results.

6
Conclusions

In this thesis we developed a technique based on adversarial learning to address domain adaptation in the context of semantic segmentation. We place ourself in the challenging, but also more practical, scenario of unsupervised adaptation, where the segmentation network has not direct access to annotated data from the target domain on which it has to perform its final predictions.
To address the unsupervised domain adaptation task we resort to a generative strategy. We reduce the domain shift between the source and target datasets by means of an image-to-image translation network based on the GAN model. The synthesized target-like source images are then used to train the semantic segmentator and enforce a form of target supervision. Additional semantic consistency to the pixel-level adaptation process is effectively achieved by providing the segmentation network with a supervisory role inside the generative module. On this regard, we rely on a light-weight implementation of the semantic segmentator in order to fulfil memory constraints.
To prove the effectiveness of our model, we validate it on a synthetic to real adaptation scenario, in which both source and target datasets contain high resolution images of urban scenes. Results show a consistent improvement with respect to the no adaptation case.

In addition, we augment the generative module with an adversarial feature framework, with the intent to assist pixel-level adaptation with a supplementary alignment of source and target image features extracted by the semantic predictor. Moreover, we unify the optimization phase in a single step in which both the image-to-image translation module and the semantic segmentator network are concurrently trained. Further experimental results show that the additional feature level adaptation affects positively the final semantic prediction accuracy, proving to be an useful addition to the original pixel-level strategy.

As regards future works, we could modify the image-to-image translation framework in order to introduce different types of constraints to the generative adversarial module, by resorting, for example, to some recently proposed techniques that allow one-sided mappings. This should effectively reduce complexity and memory occupation of the generative framework, while leaving room for the adaptation of more computationally demanding segmentation networks achieving state-of-the-art performance in the semantic segmentation field.
Furthermore, additional investigations can be directed toward the improvement of the adaptation at feature-level. For instance, we could analyze feature statics to reveal which activations exhibit the highest sensitivity to domain discrepancy between source and target inputs and focus our attention on them. Moreover, we could act on the output probability map produced by the segmentator rather than on intermediate representations in order to reach a more efficient adaptation performed directly inside the prediction space.

References

[1] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 4510–4520.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14, 2014, pp. 2672–2680.

[3] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017, pp. 2242–2251.

[4] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell, "CyCADA: Cycle-consistent adversarial domain adaptation," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 10–15 Jul 2018, pp. 1989–1998.

[5] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2014.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, 1998, pp. 2278–2324.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," vol. abs/1512.03385, 2015.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 2015, pp. 3431–3440.

[14] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *CoRR*, vol. abs/1506.04579, 2015.

[15] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 6230–6239.

[16] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[17] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.

[18] F. Yu, V. Koltun, and T. A. Funkhouser, "Dilated residual networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 636–644.

[19] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[21] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, 2018, pp. 833–851.

[22] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *CoRR*, vol. abs/1412.3474, 2014.

[23] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 97–105.

[24] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 136–144.

[25] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 2058–2065.

[26] B. Sun and K. Saenko, "Deep CORAL: correlation alignment for deep domain adaptation," in *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, 2016, pp. 443–450.

[27] B. J. Frey, G. E. Hinton, and P. Dayan, "Does the wake-sleep algorithm produce good density estimators?" in *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, 1995, pp. 661–667.

[28] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[29] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[30] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2015, pp. 262–270.

[31] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 2414–2423.

[32] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision (ECCV)*, 2016.

[33] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6997–7005.

[34] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," in *European Conference on Computer Vision (ECCV)*, 2018.

[35] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 5967–5976.

[36] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[37] M. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 469–477.

[38] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 700–708.

[39] Z. Yi, H. R. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017, pp. 2868–2876.

[40] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017, pp. 1857–1865.

[41] S. Benaim and L. Wolf, "One-sided unsupervised domain mapping," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 752–762.

[42] H. Fu, M. Gong, C. Wang, K. Batmanghelich, K. Zhang, and D. Tao, "Geometry-consistent generative adversarial networks for one-sided unsupervised domain mapping," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[43] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[44] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 95–104.

[45] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 2242–2251.

[46] Y. Ganin and V. S. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 1180–1189.

[47] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky, "Domain-adversarial training of neural networks," *CoRR*, vol. abs/1505.07818, 2015.

[48] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 2962–2971.

[49] Z. Ren and Y. J. Lee, "Cross-domain self-supervised multi-task feature learning using synthetic imagery," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 762–771.

[50] J. Hoffman, D. Wang, F. Yu, and T. Darrell, "Fcns in the wild: Pixel-level adversarial and constraint-based adaptation," *CoRR*, vol. abs/1612.02649, 2016.

[51] Y. Zhang, P. David, and B. Gong, "Curriculum domain adaptation for semantic segmentation of urban scenes," in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 2017, pp. 2039–2049.

[52] A. Dundar, M. Liu, T. Wang, J. Zedlewski, and J. Kautz, "Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation," *CoRR*, vol. abs/1807.09384, 2018.

[53] Z. Wu, X. Wang, J. E. Gonzalez, T. Goldstein, and L. S. Davis, "ACE: adapting to changing environments for semantic segmentation," *CoRR*, vol. abs/1904.06268, 2019.

[54] S. Sankaranarayanan, Y. Balaji, A. Jain, S. Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 3752–3761.

[55] S. Sankaranarayanan, Y. Balaji, C. D. Castillo, and R. Chellappa, "Generate to adapt: Aligning domains using generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 8503–8512.

[56] Y. Tsai, W. Hung, S. Schulter, K. Sohn, M. Yang, and M. Chandraker, "Learning to adapt structured output space for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 7472–7481.

[57] Z. Murez, S. Kolouri, D. J. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 4500–4509.

[58] T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Perez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[59] Y.-C. Chen, Y.-Y. Lin, M.-H. Yang, and J.-B. Huang, "Crdoco: Pixel-level domain transfer with cross-domain consistency," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[60] W. Hung, Y. Tsai, Y. Liou, Y. Lin, and M. Yang, "Adversarial learning for semi-supervised semantic segmentation," in *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, 2018, p. 65.

[61] M. Biasetton, U. Michieli, G. Agresti, and P. Zanuttigh, "Unsupervised domain adaptation for semantic segmentation of urban scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[62] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2813–2821.

[63] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[64] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 102–118.

[65] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.