Adversarial Learning Strategies for Semantic Segmentation

Master Candidate:
Matteo Biasetton

Supervisors:
Pietro Zanuttigh
University of Padova
Umberto Michieli
University of Padova

Master's Degree in Computer Engineering

Academic Year 2018/2019

Padova, 15/04/2019

University of Padova

Department of Information Engineering

Master's Degree in Computer Engineering

Adversarial Learning Strategies for Semantic Segmentation

Master Candidate:

Matteo Biasetton

Supervisors:

Pietro Zanuttigh

University of Padova

Umberto Michieli

University of Padova

15/04/2019

ACADEMIC YEAR 2018/2019

UNIVERSITY OF PADOVA

Abstract

Computer Engineering
Department of Information Engineering

Adversarial Learning Strategies for Semantic Segmentation

by Matteo BIASETTON

Semantic image segmentation is a computer vision task in which we label specific regions of an image according to their semantic content. This task is of essential importance for a wide range of applications like robotics, autonomous driving, medicine and image editing. Although many datasets have been built for this task, they are typically generic while a specific problem could require to focus more on the data related to it.

One of the biggest problems is represented by the difficulty of gathering large datasets. This is caused by the intrinsic complexity and cost of producing fine detailed ground truth for the interested data, as it consists in manually classifying each pixel of the images.

In this work we tried to mitigate this problem developing and testing new techniques to perform semi-supervised training and domain adaptation with unlabeled data. Our framework started from some works, presented in the literature, which exploit an adversarial learning framework in order to train a segmentation network using both supervised and unsupervised data. Finally, we developed some extensions that further improve the performances of the unsupervised training process.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Professor Pietro Zanuttigh for helping and advising me throughout the entire duration of the thesis.

I am particularly grateful for the assistance given by my advisor PhD student Umberto Michieli who supported me during the development and writing of this thesis.

I would also like to thank my friends, for the good times spent together, the laughs and all the adventures faced, hoping there will be others in the future.

Finally, I must express my very profound gratitude to my family for the continuous support and encouragement throughout my years of study. This accomplishment would not have been possible without them. Thank you.

Contents

1 Introduction

Semantic image segmentation is a computer vision task in which we label specific regions of an image according to their semantic content. More specifically, the goal of semantic image segmentation is to label each pixel of an image with the corresponding class of what is being represented. Since the prediction is done for every pixel in the image, this task is commonly referred to as dense prediction.
The task of semantic segmentation is of essential importance for a wide range of real world applications, for example: road segmentation for autonomous vehicles, scene segmentation for robot perception, medical image segmentation and image editing tools.

Numerous methods have been proposed to tackle this task and large datasets have been constructed with focus on different sets of scenes/objects to target various real world applications. However, this task remains challenging because of large object/scene appearance variations, occlusions, and lack of context understanding.

Although many datasets have been built for this task, they are typically generic while a specific problem could require to focus more on the data related to it. One of the biggest problems is represented by the difficulty of gathering large datasets. This is caused by the intrinsic complexity and cost of producing fine detailed ground truth for the interested data, as it consists in manually classifying each pixel of the images.

In this work we tried to mitigate this problem developing and testing new techniques to perform semi-supervised training and domain adaptation with unlabeled data. In particular, we investigated the use of datasets which are only partially annotated and, for the domain adaptation task, we considered a scenario where a large amount of annotated synthetic data is available but labeled real world samples are not available.
We started from the framework proposed by Hung et al. [1], which exploits an adversarial learning framework, where a segmentation network is trained using both labeled and unlabeled data thanks to the combination of three different losses. The first loss is a standard supervised cross-entropy loss exploiting

ground truth annotations which allows to perform an initial supervised training phase on labeled data. The second one is an adversarial loss derived from a fully convolutional discriminator, which takes in input the semantic segmentation from the generator network and the ground truth segmentation maps and produces a pixel-level confidence map distinguishing between the two types of data. The third one is based on a self-teaching framework, where the predicted segmentation is passed through the discriminator in order to obtain a confidence map and then high confidence regions are considered reliable and used as ground truth for self-teaching the network over them. Finally, we developed some extensions that further improve the performances of the unsupervised training process.

The remainder of this thesis is organized as follows. Chapter 2 introduces the problem of semantic segmentation and describes the state-of-the-art frameworks that use deep learning techniques to solve this problem. Chapter 3 describes Generative Adversarial Networks and some advanced techniques that exploit unsupervised or weakly supervised data to improve the performance of image segmentation. Chapter 4 describes the framework utilized for this work and the proposed techniques that have been developed. Chapter 5 reports the main results obtained from the experiments. Chapter 6 discusses the current status of the project and also outlines possible directions for future work. Chapter 7 reports some additional visual results of the developed technique. Finally Chapter 8 describes some additional results on the domain adaptation task.

2 Semantic Segmentation

Image segmentation is a computer vision technique that consists in dividing or partitioning an image into parts that have similar features or properties. Semantic image segmentation is a more challenging extension of this task which aims to label each pixel of an image with a corresponding class of what is being represented. Since we are making a prediction for every pixel in the image, this task is commonly referred to as dense prediction.

The task of semantic segmentation is of essential importance for a wide range of real world applications. For example, an autonomous car needs to detect the roadsides with a high precision in order to move by itself. In robotics, production machines should be able to delineate the exact shape of an object to perform advanced automatic tasks. Further examples could also include medical image segmentation for automatic disease diagnosis and advanced image editing tools.

It is important to note that semantic segmentation does not separate instances of the same class but only focuses on the category of each pixel. In other words, two objects of the same category in the input image will not be distinguished as separate objects. There exists a different class of models, known as instance segmentation models, which do distinguish between separate objects of the same class. Figure 2.1 reports an illustration of this difference.

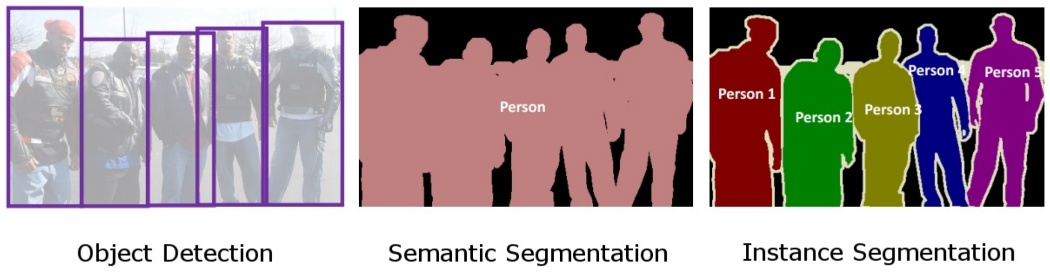

FIGURE 2.1: Illustration of different computer vision task related to image understanding

The goal of semantic segmentation, in the simplest formulation, is to take either a RGB color image ($height \times width \times 3$) or a grayscale image ($height \times width \times 1$) and output a segmentation map where each pixel contains a class label represented as an integer ($height \times width \times 1$). More advanced techniques add additional channels to the input image to include 3D information of the environment (depth maps).

Currently, the most performing techniques for semantic segmentation are based on Deep Convolutional Neural Networks. In the following sections these techniques will be introduced as well as some state-of-the-art network for semantic segmentation.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) [2] are a specialized kind of neural network for processing data that has a known grid-like topology. Some examples include time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be considered as a 2-D grid of pixels.

The name "convolutional neural network" indicates that the network employs a mathematical operation called convolution that is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers [3].

CNNs are composed of multiple building blocks, some common examples incude: convolutional layers, pooling layers, and fully connected layers, that are designed to automatically learn spatial hierarchies of features through a backpropagation algorithm [4].

Convolutional Layer

Convolution is a specialized type of linear operation which in this particular case is used for feature extraction, where a kernel (composed by a 2D array of numbers), is applied across the input (denoted as input tensor). An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map. This procedure is repeated applying multiple kernels, with different sizes, to form an arbitrary number of feature maps, which represents different characteristics of the input tensors. Figure 2.2 shows an illustration of the standard convolution process for 2D data.

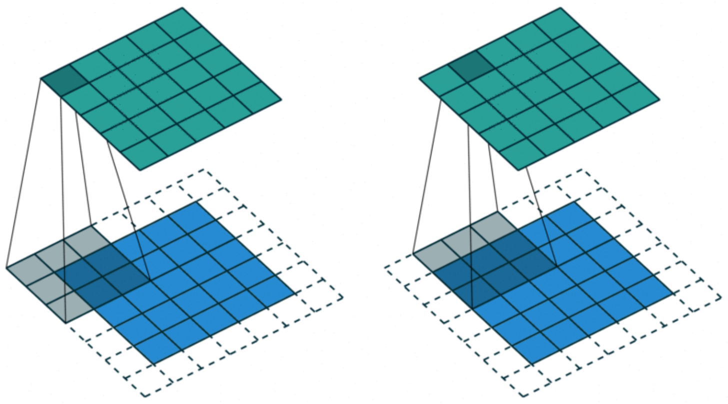

FIGURE 2.2: Example of standard convolution with a kernel size of 3×3.

There exists some variations of the standard convolution described above. For example, we may want to skip over some positions of the kernel to reduce the computational cost. This operation is called strided convolution and it can be seen as a downsampling of the output of the full convolution function. It is also possible to define a separate stride for each direction of motion of the kernel.
Another variation of the standard convolution operation is the dilated convolution, also called "Atrous Convolution" [5], that consists in inserting "holes" in the kernel matrix to capture features of the input tensor at a different scale. Compared to the increase in the kernel size, this operation does not require additional computational costs. Figure 2.3 shows an illustation of this technique.

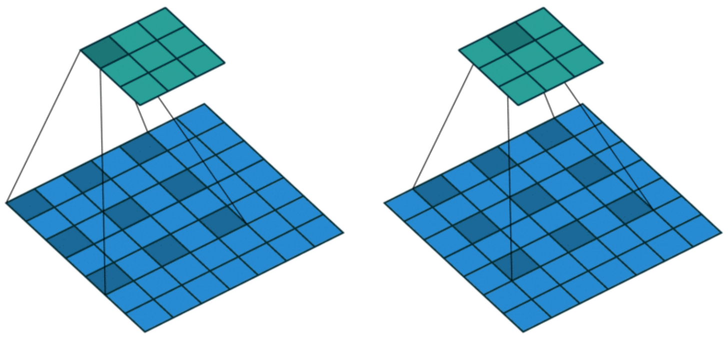

FIGURE 2.3: Example of atrous convolution with a kernel size of 3×3 and dilation factor equal to 1.

Finally, the outputs of a linear operation such as convolution are then passed through a nonlinear activation function. There are three functions that are commonly used for this task:

- Sigmoid function: $f(x) = \dfrac{1}{1+e^{-x}}$

- Hyperbolic tangent: $f(x) = \dfrac{2}{1 + e^{-2x}} - 1$
- Rectified Linear Unit (ReLU): $f(x) = \max(0, x)$

Pooling Layer

The pooling layer is often used between convolutional layers in a CNN architecture. This layer provides a typical downsampling operation which reduces the in-plane dimensionality of the feature maps in order to introduce a translation invariance to small shifts and distortions, and to decrease the number of subsequent learnable parameters. Typically this layer is used to perform two operations: average pooling and maximum pooling. Maximum pooling extracts patches from the input feature maps, outputs the maximum value in each patch, and discards all the other values. Average pooling instead outputs the average value in each patch. Figure 2.4 shows an illustation of max pooling operation.

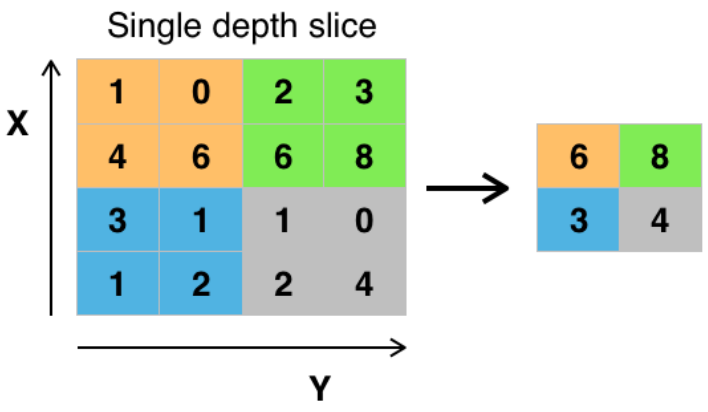

FIGURE 2.4: Example of max pooling operation with a filter size of 2×2.

Fully Connected Layer

In the classification problems the output feature maps of the final convolution or pooling layer is typically flattened (i.e. transformed into a one-dimensional vector), and connected to one or more fully connected layers (known as dense layers), in which every input is connected to every output by a learnable weight. In a classification task, the last of these layers typically has the same number of output nodes as the number of considered classes. A common activation function applied to the multi-class classification task is the softmax function which normalizes output real values from the last fully connected layer to target class probabilities, where each value ranges between 0 and 1 and all values sum to 1.

The softmax function is defined as:

$$f(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \tag{2.1}$$

Where i is the i-th class and K is the number of classes considered for the classification problem.
This layer was initially used also to perform semantic segmentation, however this had the drawback of limiting the input images to a fixed size. For this reason, starting from the idea introduced by the Fully Connected Network (FCN) [6], the dense layers have been substituted by convolutional ones which solve the problem of multiple resolution images and also reduce the computational costs needed to produce the output segmentation maps.

Network Training

Training a network is a process which consists in finding kernels in convolutional layers and weights in fully connected layers which minimize the differences between output predictions and given ground truth labels on a training dataset. The backpropagation algorithm is the method commonly used for training neural networks where loss function and gradient descent optimization algorithm play essential roles. A loss function, also referred to as a cost function, measures the compatibility between output predictions of the network and given ground truth labels.

Gradient descent is commonly used as an optimization algorithm that iteratively updates the learnable parameters (i.e. kernels and weights) of the network to minimize the loss. The gradient of the loss function provides us the direction in which the function has the steepest rate of increase and each learnable parameter is updated in the negative direction of the gradient with an arbitrary step size based on a hyperparameter called learning rate. The gradient is, mathematically, a partial derivative of the loss with respect to each learnable parameter, and a single update of a parameter is formulated as follows:

$$w = w - \eta \cdot \frac{\partial L}{\partial w} \tag{2.2}$$

Where w stands for each learnable parameter, η stands for a learning rate, and L stands for a loss function.

In practice, the learning rate is one of the most important hyperparameters to be set before the training starts. As a consequence of memory limitations, the gradients of the loss function with regard to the parameters are computed by using a subset of the training dataset called mini-batch, and applied to the parameter updates. This method is called mini-batch gradient descent, also frequently referred to as stochastic gradient descent (SGD), and the mini-batch size is also a hyperparameter. In addition, many improvements on the gradient descent algorithm have been proposed and widely used, such as SGD with momentum, and Adam [7].

2.2 CNNs for Image Classification

The ImageNet challenge has been traditionally tackled with image analysis algorithms such as SIFT with mitigated results until the late 90s. However, a gap in performance has been brought by using neural networks.

The first deep learning model published by A. Krizhevsky et al. [8] won the 2012 ImageNet competition with a test accuracy of 84.6% outperforming the previous best one with an accuracy of 73.8%. This famous model, the so-called "AlexNet" is what can be considered today as a simple architecture with five consecutive convolutional filters, max-pool layers and three fully-connected layers.

Simonyan at al. [8] (2014) released the VGG16 model, composed of sixteen convolutional layers, multiple max-pool layers and three final fully-connected layers. In particular, they chained multiple convolutional layers with ReLU activation functions creating non-linear transformations. Indeed, introducing non-linearities allows models to learn more complex patterns. Moreover they introduced 3x3 filters for each convolution (as opposed to 11x11 filters for the AlexNet model) and noticed they could recognize the same patterns than larger filters while decreasing the number of parameters to train. This model won the 2013 ImageNet competition with 92.7% accuracy.

Szegedy et al. (2014) [9] proposed GoogLeNet (as known as Inception V1), a deeper network with 22 layers using such "inception modules" for a total of over 50 convolutional layers. Each module is composed of 1x1, 3x3, 5x5 convolutional layers and a 3x3 max-pool layer in order to increase sparsity in the model and obtain different type of patterns. The feature maps produced are then concatenated and analyzed by the next inception module. The GoogLeNet model won the 2014 ImageNet competition with accuracy of 93.3%.

Microsoft ResNet [10] brought back the idea of going deeper. This model won the 2016 ImageNet competition with 96.4% accuracy. It is well-known due to its depth (152 layers) and the introduction of residual blocks. The residual blocks address the problem of training a really deep architecture by introducing identity skip connections between the output of one or multiple convolutional layers and their original input. Consequently, patterns from the input image can be learned in deeper layers. Moreover, this method does not add any additional parameter and does not increase the computational complexity of the model. The residual block is shown in figure 2.5.

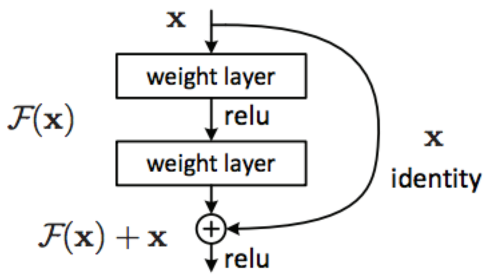

FIGURE 2.5: Basic building block of residual learning framework

2.3 CNNs for Semantic Segmentation

A naive approach towards constructing a neural network architecture for semantic segmentation simply consists in stacking a number of convolutional layers (with same padding to preserve dimensions) and output a final segmentation map. This directly learns a mapping from the input image to its corresponding segmentation through the successive transformation of feature mappings. However this approach is quite computationally expensive and is not used in practice.
It is important to note that for deep convolutional networks, earlier layers tend to learn low-level concepts while later layers develop more high-level (and specialized) feature mappings. A common technique to maintain expressiveness consists in increasing the number of feature maps (channels)as we get deeper in the network.
Currently, the most successful techniques for semantic segmentation are based on the same macro structure that is called *Autoencoder*. In general, these types of models are composed by two main components: the first one, which is called Encoder, is a state-of-the-art CNN for classification without its final fully connected layers. The second one, which is called Decoder, is the component that upsamples the feature maps produced by the Encoder to the final pixel-wise prediction. Decoders are the components that most determine the difference

between those models as they differ in the approaches utilized to upsample the resolution of the feature map.

The first network that adopted such structure was the Fully Convolutional Network (FCN) by Long et al. [6]. They transformed the existing and well-known classification models into fully convolutional ones by replacing the fully connected layers with convolutional ones to output spatial maps instead of classification scores. Those maps are upsampled using fractionally strided convolutions to produce dense per-pixel labeled outputs. This work is considered a milestone since it showed how CNNs can be trained end-to-end for this problem, efficiently learning how to make dense predictions for semantic segmentation with inputs of arbitrary sizes [11].

Badrinarayanan et al.[12] presented SegNet, a variant of FCN in which the decoder stage is composed by a set of upsampling and convolutional layers which are at last followed by a softmax classifier to predict pixel-wise labels. Each upsampling layer in the decoder corresponds to a max-pooling one in the encoder-part. This operation is performed using the max-pooling indices from the corresponding feature maps in the encoder phase. The upsampled maps are then convolved with a set of trainable filters to obtain dense features maps.

Ronneberger et al.[13] introduced the U-Net architecture which is an improvement of FCN. They modified the fully convolutional architecture by expanding the capacity of the decoder module. The architecture consists in a contracting path to capture context and a symmetric expanding path that enables precise localization.

Lin et al. [14] introduced the Feature Pyramid Network (FPN) an architecture that shows significant improvement as a generic feature extractor in several applications. In particular, they exploited the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with a marginal extra cost. A top-down architecture with lateral connections is developed for building high-level semantic feature maps at all scales.

Zhao et al.[15] proposed the Pyramid Scene Parsing Network (PSPNet) to better learn the global context representation of a scene. Patterns are extracted from the input image using a feature extractor like ResNet [10] with a dilated network strategy. Then the feature maps are fed to a Pyramid Pooling Module to distinguish patterns with different scales. Features are pooled with four different scales each one corresponding to a pyramid level and processed by a 1x1 convolutional layer to reduce their dimensions. With this technique each pyramid level analyses sub-regions of the image with different location. The outputs of the pyramid levels are upsampled and concatenated to the inital feature maps to contain the local and the global context information. Finally, they are processed by a convolutional layer to generate the pixel-wise predictions.

Chen et al. [16] presented the Deeplab v2, an autoencoder network based on ResNet network [10]. In particular, they removed the down-sampling operator from the last few max pooling layers of DCNNs and instead upsampled the filters in subsequent convolutional layers. Filter upsampling consists in inserting holes between nonzero filter taps. In [16] they used the term atrous convolution as a shorthand for convolution with upsampled filters. Moreover to handle objects at multiple scales, they employed multiple parallel atrous convolutional layers with different sampling rates and they called the proposed technique "Atrous Spatial Pyramid Pooling" (ASPP). Finally, they boosted the model's ability to capture fine details by employing a fully connected Conditional Random Field (CRF) [17].

3 Unsupervised Techniques for Semantic Segmentation

As mentioned in the initial problem statement, the goal of this work is to develop and test new techniques which improve the accuracy of semantic segmentation models in all the situations in which large datasets are not available or very costly to produce.
The way to achieve this is to find the best method to extract useful information from unlabeled data that can be used to reinforce the standard supervised training. We chose to utilize a framework based on GANs [18] that is very commonly used for this task.
The work for this thesis is based on the work of Hung et al. [1] that proposed a new technique to reinforce the adversarial training with unsupervised data. This and other techniques will be discussed in details in the following sections.

3.1 Generative Adversarial Networks

Goodfellow et al. [18] in 2015 proposed a new framework for estimating generative models via an adversarial network. This framework, called Generative Adversarial Networks (GANs), is composed by two networks pitted against each other: a generative model G (Generator) that captures the data distribution, and a discriminative model D (Discriminator) that estimates the probability that a sample came from the training data rather than from G. Figure 3.1 illustrates an example of a basic GAN structure.
As described in the paper [18], the network G takes samples z from a fixed distribution $P_z(z)$, and transforms them to approximate the distribution of training samples x. The adversarial network D is used to define a loss function which is used to explicitly evaluate the output produced by G. This framework corresponds to a min-max two-player game with the following value function:

$$V(G,D) : \min_G \max_D V(D;G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

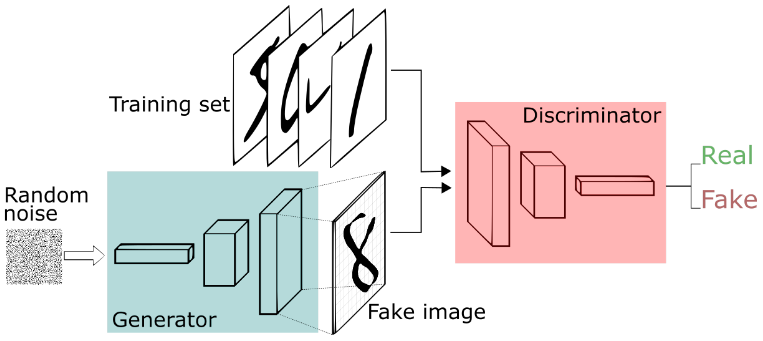

FIGURE 3.1: Example of a GAN Structure.

In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $1/2$ everywhere.
The adversarial model is trained to optimally discriminate samples from the empirical data distribution and samples from the deep generative model. The generative model is concurrently trained to minimize the accuracy of the adversarial, which provably drives the generative model to approximate the distribution of the training data. The adversarial network can be interpreted as a "variational" loss function, in the sense that the loss function of the generative model is defined by auxiliary parameters that are not part of the generative model.
As originally explained by the authors, GANs can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles [18].
The original GAN was developed using Multi-Layer Perceptrons, but later versions using deep convolutional GANs (DCGAN [19]) instead have shown impressive improvements in the task of generating realistic data.
As originally suggested by the authors, adversarial networks can be used to perform semi-supervised learning: features captured by the discriminator can be used to improve the performance of classifiers when limited labeled data is available. The work of this thesis is based on this idea applied to semantic image segmentation.

3.2 Adversarial Networks Applied to Semantic Segmentation

The adversarial networks, as defined in the section above, were used in the original paper to discriminate between the data produced by the generator and the sample from the real distribution. To apply this framework to the field of semantic segmentation we need to take in account the difference between the two problems.

As discussed in Chapter 2, semantic segmentation models aim to map an input image into the corresponding segmentation map. This can be done by training the network in a supervised manner using the labels provided with the input data.

Using the same terminology of the GANs framework we can identify the segmentation network as the generator that "generates" segmentation maps corresponding to the input images. The adversarial network (discriminator) in this case has the task of discriminate between the segmentation maps produced by the generator and the ground truth labels associated to the input images.

Luc et al. [20] in 2016 proposed this approach for semantic segmentation. They utilized a FCN as the generator network and a CNN as the discriminator network. In their setup the generator network is trained with a combination of two loss functions: the classical cross-entropy loss between input data and corresponding labels and the adversarial loss provided by the discriminator. As discussed by the authors of [20], the adversarial loss encourages the segmentation model to produce label maps that cannot be distinguished from ground-truth ones by an adversary binary classification model.

3.3 Semi-Supervised Semantic Segmentation Techniques

Semantic segmentation architectures are typically trained on huge datasets with pixel-wise annotations (e.g., the Cityscapes [21] or CamVid [22] datasets), which are highly expensive, time-consuming and error-prone to generate. To overcome this issue, semi-supervised methods are emerging, trying to exploit weakly annotated data (e.g., with only image labels or only bounding boxes) [23, 24, 25, 26, 27, 28, 29] or completely unlabeled [30, 31] data after a first stage of supervised training.

In 2015 Papandreou et al. [32] introduced a novel Expectation-Maximization (EM) methods for training DCNN semantic segmentation models from weakly annotated data. The proposed algorithms alternate between estimating the latent pixel labels (subject to the weak annotation constraints), and optimizing the DCNN parameters using stochastic gradient descent (SGD). Additionally they show how the EM approach also excels in the semi-supervised scenario. In particular, they show that having access to a small number of strongly (pixel-level) annotated images and a large number of weakly (bounding box or image-level) annotated images, the proposed algorithm can almost match the performance of the fully-supervised system.

In 2017 Souly et al. [33] proposed a weakly supervised semantic segmentation framework using GANs. In their work they extended GANs by replacing the traditional discriminator D with a fully convolutional multi-class classifier, which, instead of predicting whether a sample x belongs to the data distribution, assigned to each input image pixel a label y from the K semantic classes or mark it as fake sample assigning the class $K + 1$. To train this network they fed three inputs to the discriminator: labeled data, unlabeled data and fake data.

Hung et al. [1] developed a different framework for semi-supervised semantic segmentation. Differently from other competing approaches they did not utilize weakly annotated images to improve the accuracy, instead they proposed a framework based on GANs that uses unlabeled data to boost the standard training process. In contrast to the typical GANs discriminators, which take fixed sized input images and output a single probability value, they employed a fully-convolutional network that can take inputs of arbitrary sizes. After obtaining the initial segmentation prediction of the unlabeled image from the segmentation network, they computed a confidence map by passing the segmentation prediction through the discriminator network. This confidence map is then used as a supervisory signal for a self-taught scheme to train the segmentation network with a masked cross-entropy loss.

We based the work of this thesis on the work proposed in [1] since it has demonstrated good performances on some commonly used datasets for semantic segmentation.

3.4 Domain Adaptation Techniques

In addition to the aforementioned approaches to tackle the lack of data, an increasingly popular alternative is represented by domain adaptation from synthetic data. The development of sophisticated computer graphics techniques enabled the production of huge synthetic datasets for semantic segmentation purposes at a very low cost. To this end, several synthetic datasets have been built, e.g., GTA5 [34] and SYNTHIA [35] which have been employed in this work. The real challenge is then to address the cross-domain shift when a neural network trained on synthetic data needs to process real-world images since in this case training and test data are not drawn i.i.d. from the same underlying distribution as usually assumed [36, 37, 38, 39].

A possible solution is to process synthetic images in order to reduce the inherent discrepancy between source and target domain distributions mainly using generative networks (i.e., GANs) [40, 41, 42, 43, 44]

Unsupervised domain adaptation has been already widely investigated in classification tasks [45, 46, 47]. On the other hand, its application to semantic segmentation is still a quite new research field.

The first work to investigate cross-domain urban scene semantic segmentation is [48], where an adversarial training is employed to align the features from the different domains. In particular, they introduced a pixel-level adversarial loss to the intermediate layers of the network and imposed constraints to the network output.

In 2017 Zhang et al. [49] presented a curriculum-style learning approach to solve the problem of domain adaptation. In particular, they firstly learn to estimate the global label distributions of the images and local label distributions of the landmark superpixels of the target domain. Then they used these results to effectively regularize the training of the semantic segmentation network forcing its predictions to meet the inferred label distributions over the target domain.

Following these approaches, many works addressed the source to target domain shift problem with various techniques [50, 51, 52, 53, 53, 54].
As an example, Sankaranarayanan et al. [55] in 2017, proposed an approach based on GANs to reduce the domain shift between two domains. In particular, they proposed a joint adversarial approach that transfers the information of the target distribution to the learned embedding using a generator-discriminator pair.

Hoffman et al. [56] in 2018 presented a cycle-consistent adversarial domain adaptation method that unifies cycle-consistent adversarial models with adversarial adaptation methods. The proposed framework is able to adapt even in the absence of target labels and is broadly applicable at both the pixel-level and in feature space.

4 Proposed Methods

The work of this thesis is based on the work proposed in [1], which has been introduced in Chapter 3.
We utilized this framework as a baseline for our tests and we developed some extensions that further improve the performances of the unsupervised training process.

Firstly, we considered a scenario in which a limited amount of annotated data are available for a specific problem. We used this framework to train the network gathering information from both labeled and unlabeled images. This has a lot of advantages since unlabeled data can be gathered without any effort compared to the annotated ones, thus we can exploit huge unlabeled datasets to boost the segmentation network training.
As introduced in [18], this technique follows the idea of using the features learned by the discriminator network to improve the performances of the generator even with unlabeled data.

Differently from the previous problem, for domain adaptation we used this framework with data coming from two different datasets. In particular, we used a computer generated dataset to perform supervised training and a real world scenes dataset as our unsupervised input.
We investigated a scenario where a large amount of annotated synthetic data is available but there is no labeled real world data available.
Using this framework we tried to take advantage of unsupervised loss provided by the discriminator to reduce the domain shift between synthetic and real data.

4.1 Network Structure

In this section the overall system architecture will be introduced.
The general architecture of the proposed network is shown in Figure 4.1. This network is based on the framework proposed in [1]. The network is composed by two main blocks: the segmentation network and the discriminator network.

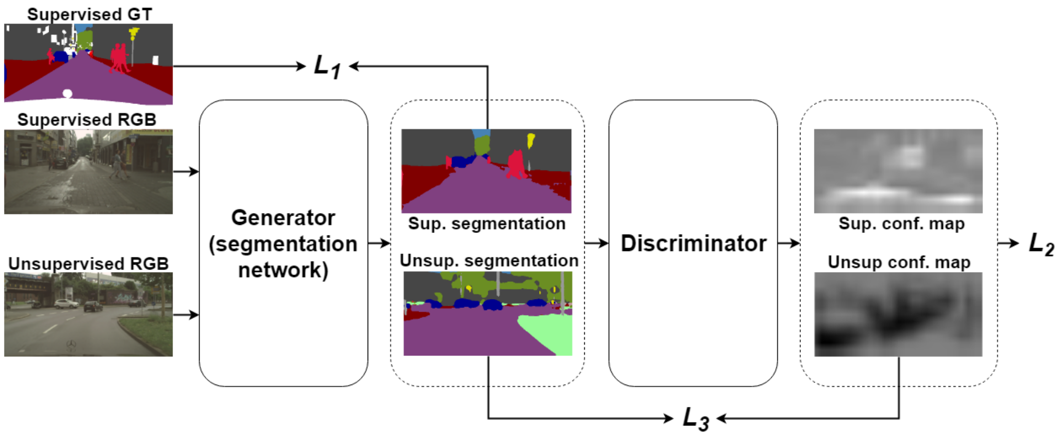

FIGURE 4.1: The architecture of the proposed framework for semi-supervised semantic segmentation. A first stage of supervised learning with annotated data is followed by a second stage of unlabeled data to boost the performance of the segmentation network through the combination of 3 losses. $\mathcal{L}_1$ is a standard cross-entropy loss between the generated synthetic segmentation maps and their respective ground truth. $\mathcal{L}_2$ is an adversarial loss based on the confidence map generated by a fully-convolutional discriminator network, which is trained with a spatial cross-entropy loss ($\mathcal{L}_D$). $\mathcal{L}_3$ is a novel loss for unlabeled data.

The segmentation network is a Deeplab v2 [16] which, as described in Chapter 2, is an autoencoder network based on ResNet [10] CNN. We considered to use Deeplab v2 since it has very good performances, however this approach does not rely on specific properties of this network and it can be substituted with any network for semantic segmentation. Furthermore in this work we have not employed the CRF since it is a post-processing technique that is used only for the output segmentation map.

The discriminator network in the problem of semantic segmentation aims at discriminating images produced by the segmentation network (fake images) from the corresponding ground truth images (real images).
The network used for the experiments is a fully convolutional network composed by 5 convolutional layers each followed by a leaky Rectified Linear Unit (ReLU) activation function.
Differently from the regular ReLU function, Leaky ReLU allows the pass of a

small gradient signal for negative values. As a result, it makes the gradients from the discriminator flow stronger into the generator.
Differently from other adversarial learning models, this network produces a per-pixel prediction instead of a single binary value for the whole input image.
The parameters of the discriminator network are reported in Figure 4.2.

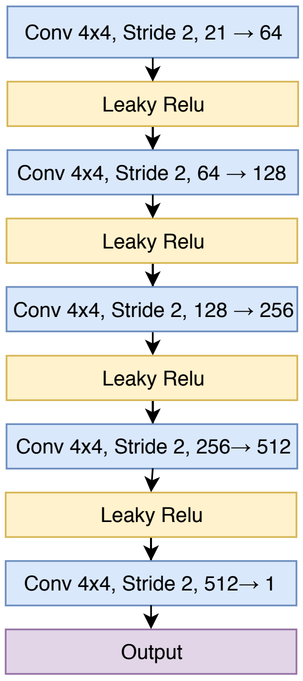
Conv 4x4, Stride 2, 21 → 64
Leaky Relu
Conv 4x4, Stride 2, 64 → 128
Leaky Relu
Conv 4x4, Stride 2, 128 → 256
Leaky Relu
Conv 4x4, Stride 2, 256→ 512
Leaky Relu
Conv 4x4, Stride 2, 512→ 1
Output

FIGURE 4.2: Discriminator network used for the experiments. For each block are reported: block type, kernel size, stride dimension, channels. The dimension of the output of the last block is referred to the training on the PASCAL VOC2012 dataset

4.2 Network Training Strategies

In this section we explain in details the techniques utilized to train the framework described above.
The main idea is to use a composed loss function to optimize the generator network with the standard back-propagation algorithms. This function is composed by different terms that exploit informations coming from both labeled and unlabeled data.

Given an input image $\mathbf{X}_n$ of size $H \times W \times 3$ and its one-hot encoded ground truth $\mathbf{Y}_n$, we denote the segmentation network by $G(\cdot)$ and the predicted probability map by $G(\mathbf{X}_n)$ of size $H \times W \times C$ where C is the classes number. We denote the fully convolutional discriminator by $D(\cdot)$ which takes a probability map of size $H \times W \times C$ and outputs a confidence map of size $H \times W \times 1$. Finally to distinguish between data coming from the supervised dataset and the unsupervised one we use the terms $\mathbf{X}_n^s$ and $\mathbf{X}_n^u$ respectively.

The loss of the discriminator $\mathcal{L}_D$ is a standard cross-entropy loss between the produced map and the one-hot encoding related to the fake domain (class 0) or ground truth domain (class 1) depending on the fact that the input has been respectively drawn from the generator or from ground truth. This loss term is defined by:

$$\mathcal{L}_D = -\sum_{h,w} \log(1 - D(G(\mathbf{X}_n^{s,u}))^{(h,w)}) + \log(D(\mathbf{Y}_n^s)^{(h,w)}) \tag{4.1}$$

Where $D(G(\mathbf{X}_n))^{(h,w)})$ is the confidence map of $\mathbf{X}_n$ at location (h,w), and $D(\mathbf{Y}_n^s)^{(h,w)}$ is the confidence map of $\mathbf{Y}_n^s$ at location (h,w) .
Notice that the discriminator has to label with 0 the segmentation maps produced by the generator using both annotated data from the supervised dataset (denoted with X_n^s) and unlabeled data from the unsupervised dataset (i.e., X_n^u).

The loss term indicated as $\mathcal{L}_1$ is the standard cross-entropy function utilized to train the network only with annotated data and is defined by:

$$\mathcal{L}_1 = -\sum_{h,w} \sum_{c \in C} (\mathbf{Y}_n^s)^{(h,w,c)} \log(G(\mathbf{X}_n^s)^{(h,w,c)}) \tag{4.2}$$

The loss term indicated as $\mathcal{L}_2$ is the adversarial loss driven by the discriminator network and is defined by:

$$\mathcal{L}_2^{s,u} = -\sum_{h,w} \log(D(G(\mathbf{X}_n^{s,u})^{(h,w)})) \tag{4.3}$$

This term force the training of the generator network in the direction of fooling the discriminator producing data that resembles the ground truth statistics. Moreover it can be applied even with unlabeled data since it requires only the output of the segmentation network to be evaluated.

Finally the loss term $\mathcal{L}_3$ introduced in [1] is defined by the authors as a self-taught learning framework. The main idea is that the trained discriminator can generate a confidence map $D(G(\mathbf{X}_n^u))$ which can be used to infer the regions sufficiently close to those from the ground truth distribution. The self-taught, one-hot encoded ground-truth $\hat{\mathbf{Y}}$ is an element-set with $\hat{\mathbf{Y}}_n^{(h,w,c^*)} = 1$ if $c^* = \text{argmax}_c(G(\mathbf{X}_n)^{(h,w,c)})$. The resulting semi-supervised loss is defined by:

$$\mathcal{L}_3 = -\sum_{h,w}\sum_{c\in C} I(D(G(\mathbf{X}_n^u))^{(h,w)} > T_{semi}) \cdot \hat{\mathbf{Y}}_n^{(h,w,c)} \log(G(\mathbf{X}_n^u)^{(h,w,c)}) \tag{4.4}$$

where $I(\cdot)$ is the indicator function and T_{semi} is the threshold to control the sensitivity of the self-taught process.
Since $\hat{\mathbf{Y}}_n$ and $I(\cdot)$ are used as constant during training, Equation (4.4) can be simply viewed as a masked spatial cross entropy loss.
To conclude, a weighted average of the three losses is used to train the generator exploiting the proposed adversarial learning framework, i.e.,:

$$\mathcal{L}_G = \mathcal{L}_1 + \lambda_{adv}^s \cdot \mathcal{L}_2^s + \lambda_{adv}^u \cdot \mathcal{L}_2^u + \lambda_{semi} \cdot \mathcal{L}_3 \tag{4.5}$$

where λ_{adv}^s, λ_{adv}^u and λ_{semi} are three parameters that controls the influence of each related loss.
The discriminator is fed both with ground truth labels and with the generator output computed on a mixed batch containing both labeled and unlabeled data and is trained aiming at minimizing $\mathcal{L}_D$. Concerning the generator, instead, during the first 5000 steps $\mathcal{L}_3$ is disabled (i.e. λ_{semi} is set to 0) thus allowing the discriminator to enhance its capabilities to produce higher quality confidence maps before using them.

Proposed loss variants

As we can observe in Figure 4.3 the semi-supervised framework described above produces a confidence map that has high confidence (represented in white in the third image of Figure 4.3) in the center of the biggest segmented areas and very low confidence (represented in black in the third image of Figure 4.3) in the area corresponding to a change of class (i.e. edges/boundaries) in the segmentation map.

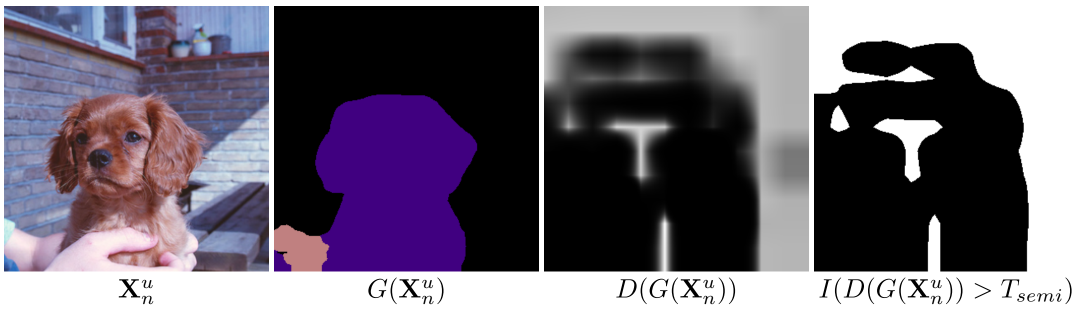

FIGURE 4.3: Overview of discriminator output during the training phase.

We designed a modified semi-supervised loss to tackle this problem with the goal of gathering more information from the generated confidence map. We indicate as $\widetilde{D}(\cdot)$ the normalized version of $D(\cdot)$ defined by the following linear function:

$$\widetilde{D}(G(\mathbf{X}_n))^{(h,w)} = \frac{D(G(\mathbf{X}_n))^{(h,w)} - D_{min}(G(\mathbf{X}_n))}{D_{max}(G(\mathbf{X}_n)) - D_{min}(G(\mathbf{X}_n))} \tag{4.6}$$

Where $D_{max}(\cdot)$ and $D_{min}(\cdot)$ indicate the maximum and the minimum values assumed by $D(\cdot)$ respectively.

Instead of selecting only the most confident regions of $D(\cdot)$, we used the full output of the discriminator as a weighting function for the cross-entropy loss evaluation. In particular we give large relevance to the regions that look like ground truth and then smaller and smaller up to no relevance to region marked as *fake* by the discriminator.

The designed semi-supervised loss indicated as $\mathcal{L}_{3,1}$ is defined by the following function:

$$\mathcal{L}_{3,1} = -\sum_{h,w}\sum_{c\in C} \widetilde{D}(G(\mathbf{X}_n^u))^{(h,w)} \cdot \hat{\mathbf{Y}}_n^{(h,w,c)} \log(G(\mathbf{X}_n^u)^{(h,w,c)}) \tag{4.7}$$

This loss can be seen as smoothed self cross-entropy considering that $\widetilde{D}(G(\mathbf{X}_n))$ acts as a weighting function for the term $\hat{\mathbf{Y}}_n$ described in Equation (4.4). Notice that, as in Equation (4.4), only the term $\mathbf{X}_n^u$ corresponding to unlabeled data is used for the evaluation of this loss.

Considering the task of domain adaptation, the unsupervised loss terms $\mathcal{L}_3$ and $\mathcal{L}_{3,1}$ forces the generator to adapt to the target domain, thus producing maps that resemble the ground truth ones as in the scenario of a single dataset.

As we can observe in Figure 4.4, unlabeled data would lead the model to produce a less noisy result in the areas corresponding to large classes in the input

images. However, at the same time, this loss contribution leads the model to mislead rare and tiny objects (such as *traffic lights*, *pole* or *person*).
In particular we can observe that the contribution of the designed loss $\mathcal{L}_{3,1}$ in this case produces worse visual results compared to Hung et al. ($\mathcal{L}_3$).

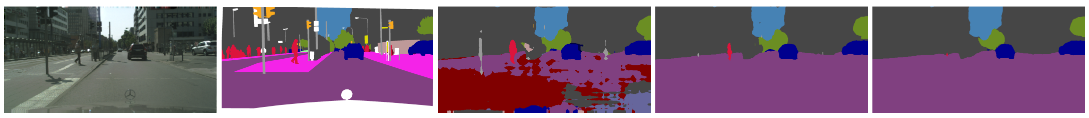

FIGURE 4.4: Overview of proposed technique applied to domain adaptation

To overcome this behavior we designed another variation of the $\mathcal{L}_3$ loss. The new loss term indicated by $\mathcal{L}_{3,2}$ is defined by:

$$\mathcal{L}_{3,2} = -\sum_{h,w}\sum_{c\in C} I(D(G(X_n^u))^{(h,w)} > T_{semi}) \cdot W_c^s \cdot \hat{\mathbf{Y}}_n^{(h,w,c)} \log(G(X_n^u)^{(h,w,c)})) \tag{4.8}$$

Where W_c^s is a weighting function computed on the source domain defined as:

$$W_c^s = 1 - \frac{\sum_n |p \in \mathbf{X}_n \wedge p \in c|}{\sum_n |p \in \mathbf{X}_n|} \tag{4.9}$$

Where we indicated as p a pixel of image X_n and $|\cdot|$ represents the cardinality of the considered set.
This corrective term serves as a balancing factor when unlabeled data of the target set are used. Notice that the term comes into play only when using unlabeled data of the target domain but the class frequencies have to be computed on the labeled data of the source domain since we need the ground truth labels to evaluate it. This calculation has only to be performed a priori and it is not changed as the learning progresses.
The results of the different modified losses compared to [1] are reported in Chapter 5.

5 Results

5.1 Experimental Setup

The original framework on which this thesis is based [1] was developed using Pytorch[1] version 0.2. We chose to re implement this framework using Tensorflow[2] version 1.12.0 because it is more supported than Pytorch and includes some tools like Tensorboard, which is a powerful suite that allows debug and visualization of the learning process.
To perform the trainings we utilized a single Nvidia 1080Ti GPU, which has 12GB of dedicated memory. The limited amount of available resources forced us to reduce the batch size and the images resolution until the networks fitted in memory. Moreover with this configuration the longest training we performed took about 20 hours to complete.
All the experiments were performed using the same parameters to train the network. We chose these parameters after some preliminary tuning of the proposed architecture. In particular we trained the generator network (G) using the standard technique proposed by [16] with Stochastic Gradient Descent (SGD) as optimizer with momentum set to 0.9 and weight decay to 10^{-4}. The discriminator network (D) has been trained using the Adam optimizer [7]. The learning rate employed for both G and D started from 10^{-4} and was decreased up to 10^{-6} by means of a polynomial decay with power 0.9.
We set the weighting parameters empirically to balance between the three components as: $\lambda^{s}_{adv} = 10^{-2}$ for annotated data, $\lambda^{u}_{adv} = 10^{-3}$ to give less weight in case of unlabeled data and $\lambda_{semi} = 10^{-1}$. Finally we set $T_{semi} = 0.2$ to obtain a significant mask from the confidence map.
For the generator network we used the standard Deeplab v2[3] without CRF [17] and based on the ResNet-101 model whose weights were pre-trained on the MSCOCO dataset [57][4].

[1]https://pytorch.org/
[2]https://www.tensorflow.org/
[3]We used the network provided by Wang and Ji available at
https://github.com/zhengyang-wang/Deeplab-v2--ResNet-101--Tensorflow/
[4]We used the weights computed by V. Nekrasov available at
https://github.com/DrSleep/tensorflow-deeplab-resnet

Tensorflow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

5.2 Datasets

In this section we introduce the datasets that we used to evaluate the performances of the semi-supervised framework and the proposed method for unsupervised domain adaptation.
To test the effectiveness of the proposed semi-supervised framework we used two publicly available datasets, namely PASCAL VOC2012 [58] and Cityscapes [21].
For the domain adaptation task we want to show that it is possible to train a semantic segmentation network in a supervised way on synthetic datasets and then apply unsupervised domain adaptation to real data in autonomous driving scenarios. Thus, we used two synthetic datasets, namely GTA5 [34] and SYNTHIA [35] for the supervised part of the training, while the unsupervised adaptation and the result evaluation have been performed on the real world Cityscapes [21] dataset.
PASCAL VOC2012 [58] is composed by $10,582$ color images with different resolutions, representing a large number of visual object in realistic scenes. They have a pixel level semantic annotation with 21 classes. Since the labels for the original test set are not available, we rearranged the original training and validation sets for our experiments. Accordingly to what has been done in [1], we used the original validation set, composed by 1449 annotated images, as our validation and test dataset.
Before feeding the images to the network we performed data augmentation applying a random scale between 0.5 and 1.5 and then a random crop of size 321×321 to have images of the same dimension.

CITYSCAPES [21] is composed by $2,975$ high resolution color images captured on the streets of 50 different European cities. They have a pixel level semantic annotation with 34 classes of which only 19 are taken in consideration

FIGURE 5.1: Examples of images of the PASCAL VOC2012 dataset

for training and testing. Since the labels for the original test set are not available, we rearranged the original training and validation sets for our experiments. We randomly divided the original training split in a training set, composed by $2,475$ images, and a validation set, composed by 500 images. The original high resolution images have been resized to 375×750 pixel for memory constraints. The testing was instead carried out on the original resolution of 2048×1024 pixel.

FIGURE 5.2: Examples of images of the Cityscapes dataset

GTA5 [34] is a huge dataset composed by 24966 photo-realistic synthetic images with pixel level semantic annotation. The images have been recorded from the prospective of a car in the streets of virtual cities (resembling the ones in California) in the open-world video game *Grand Theft Auto 5*. Being taken from a high budget commercial production they have an impressive visual quality and are very realistic. In our experiments, we used 23966 images for the supervised training and 1000 images for validation purposes. There are 19 semantic classes which are compatible with the ones of the Cityscapes dataset. The original resolution of the images is 1914×1052 pixel but we rescaled and cropped them to the size of 375×750 pixel for memory constraints before being fed to the architecture.

SYNTHIA [35] is a very large dataset of photo-realistic images. It has been produced with an ad-hoc rendering engine, allowing to obtain a large variability of the images. On the other hand, the visual quality is not the same of the commercial video game GTA5. We used the *SYNTHIA-RAND-CITYSCAPES*

FIGURE 5.3: Examples of images of GTA5 dataset

version of the dataset, which contains 9400 images with annotations compatible with 16 of the 19 classes of Cityscapes. These images have been captured on the streets of a virtual European-style town in different environments under various light and weather conditions. As done in previous approaches, we randomly extracted 100 images for validation purposes from the original training set, while the remaining part, composed by 9300 images, is used for the supervised training of our networks. Again, the images have been rescaled and cropped from the original size of 760×1280 pixel to 375×750 pixel. For the evaluation of the proposed unsupervised domain adaptation on the Cityscapes dataset, only the 16 classes contained in both datasets are taken into consideration.

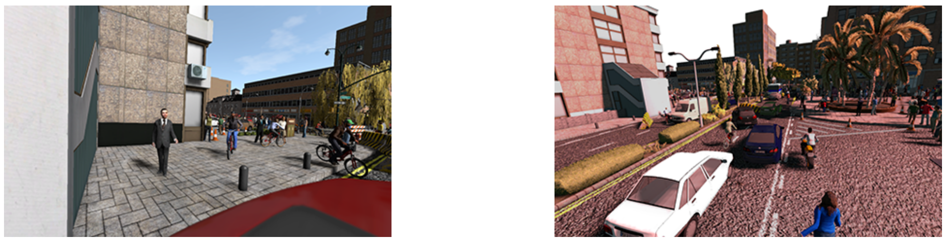

FIGURE 5.4: Examples of images of SYNTHIA dataset

5.3 Semi-Supervised Semantic Segmentation

In this section we present the results obtained from each of the techniques described in Chapter 4.

The first goal of this work was to demonstrate the effectiveness of the chosen semi-supervised framework. To test this we utilized two real-world datasets: PASCAL VOC2012 and Cityscapes. For each dataset we followed the same training strategy: we utilized half of the training set with annotations to compute the supervised loss terms and the remaining data as unsupervised input to perform the unsupervised learning.

To test the effectiveness of the proposed approach, for each considered dataset, we performed 4 different tests in which we use different combinations of loss term to optimize the network:

- supervised baseline: $\mathcal{L}_1$ (indicated as ***Baseline***)
- supervised adversarial: $\mathcal{L}_1 + \lambda^s_{adv} \cdot \mathcal{L}^s_2$ (indicated as ***Adversarial***)
- semi-supervised framework: $\mathcal{L}_1 + \lambda^s_{adv} \cdot \mathcal{L}^s_2 + \lambda^u_{adv} \cdot \mathcal{L}^u_2 + \mathcal{L}_3$ (indicated as ***Hung et al. [1]***)
- proposed method: $\mathcal{L}_1 + \lambda^s_{adv} \cdot \mathcal{L}^s_2 + \lambda^u_{adv} \cdot \mathcal{L}^u_2 + \mathcal{L}_{3,1}$ (indicated as ***Ours***)

Results on the PASCAL VOC2012 Dataset

With this dataset we trained the network for 40000 steps with a batch composed by 3 images, which was the maximum number allowed by the memory constraints.
Table 5.1 shows two evaluation metrics: the mean Intersection over Union (IoU) and mean pixel accuracy evaluated on the different tested techniques. Table 5.2 shows the mean intersection over union evaluated for each single class of the dataset.
As we can see from Table 5.1, the original semi-supervised framework presented by [1] improves mean IoU score by 0.5% and mean pixel accuracy score by 0.13%, compared to the fully supervised adversarial training technique. Furthermore the proposed variation of the semi-supervised framework brings a further improvement in both the evaluation metrics. In particular from Table 5.2 we can see how the proposed technique improves the results in some of the considered classes.

	Mean IoU	Mean Pixel Accuracy
Baseline	72.38	93.71
Adversarial	73.51	93.97
Hung et al. [1]	74.02	94.10
Ours ($\mathcal{L}_{3,1}$)	74.38	94.17

Table 5.1: Mean results of different techniques evaluated on the original PASCAL VOC2012 validation dataset.

	background	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dining table	dog	horse	motorbike	person	potted plant	sheep	sofa	train	tv/monitor
Baseline	93.2	**87.2**	37.4	84.9	64.6	75.5	92.2	84.4	88.6	32.8	75.5	53.6	**81.9**	77.9	78.2	**83.3**	52.4	77.8	43.6	83.9	71.1
Adversarial	93.7	86.3	**38.9**	85.2	66.9	78.7	**93.2**	**85.5**	88.0	**35.8**	75.9	58.8	80.3	77.5	**80.5**	83.0	54.1	80.9	44.2	83.8	72.8
Hung et al. [1]	93.7	86.6	38.5	**86.7**	**69.7**	76.5	93.1	85.2	**88.8**	33.2	80.1	60.0	81.8	80.6	78.8	83.1	55.3	81.5	44.5	**85.9**	71.0
Ours ($\mathcal{L}_{3,1}$)	**93.8**	86.6	38.8	86.3	67.7	**76.9**	92.6	83.9	88.7	35.0	**81.0**	**62.9**	81.7	**82.1**	79.6	83.2	**55.6**	**82.4**	**46.3**	83.2	**73.7**

TABLE 5.2: Mean intersection over union on the different classes of PASCAL VOC2012 evaluated on the original validation dataset.

Figures 5.5 and 5.6 reports the plot of the training error of G and D respectively during the training process. Figures 5.7 and 5.8 reports the plot of the mean IoU and mean pixel accuracy evaluated on the validation set. Finally Figure 5.9 reports the plot of the validation error on the PASCAL VOC2012 validation set.
We evaluated the metrics on the validation error every 1000 steps to reduce the time needed for the training.

As we can see in Figure 5.9, when we include the discriminator network in the training process the validation error tends to increase over time.
On the contrary the mean IoU metric and the pixel accuracy continue to improve. In a standard scenario the raise of the validation error suggests that the network is overfitting on the training data, however this is not the case since the other evaluation metrics, specific for semantic segmentation, are improving during the training process. This is a common behavior that has been observed when using adversarial networks: generator and discriminator are competing against each other, hence improvement on the one means a higher loss on the other. However the increment of the generator error (after that the discriminator starts to learn) does not affect the quality of the output images, which instead seems to be improved by the contribution brought by the discriminator. This increase in the network accuracy can be achieved only with an accurate tuning of the training parameters, i.e. learning rate or the various weighting terms. First of all it is essential to tune the learning rate of the generator and discriminator in a way that allows both to improve simultaneously. Moreover a too high value for the weighting terms $\lambda_{adv}^{s,u}$ and λ_{semi} can lead the generator error to diverge rapidly by the influence of the discriminator.

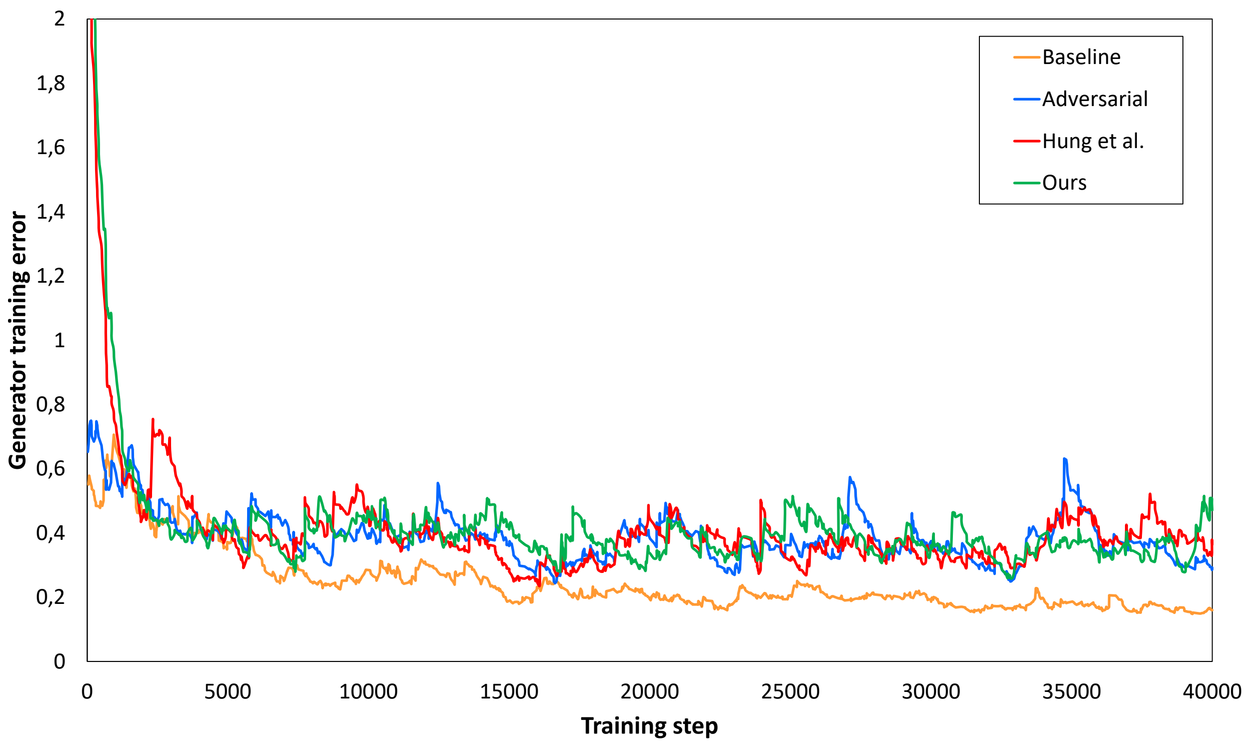

FIGURE 5.5: Generator training error on the PASCAL VOC2012 dataset. The error is computed on the batch fed to the network at the corresponding step.

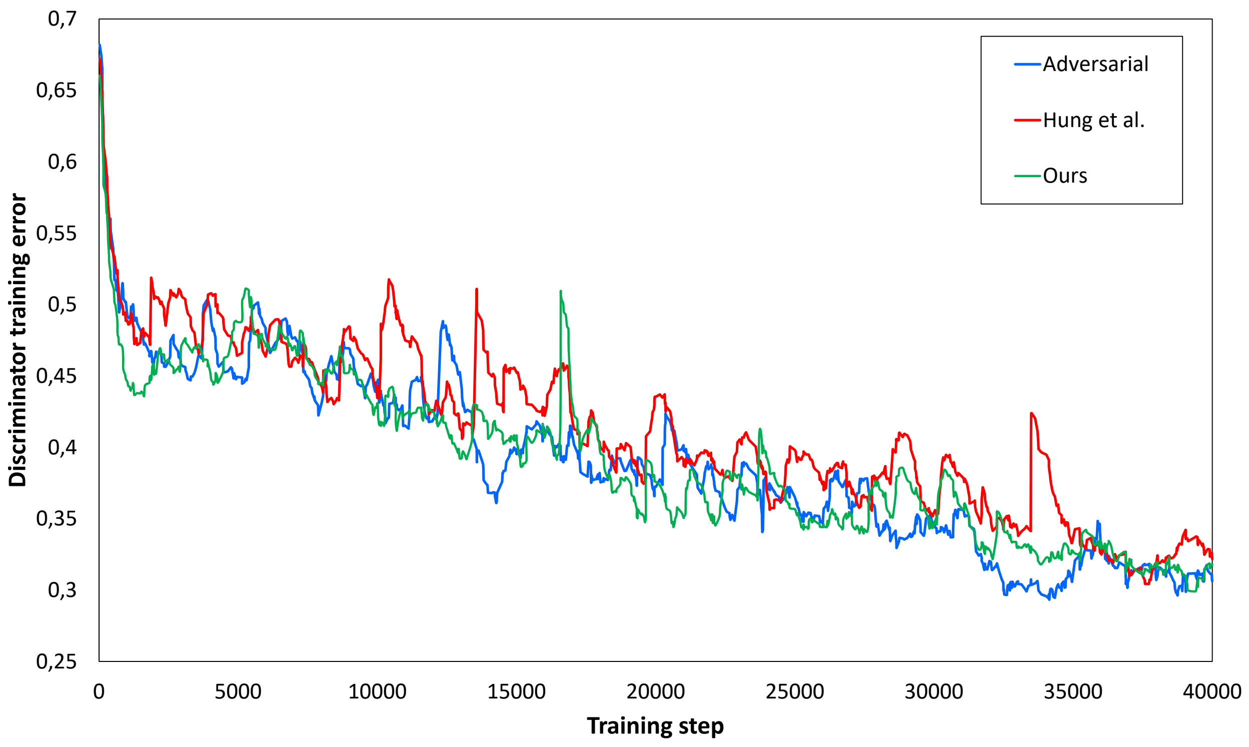

FIGURE 5.6: Discriminator error on the PASCAL VOC2012 training dataset. The error is computed on the batch fed to the network at the corresponding step.

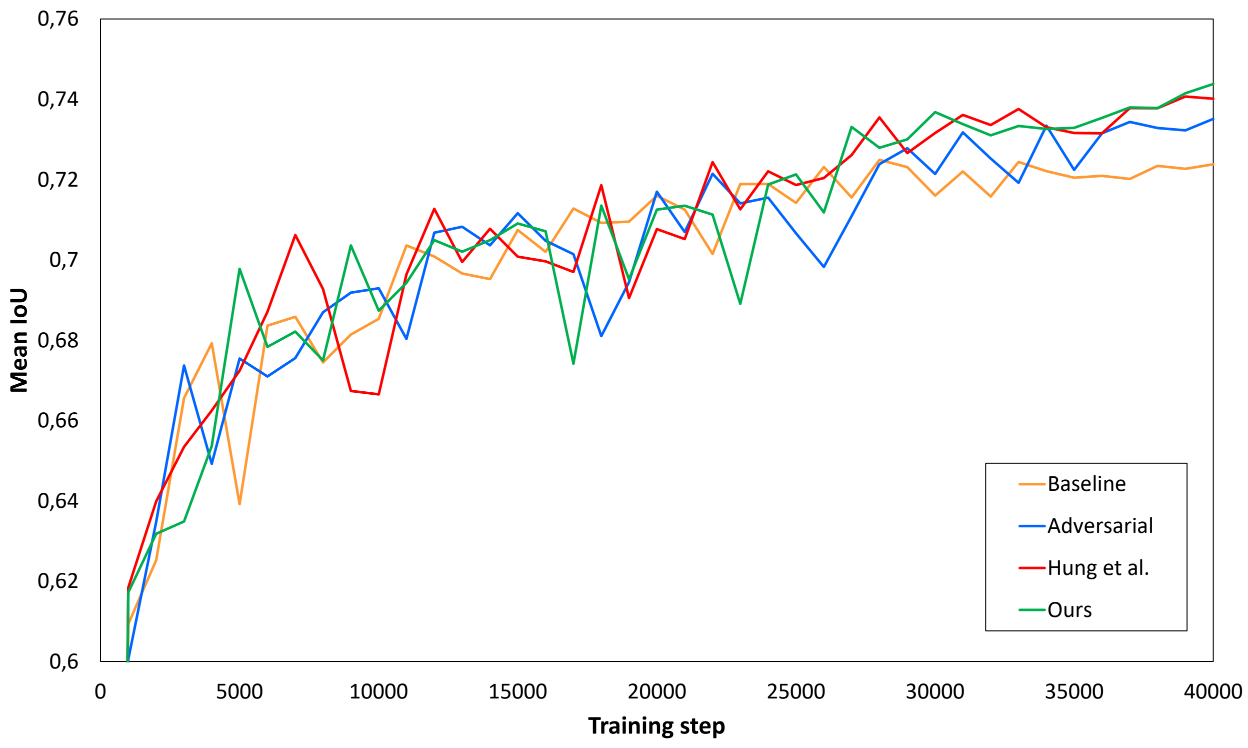

FIGURE 5.7: Mean IoU on the PASCAL VOC2012 validation dataset evaluated in the training phase.

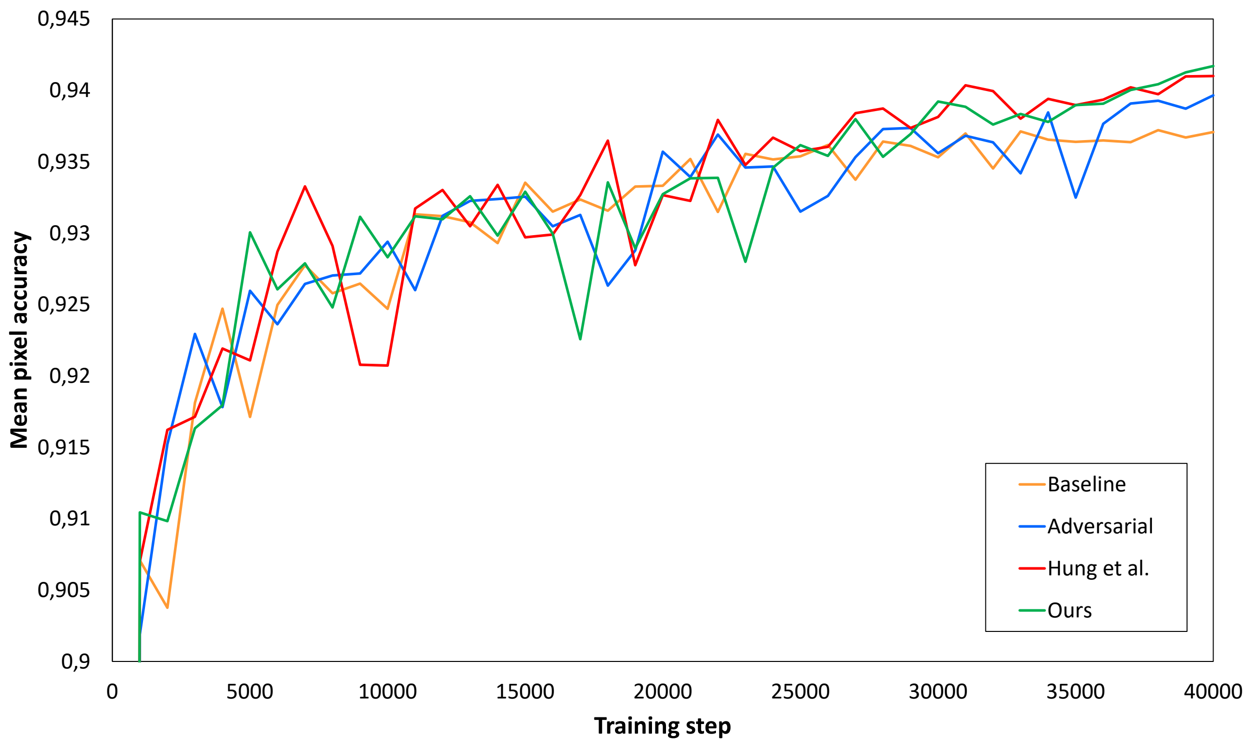

FIGURE 5.8: Mean Pixel Accuracy on the PASCAL VOC2012 validation dataset evaluated in the training phase.

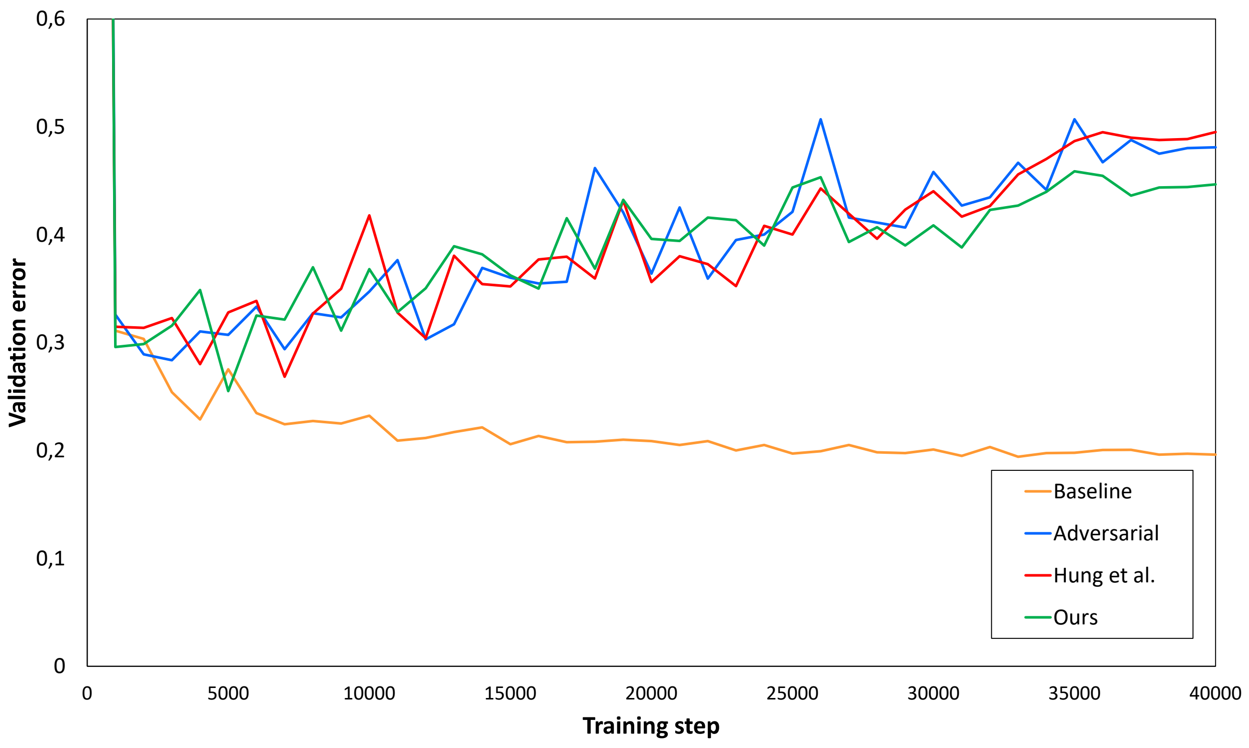

FIGURE 5.9: Validation error on the PASCAL VOC2012 validation dataset evaluated in the training phase.

In Figure 5.10 are reported some examples of output segmentation maps produced after the training of the network. As we can see from the images, the visual results confirm what emerges from the evaluation metrics: the network trained with the proposed method produces better segmentation maps compared to the other techniques. In particular we can observe how the proposed method enhances the boundaries of the segmentation map in correspondence of a class change and it helps to reduce the overall noise of the image.
As we can see in Figure 5.11 there are also some cases in which the proposed method has worse results compared to the other versions. If we focus on the examples in the last row, we can clearly see that the proposed method has some issues in predicting the *chair's* class. This particular case is a challenging one because it can have an ambiguous interpretation, indeed the object represented in the image is shaped like a chair but has some features that are commonly present on a sofa.
Additional output segmentation maps produced on the PASCAL VOC2012 validation set can be found in Appendix A.

Backgound	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus
Car	Cat	Chair	Cow	Dining-Table	Dog	Horse
Motorbike	Person	Potted_Plant	Sheep	Sofa	Train	TV/Monitor

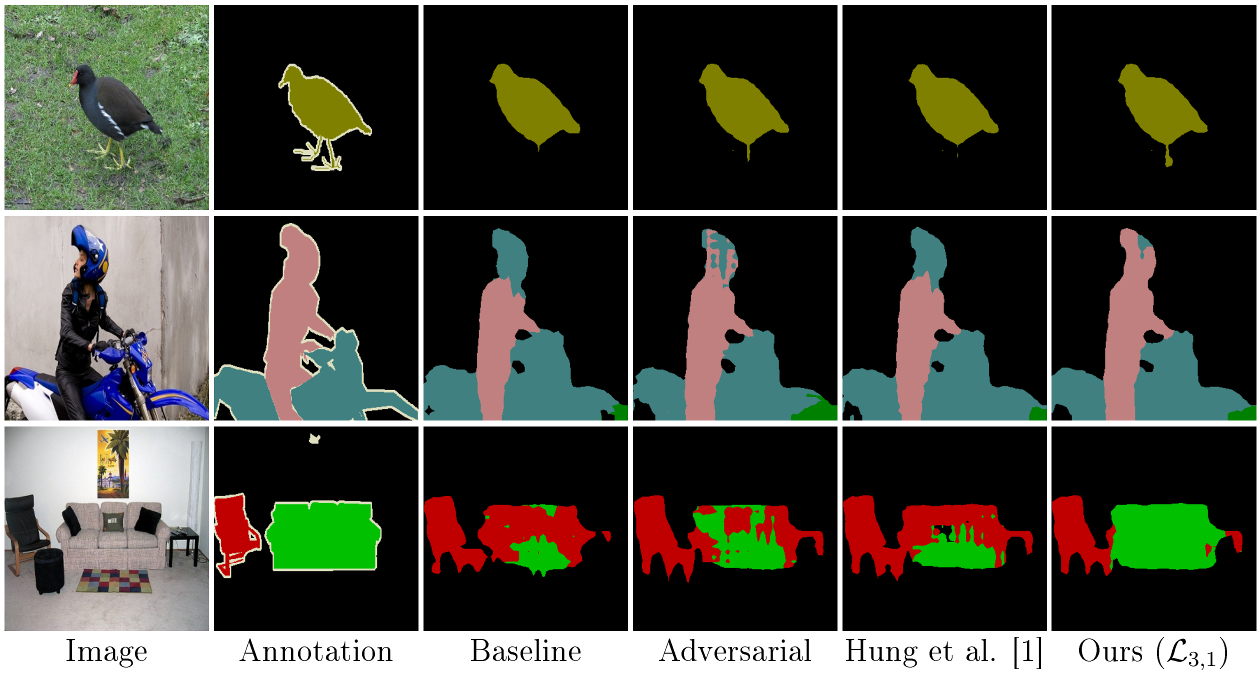

FIGURE 5.10: Examples of correct semantic segmentation of some sample scenes extracted from the PASCAL VOC2012 validation dataset. The network is trained using half of the dataset as annotated data and the remaining as unsupervised data

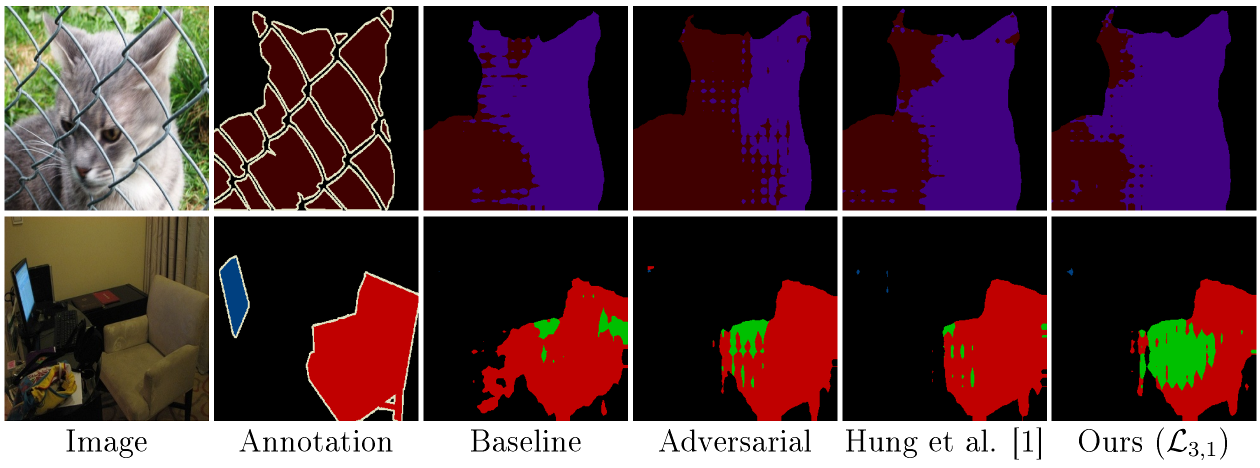

FIGURE 5.11: Examples of incorrect semantic segmentation of some sample scenes extracted from the PASCAL VOC2012 validation dataset. The network is trained using half of the dataset as annotated data and the remaining as unsupervised data

Results on the Cityscapes Dataset

With this dataset we trained the network for 40000 steps with a batch composed of only one image to fit the memory constraints.
For this dataset we performed the same tests that were performed for the PASCAL VOC2012 dataset to have a fair comparison.
As in the previous dataset we reported the two evaluation metrics and the mean IoU computed for each different class of the dataset.
As we can see from the results (Tables 5.3 and 5.4), even using this dataset the proposed method has higher scores in both the evaluation metrics. The semi-supervised technique of Hung et al. [1] instead has a lower mean IoU compared to the supervised adversarial training.
If we observe Table 5.2 we can see how the proposed technique improves the results in the majority of the considered classes.

These results are quite different from the ones reported in the original paper [1] in which the semi-supervised framework has an higher mean IoU than the supervised adversarial training. Moreover each obtained result is lower than the corresponding original one. This is probably caused by the lower resolution of images used to train the network.

	Mean IoU	Mean Pixel Accuracy
Baseline	49.81	87.37
Adversarial	50.91	86.38
Hung et al. [1]	50.80	86.69
Ours ($\mathcal{L}_{3,1}$)	53.71	87.79

TABLE 5.3: Mean results of different techniques evaluated on the original Cityscapes validation dataset.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrein	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
Baseline	**86,7**	61,3	73,4	24,1	27,6	41,1	15,1	48,7	87,3	39,0	60,6	66,3	14,6	86,6	33,8	53,3	29,1	36,3	61,3
Adversarial	83,5	63,8	70,4	22,2	25,4	**42,5**	23,6	53,5	**88,2**	46,1	62,1	65,8	14,8	86,8	34,4	51,7	31,5	38,5	62,5
Hung et al. [1]	84,5	61,8	71,2	23,6	23,4	41,4	23,3	52,2	**88,2**	43,5	59,9	66,4	14,8	87,0	34,0	56,9	31,7	38,0	63,3
Ours ($\mathcal{L}_{3,1}$)	85,9	**66,7**	**73,8**	**27,3**	**28,7**	41,6	**26,8**	**54,1**	87,2	**49,0**	**65,3**	**67,0**	**20,6**	**87,7**	**35,4**	**59,2**	**40,7**	**39,5**	**64,1**

TABLE 5.4: Mean intersection over union on the different classes of Cityscapes evaluated on the original validation dataset.

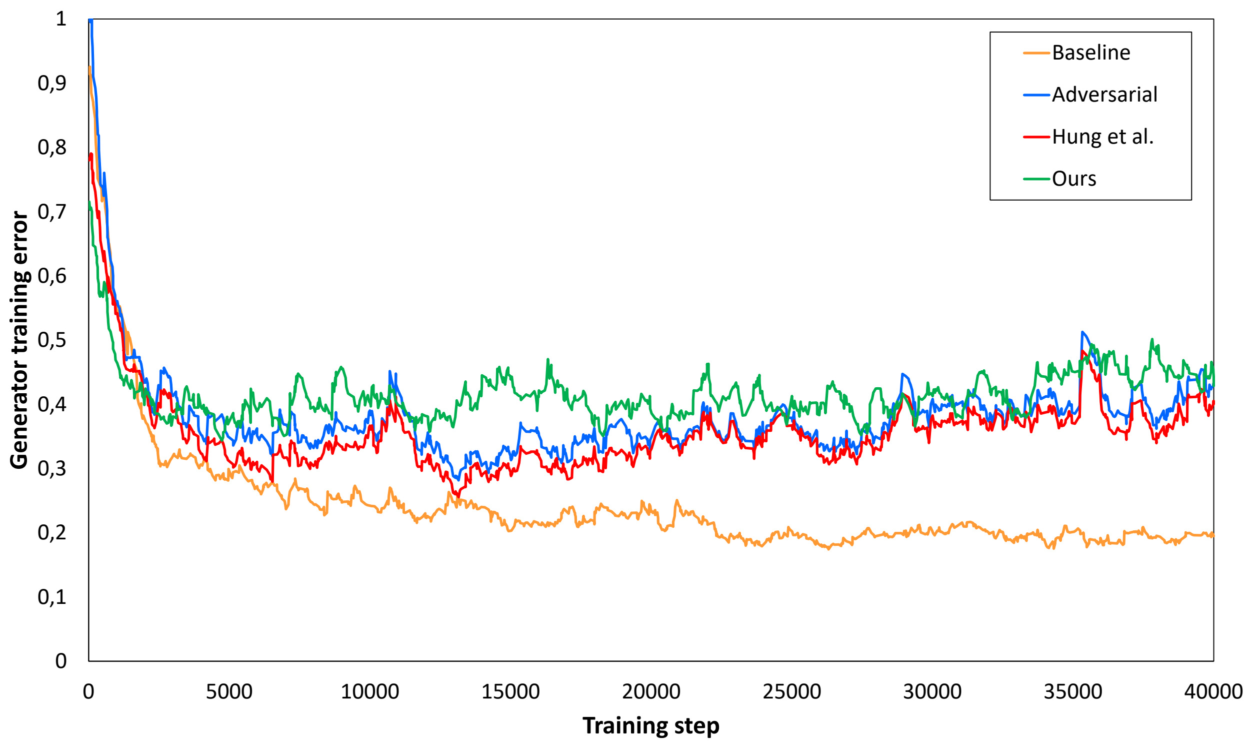

FIGURE 5.12: Generator training error on the Cityscapes dataset. The error is computed on the batch fed to the network at the corresponding step.

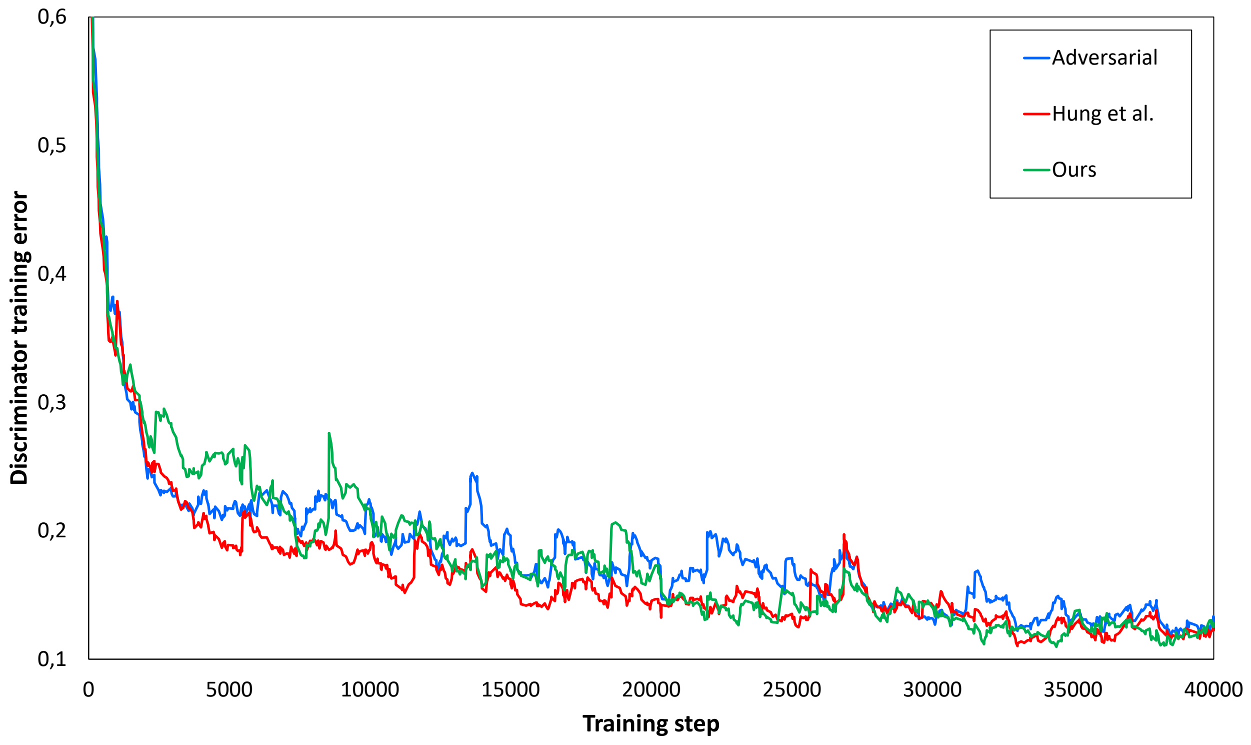

FIGURE 5.13: Generator training error on the Cityscapes dataset. The error is computed on the batch fed to the network at the corresponding step.

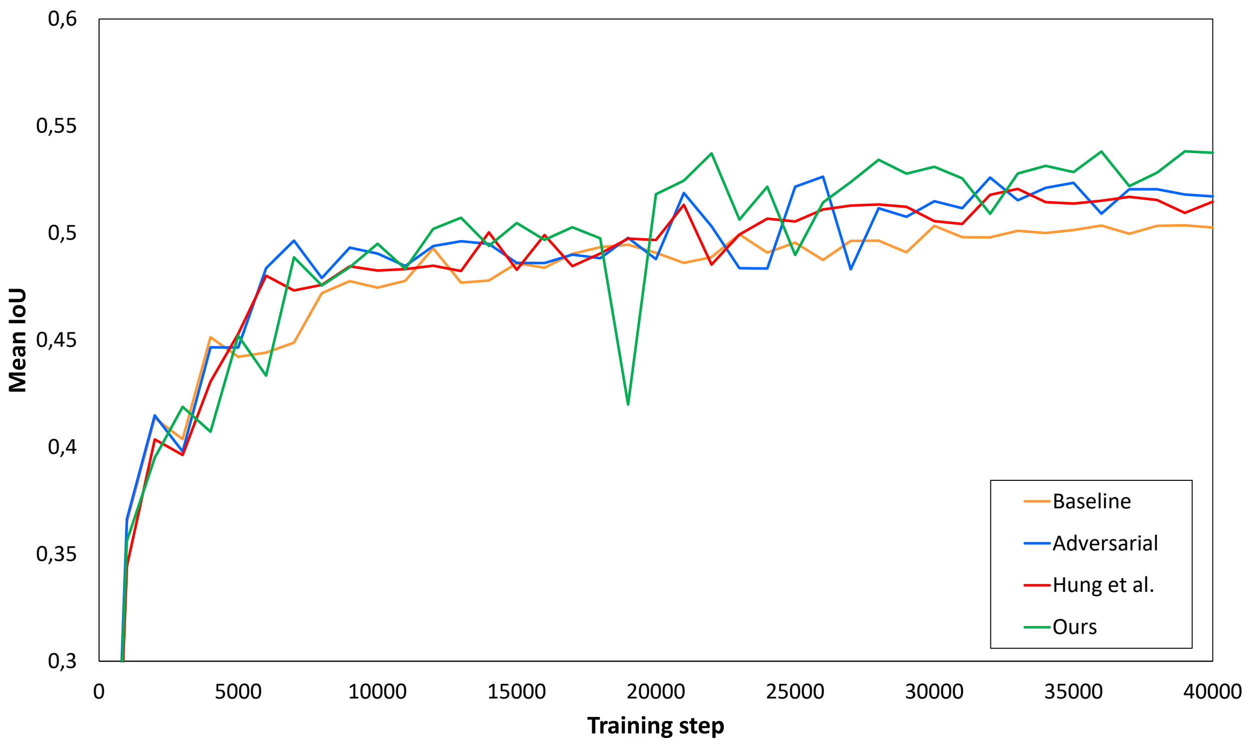

FIGURE 5.14: Mean IoU on the Cityscapes validation dataset evaluated in the training phase.

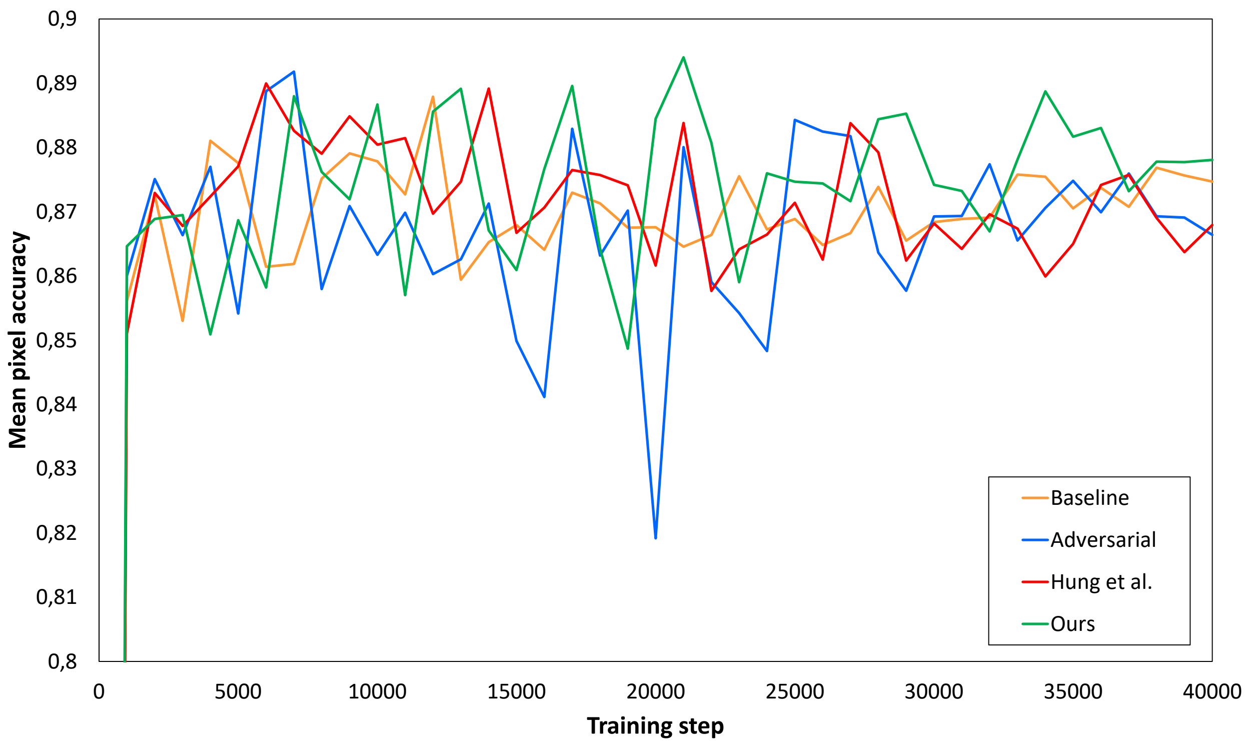

FIGURE 5.15: Mean Pixel Accuracy on the Cityscapes validation dataset evaluated in the training phase.

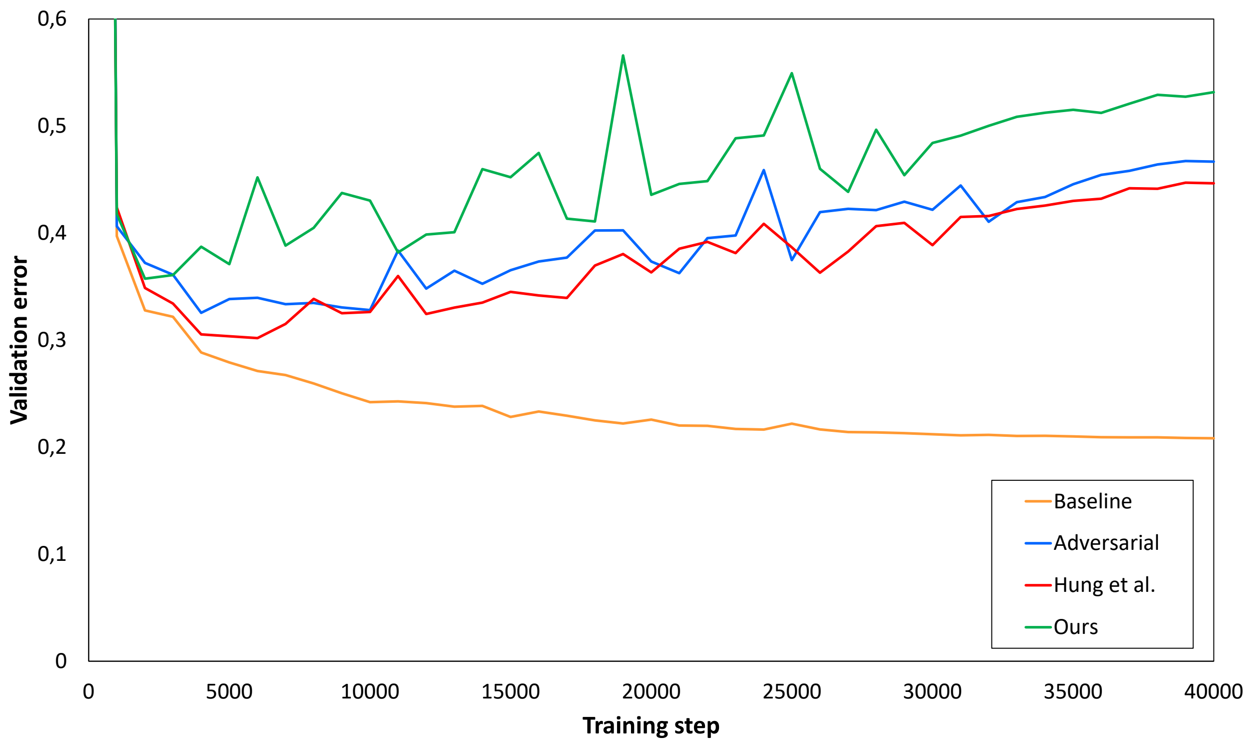

FIGURE 5.16: Validation error on the Cityscapes validation dataset evaluated in the training phase.

Figures 5.12 and 5.13 report the plot of the training error of G and D respectively during the learning process. Figures 5.14 and 5.15 reports the plot of the mean IoU and mean pixel accuracy evaluated on the validation set. Finally Figure 5.16 reports the plot of the validation error on the Cityscapes validation set.
We evaluated the metrics on the validation error every 1000 steps to reduce the time needed for the training.

As we can observe in the plot reported in Figure 5.15, the mean pixel accuracy during the training phase is very unstable compared to the results on the PASCAL dateset. This can be attributed to two factors: smaller batch size used for training the network and lower number of images in the validation dataset.
In Figure 5.16 we can observe that the validation error has a similar behavior to that of the PASCAL dataset, therefore considerations made earlier apply also in this case.
In Figures 5.17 we reported some examples of the produced output map.
Despite the improvement in the mean IoU, in this dataset we cannot appreciate a clear visual improvement of the segmentation maps in a large scale. However if we focus on the images on rows 3 and 5 of Figure 5.17 we can see that the proposed method is producing better results in the segmentation of the classes

terrain and *fence* respectively.

Additional output segmentation maps produced on the Cityscapes validation set can be found in Appendix A.

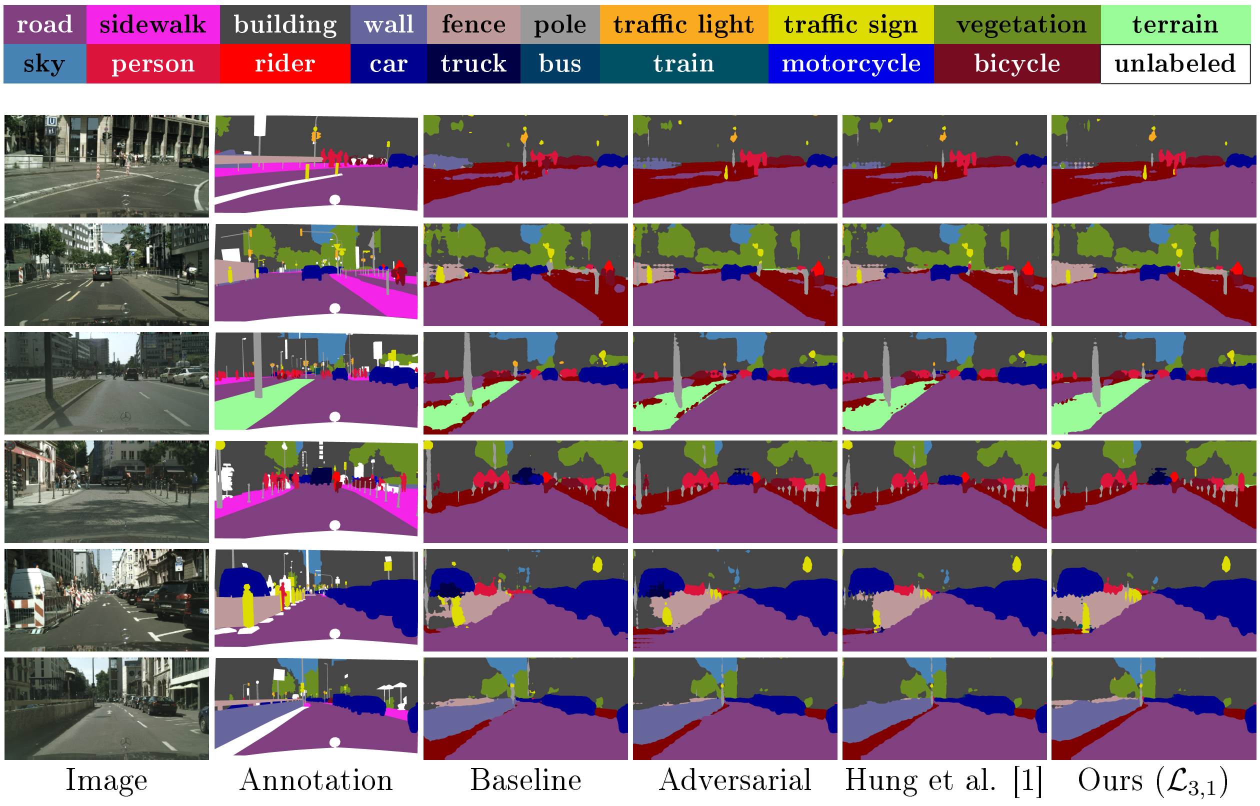

FIGURE 5.17: Examples of correct semantic segmentation maps of some sample scenes extracted from the Cityscapes validation dataset. The network is trained using half of the dataset as annotated data and the remaining as unsupervised data.

5.4 Domain Adaptation

The results obtained by the semi-supervised framework and the improvements achieved by the proposed modified loss have moved this study to a different but correlated branch of research. Domain adaptation, as discussed in previous chapters, is a form of unsupervised learning that aims at performing a domain shift between two data distributions. We tested this framework to perform a novel unsupervised domain adaptation strategy to adapt a deep network trained on synthetic data to real world scenes.
To evaluate the performances on this task we performed two different sets of experiments. In the first experiment we trained the network using the scenes from the GTA5 dataset to compute the supervised loss (i.e. $\mathcal{L}_1$) and the adversarial loss (i.e. $\mathcal{L}_2^s$). Then we used the training scenes of the Cityscapes dataset for the unsupervised domain adaptation, i.e., no labels from Cityscapes have been used and when dealing with this dataset we only computed the losses $\mathcal{L}_2^u$ and $\mathcal{L}_3$. Finally we evaluated the performances on the original validation set of Cityscapes.
In the second experiment we performed the same procedure but we replaced the GTA5 dataset with the SYNTHIA one.
In the task of domain adaptation the semantic labels of the Cityscapes dataset have been used just for test purposes, and the full training set is used without annotation to perform unsupervised learning.
To test the effectiveness of the proposed approach, as done for the single dataset, we performed 4 different tests in which we use different combinations of loss term to optimize the network:

- supervised baseline: $\mathcal{L}_1$ (indicated as ***Baseline***)
- supervised adversarial: $\mathcal{L}_1 + \lambda_{adv}^s \cdot \mathcal{L}_2^s$ (indicated as ***Adversarial***)
- semi-supervised framework: $\mathcal{L}_1 + \lambda_{adv}^s \cdot \mathcal{L}_2^s + \lambda_{adv}^u \cdot \mathcal{L}_2^u + \mathcal{L}_3$ (indicated as ***Hung et al. [1]***)
- proposed method: $\mathcal{L}_1 + \lambda_{adv}^s \cdot \mathcal{L}_2^s + \lambda_{adv}^u \cdot \mathcal{L}_2^u + \mathcal{L}_{3,2}$ (indicated as ***Ours***)

We have not reported the results with the ($\mathcal{L}_{3,1}$) loss since, as discussed in Chapter 4, in the task of domain adaptation it has bad visual results compared to the other techniques.
For all the tests we trained the network for 20000 steps with a batch composed of only one image to fit the memory constraints.

Results on GTA5 Dataset

In order to measure the performance we compared the predictions on the original Cityscapes validation set with the ground truth labels and computed the mean IoU as done in the semi-supervised experiments and by the majority of the competing approaches [48, 51, 50].
Table 5.5 shows the results of the proposed approach when exploiting different domain adaptation strategies and compares them with some state-of-the-art approaches for domain adaptation.
As a first experiment we trained the network in a supervised way on the GTA5 dataset and then we tested it on real world data from the Cityscapes dataset. In particular using only the loss term $\mathcal{L}_1$ we obtained a mean IoU of 27.9%. The addition of the adversarial loss (i.e. $\mathcal{L}_2^s$) on the synthetic data further improves the mean IoU to 29.3%. This confirms the results obtained on the single dataset in which we observed that even the influence of the discriminator on supervised data can bring an overall improvement on the segmentation maps.
Then we trained the model with unsupervised data using the framework of Hung et al. [1] obtaining a mean IoU of 29%. Observing more in detail the various class accuracy it is possible to see that the accuracy has increased on some of the most common classes corresponding to large structures (*road*, *building*, *sky*, *car*), while the behaviour on low frequency classes corresponding to small objects is more unstable (some improve but others have a lower accuracy).
As discussed in Chapter 4, this is the reason for the introduction of the modified loss term (i.e $\mathcal{L}_{3,2}$). Thanks to this when using the full framework with all the loss terms the mean IoU increases to 30.4% and in particular it is possible to appreciate a large performance boost on many uncommon classes corresponding to small objects and structures.
By comparing with state-of-the-art approaches it is possible to see how the method of Hung et al. [1] has lower accuracy than our approach, mostly because it struggles with small structures and uncommon classes. The method of [48] has even lower performances, however it is also based on a different generator network (i.e, the method of [5]).

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Baseline	45.3	20.6	50.1	9.3	**12.7**	19.5	4.3	0.7	81.9	21.1	63.3	52.0	1.7	77.9	26.0	39.8	0.1	4.7	0.0	27.9
Adversarial	61.0	18.5	51.6	15.4	12.3	**20.5**	1.4	0.0	82.6	24.7	61.0	52.1	2.2	78.5	25.9	41.5	0.4	8.0	0.1	29.3
Ours ($\mathcal{L}_{3,2}$)	54.9	23.8	50.9	**16.2**	11.2	20.0	3.2	0.0	79.7	**31.6**	64.9	**52.5**	**7.9**	79.5	27.2	41.8	**0.5**	**10.7**	**1.3**	**30.4**
Hoffman et al. [48]	70.4	**32.4**	62.1	14.9	5.4	10.9	**14.2**	**2.7**	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1
Hung et al. [1]	**81.7**	0.3	**68.4**	4.5	2.7	8.5	0.6	0.0	**82.7**	21.5	**67.9**	40.0	3.3	**80.7**	**34.2**	**45.9**	0.2	8.7	0.0	29.0

TABLE 5.5: Mean intersection over union (mIoU) on the different classes of the original Cityscapes validation set. The approaches have been trained in a supervised way on the GTA5 dataset and then the unsupervised domain adaptation has been performed using the Cityscapes training set.

Figure 5.18 shows the output examples of the different versions of our approach and of the method of [1] on some sample scenes. The supervised training leads to reasonable results but some small objects get lost or have a wrong shape (e.g., the *riders* in row 1). Furthermore, some regions of the street and of structures like the walls are corrupted by noise (see the *street* in the last two rows or the *fence* on the right in row 3). The adversarial loss $\mathcal{L}_2^s$ reduces these artifacts but there are still issues on the small objects (e.g., the *rider* in the fifth row) and the boundaries are not always very accurate (see the *fence* in the third row). The complete model leads to better performances, for example in the images of Figure 5.18 the people are better preserved and the structures have better defined edges. Finally the approach of [1] seems to lose some structures (e.g., the *fence* in the third row) and has issues with the small objects (the *riders* in row 5 get completely lost) as pointed out before.
Additional output segmentation maps produced on the Cityscapes validation set can be found in Appendix A.

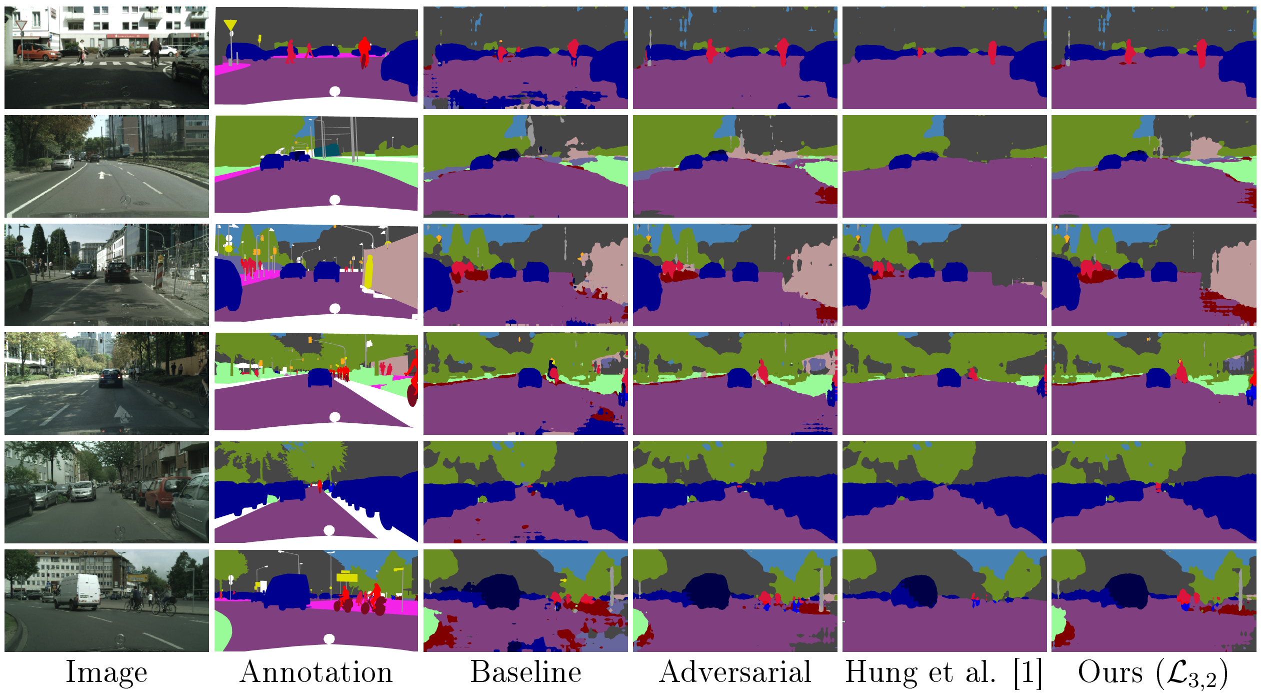

FIGURE 5.18: Examples of correct semantic segmentation maps of some sample scenes extracted from the Cityscapes validation dataset. The network has been trained using GTA5 with annotations and Cityscapes for the unsupervised part.

Results on SYNTHIA Dataset

By using the SYNTHIA dataset as source dataset, the domain adaptation task is even more challenging if compared with the GTA5 case since the computer generated graphics are less realistic. Table 5.6 shows that by training the generator network in a supervised way on the SYNTHIA dataset and then testing on the real world Cityscapes dataset, a mean IoU of 25.4% can be obtained. This value is smaller than the mean IoU of 27.9% obtained by training the generator on the GTA5 dataset. This result confirms that the GTA5 dataset has a smaller domain shift with respect to real world data, when compared with the SYNTHIA dataset (GTA5 data, indeed, have been produced by a more advanced rendering engine with more realistic graphics).
Under this training scenario, the adversarial loss (i.e. $\mathcal{L}_2^s$) does not bring to noteworthy improvements in the domain adaptation task, indeed the mean IoU is equal to the ones obtained without the contribution of the discriminator.
As in GTA5 dataset we trained the model with unsupervised data using the framework of Hung et al. [1] obtaining a mean IoU of 29.4%.
Using the proposed framework with the modified loss term (i.e $\mathcal{L}_{3,2}$) a noticeable improvement to 30.4% of mean IoU can be observed.
The method of [48] appears to be again the less performing approach. In this

comparison, it is even less accurate than our *baseline*, but it employs a different segmentation network.

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	sky	person	rider	car	bus	motorcycle	bicycle	mean
Baseline	10.3	**20.5**	35.5	1.5	0.0	28.9	0.0	1.2	83.1	74.8	53.5	**7.5**	65.8	18.1	4.7	1.0	25.4
Adversarial	9.3	19.3	33.5	0.9	0.0	**32.5**	0.0	0.5	82.3	76.9	**54.7**	5.5	64.9	17.0	**5.7**	**3.9**	25.4
Ours ($\mathcal{L}_{3,2}$)	**78.4**	0.1	**73.2**	0.0	0.0	16.9	0.0	0.2	84.3	**78.8**	46.0	0.3	74.9	30.8	0.0	0.1	**30.2**
Hoffman et al. [48]	11.5	19.6	30.8	**4.4**	0.0	20.3	**0.1**	**11.7**	42.3	68.7	51.2	3.8	54.0	3.2	0.2	0.6	20.1
Hung et al. [1]	72.5	0.0	63.8	0.0	0.0	16.3	0.0	0.5	**84.7**	76.9	45.3	1.5	**77.6**	**31.3**	0.0	0.1	29.4

TABLE 5.6: Mean intersection over union (mIoU) on the different classes of the original Cityscapes validation set. The approaches have been trained in a supervised way on the SYNTHIA dataset and then the unsupervised domain adaptation has been performed using the Cityscapes training set.

Figure 5.19 shows the output segmentation maps of the different techniques on some sample scenes. The first thing that can be noticed by looking at the qualitative results of the *baseline* supervised version is that by training on the SYNTHIA dataset some classes as *sidewalk* and *road* are highly corrupted. It is evident that a simple synthetic supervised training starting from this dataset would bring to a network which can not be used in an autonomous vehicle scenario. This is probably caused by the not completely realistic representation of streets and sidewalks in the SYNTHIA dataset, where their textures are often very unrealistic. Additionally, while the position of the camera in the Cityscapes dataset is always fixed inside the car, in SYNTHIA the camera can assume different positions, for example the view can be done from inside the car, from cameras looking from the top or from the side of the road. Similarly to the *baseline* approach, the adversarial loss $\mathcal{L}_2^s$ is unable to adapt the network to the real domain, indeed the class *road* remains very badly detected also after its usage. Differently, unsupervised data and the self-teaching component of allows to avoid all the artifacts on the *road* surface by reinforcing the segmentation network to capture the real nature of this class in the Cityscapes dataset. Also Hung's method [1] is able to correctly reconstruct the class *road*, avoiding the noise present in the *baseline*, but it suffers on small classes where it is outperformed by the proposed method. This is clearly visible on rows 5 and 6 of Figure 5.19, where our method is able to locate more precisely small classes as *person*.

Additional output segmentation maps produced on the Cityscapes validation set can be found in Appendix A.

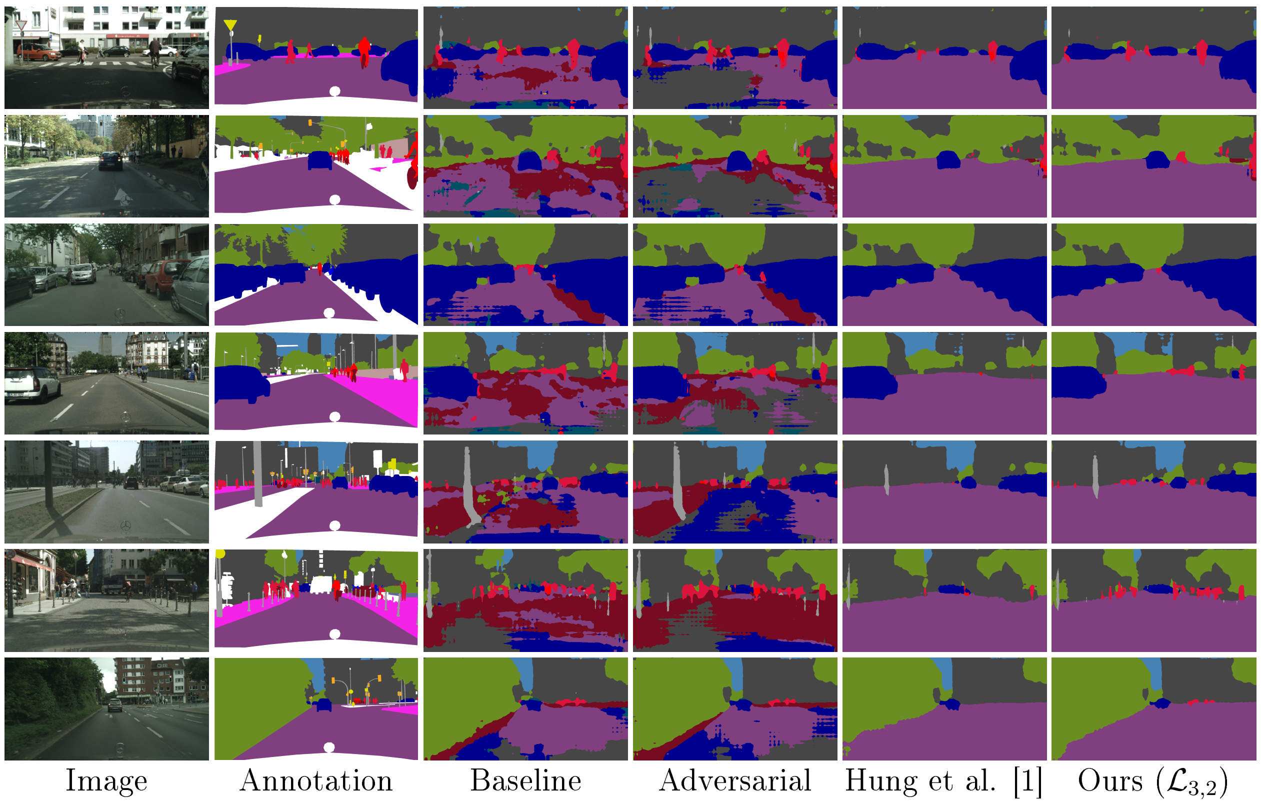

FIGURE 5.19: Examples of correct semantic segmentation maps of some sample scenes extracted from the Cityscapes validation dataset. The network has been trained using SYNTHIA with annotations and Cityscapes for the unsupervised part.

6 Conclusions

In this work we studied the use of adversarial training to improve the performances of semantic segmentation networks using unsupervised data. In particular, we developed a framework that exploits the information of unlabeled data to boost the performance of a state-of-the-art network for semantic segmentation. This was made possible by using a fully convolutional discriminator to produce a confidence map that has been used to reinforce the learning in areas with high confidence. Experimental results demonstrate the effectiveness of this approach when using data coming from the same domain of the labeled data, moreover, the proposed loss term brings a further improvement compared to the competing approach.

Furthermore we applied the developed framework to unsupervised domain adaptation from synthetic urban scenes to real world ones. Experimental results on a real world dataset prove the effectiveness of the proposed approach with different source datasets. In particular, we obtained good results also on challenging uncommon classes thanks to the proposed loss weighting term. The results of this approach have been published in [59]

Regarding future works, a fine-tuning of the network optimization parameters can be done in order to increase the performances of the domain adaptation framework.

In addition, some variations of discriminator network could be tested to improve the reliability of the produced confidence maps. Moreover, we could test some different reparameterizations of the discriminator input in order to reduce the difference between generated and real segmentation maps.

Finally, further research could be devoted to the improvement of the self-teaching strategy and to the exploitation of generative models to produce more realistic and refined synthetic training data to be fed to the framework for domain adaptation.

7 Appendix A

Backgound	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus
Car	Cat	Chair	Cow	Dining-Table	Dog	Horse
Motorbike	Person	Potted_Plant	Sheep	Sofa	Train	TV/Monitor

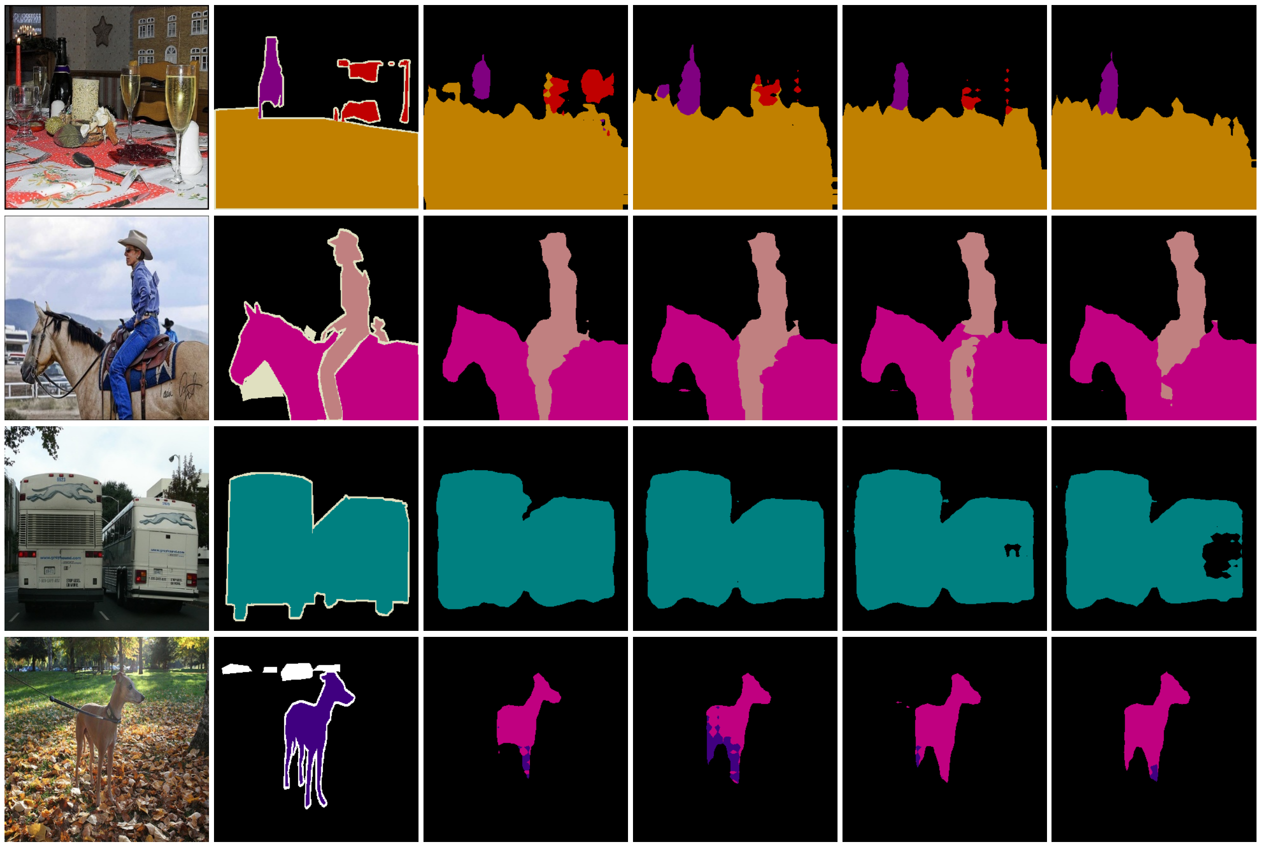

FIGURE 7.1: Extra examples of incorrect semantic segmentation of some sample scenes extracted from the PASCAL VOC2012 validation dataset. The network is trained using half of the dataset as annotated data and the remaining as unsupervised data

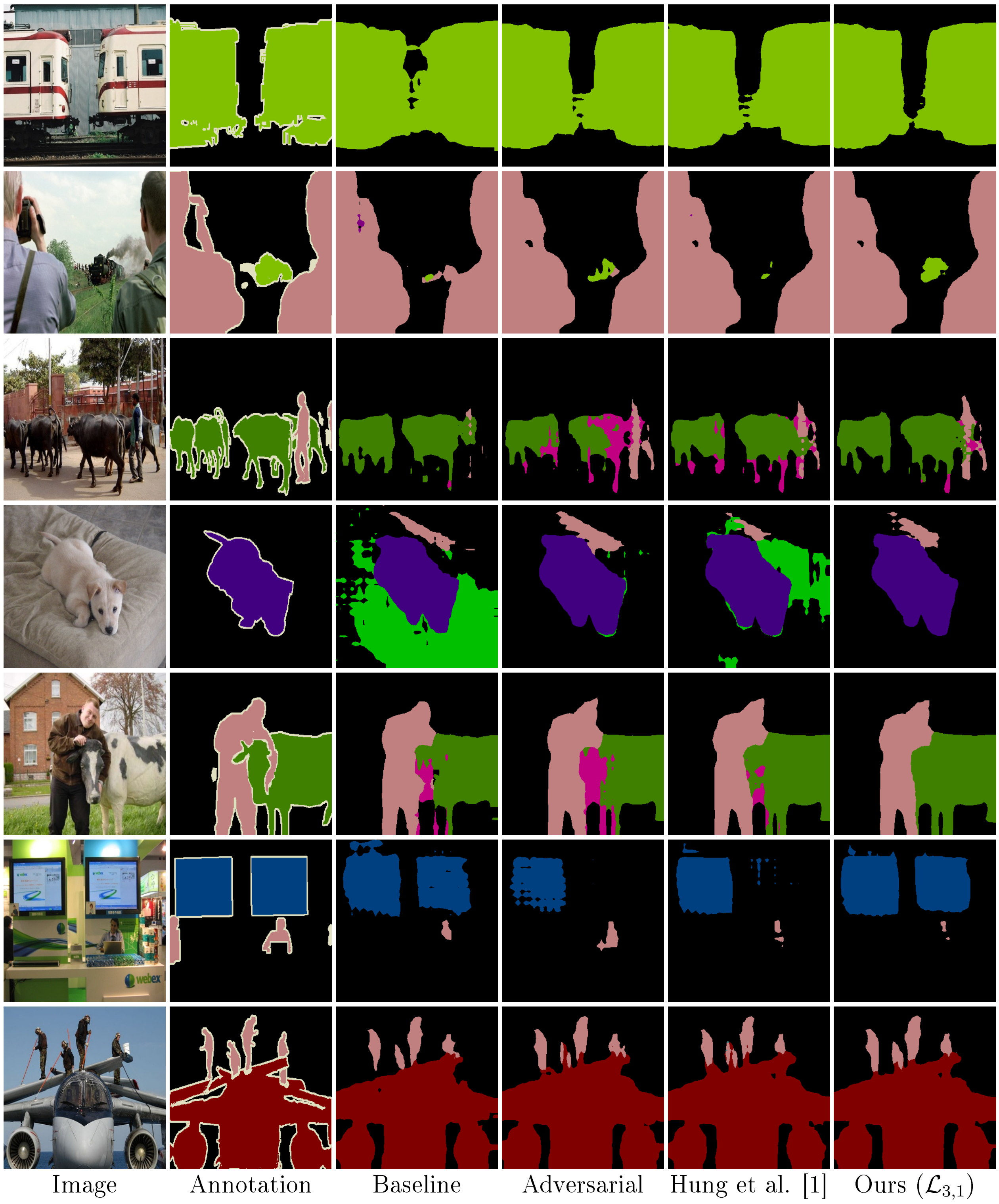

FIGURE 7.2: Extra examples of correct semantic segmentation of some sample scenes extracted from the PASCAL VOC2012 validation dataset. The network is trained using half of the dataset as annotated data and the remaining as unsupervised data

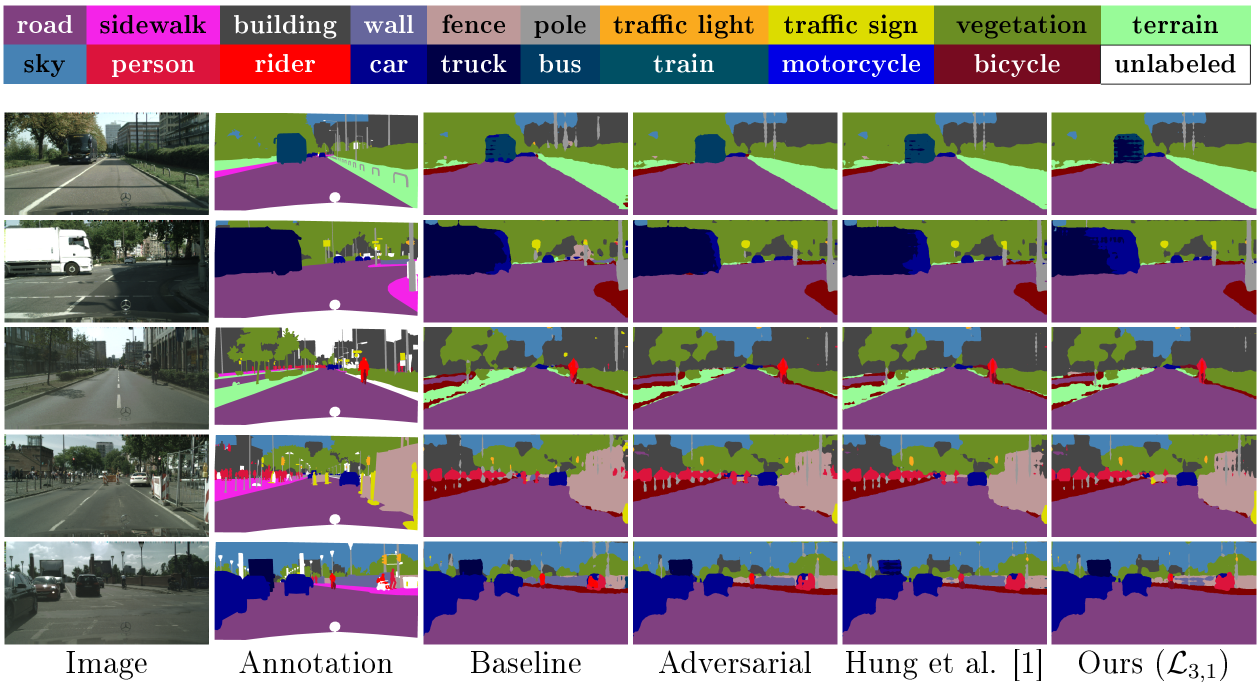

FIGURE 7.3: Extra examples of incorrect semantic segmentation maps of some sample scenes extracted from the Cityscapes validation dataset. The network is trained using half of the dataset as annotated data and the remaining as unsupervised data

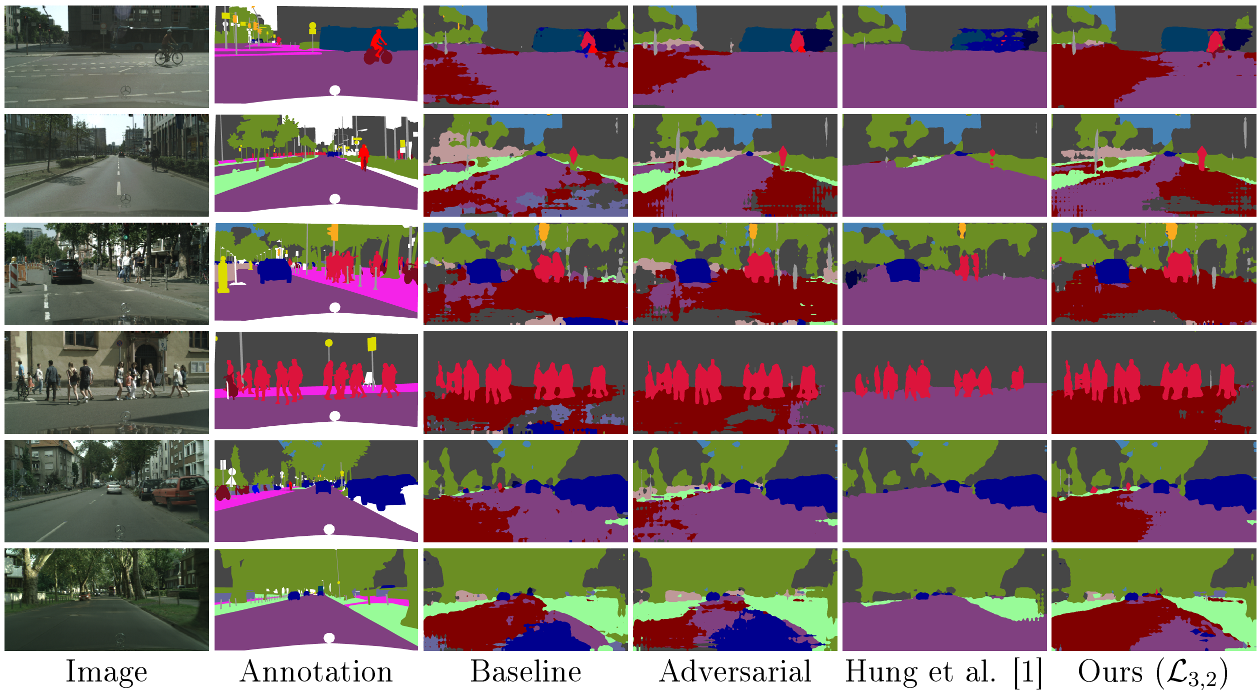

FIGURE 7.4: Extra examples of incorrect semantic segmentation maps of some sample scenes extracted from the Cityscapes validation dataset. The network has been trained using GTA5 with annotations and Cityscapes for the unsupervised part.

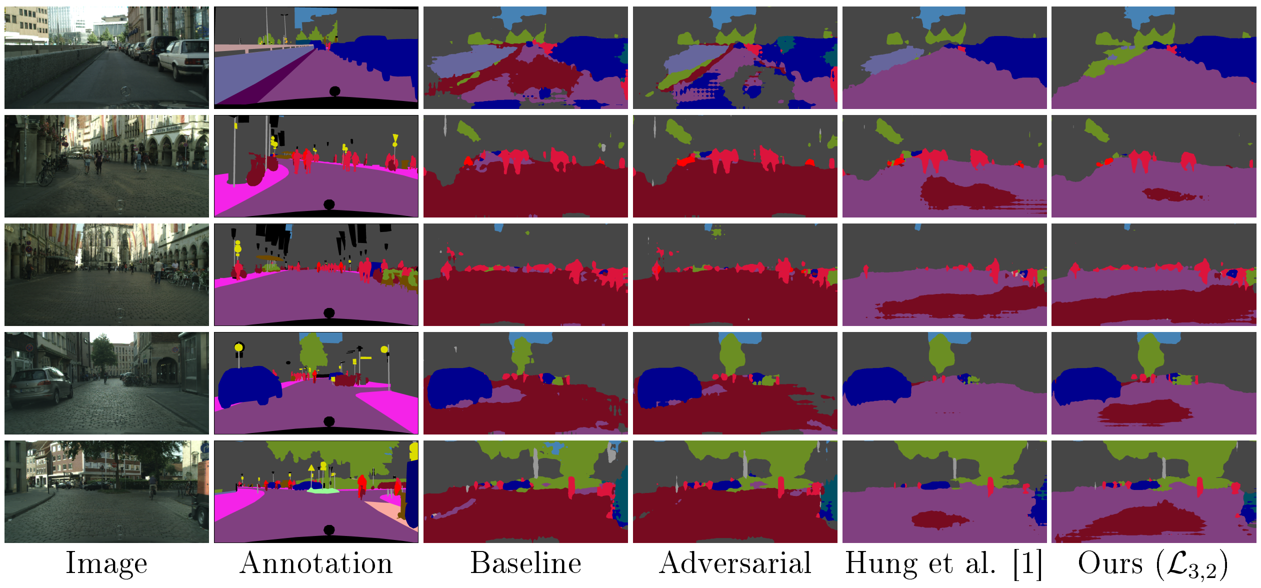

FIGURE 7.5: Extra examples of incorrect semantic segmentation maps of some sample scenes extracted from the Cityscapes validation dataset. The network has been trained using SYNTHIA with annotations and Cityscapes for the unsupervised part.

8 Appendix B

The biggest problem of domain adaptation between synthetic and a real world dataset is the intrinsic difference between the two. Computer generated graphics often presents an homogeneous structure while real world data is corrupted by noise caused by multiple factors.
We tried to reduce the domain shift between the synthetic and the real dataset adding some noise to the input images during the training on the domain adaptation framework.
As a preliminary test we used a simple Gaussian noise added to the images of GTA5 dataset. In particular we used a Gaussian distribution with $\mu = 0$ and $\sigma = 5$.

	Mean IoU	Mean IoU with Noise
Baseline	27.9	27.7
Adversarial	29.3	29.6
Hung et al. [1]	29.0	25.3
Ours	30.4	25.1

TABLE 8.1: Mean IoU for different techniques evaluated on Cityscapes validation set. The results on the second column are produced with the addition of a Gaussian noise in the synthetic images.

As we can see from the results in Table 8.1 the addition of the noise to the synthetic images does not affect too much the performances of the network when using only synthetic data. On the contrary when including real data to the training process, the performances of the network has very bad results compared to the version without the addition of noise.
This can suggest that the Gaussian distribution is not the more indicated to model the domain shift between the two datasets.
Further experiments can be made to investigate different type of noise to make the synthetic data more similar to the real ones.
Moreover a more advanced approach could involve GANs to generate more realistic synthetic data to use in the training process e.g. following the work proposed in [40].

List of Figures

List of Tables

Bibliography

[1] Wei-Chih Hung et al. "Adversarial Learning for Semi-Supervised Semantic Segmentation". In: (2018).

[2] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[4] Rikiya Yamashita et al. "Convolutional neural networks: an overview and application in radiology". In: *Insights into Imaging* 9.4 (2018), pp. 611–629. ISSN: 1869-4101.

[5] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *ICLR*. 2016.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[7] Diederik P Kingma and Jimmy Ba. *Adam: a method for stochastic optimization. ICLR (2015)*. 2015.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[9] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[10] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

[11] Alberto Garcia-Garcia et al. "A review on deep learning techniques applied to semantic segmentation". In: *arXiv preprint arXiv:1704.06857* (2017).

[12] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[14] Tsung-Yi Lin et al. “Feature pyramid networks for object detection”. In: *CVPR*. Vol. 1. 2. 2017, p. 4.

[15] Hengshuang Zhao et al. “Pyramid scene parsing network”. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2881–2890.

[16] Liang-Chieh Chen et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2018), pp. 834–848.

[17] Philipp Krähenbühl and Vladlen Koltun. “Efficient inference in fully connected CRFs with gaussian edge potentials”. In: *Advances in neural information processing systems*. 2011, pp. 109–117.

[18] Ian Goodfellow et al. “Generative adversarial nets”. In: *NIPS*. 2014, pp. 2672–2680.

[19] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *CoRR* abs/1511.06434 (2015).

[20] Pauline Luc et al. “Semantic Segmentation using Adversarial Networks”. In: *NIPS Workshop on Adversarial Training*. 2016.

[21] Marius Cordts et al. “The Cityscapes dataset for semantic urban scene understanding”. In: *CVPR*. 2016, pp. 3213–3223.

[22] G.J. Brostow, J. Fauqueur, and R. Cipolla. “Semantic object classes in video: A high-definition ground truth database”. In: *Pattern Recognition Letters* (2 2009), pp. 88–97.

[23] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. “Constrained convolutional neural networks for weakly supervised segmentation”. In: *ICCV*. 2015, pp. 1796–1804.

[24] Alexander Vezhnevets and Joachim M Buhmann. "Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning". In: *CVPR*. IEEE. 2010, pp. 3249–3256.

[25] Yunchao Wei et al. "STC: A simple to complex framework for weakly-supervised semantic segmentation". In: *PAMI* 39.11 (2017), pp. 2314–2320.

[26] Seunghoon Hong, Hyeonwoo Noh, and Bohyung Han. "Decoupled deep neural network for semi-supervised semantic segmentation". In: *Advances in neural information processing systems*. 2015, pp. 1495–1503.

[27] Jifeng Dai, Kaiming He, and Jian Sun. "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1635–1643.

[28] Zilong Huang et al. "Weakly-supervised semantic segmentation network with deep seeded region growing". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7014–7023.

[29] Fengdong Sun and Wenhui Li. "Saliency guided deep network for weakly-supervised image segmentation". In: *Pattern Recognition Letters* (2019).

[30] Swami Sankaranarayanan et al. "Learning from synthetic data: Addressing domain shift for semantic segmentation". In: *CVPR*. 2018, pp. 3752–3761.

[31] Xiaoming Liu et al. "Semi-Supervised Automatic Segmentation of Layer and Fluid Region in Retinal Optical Coherence Tomography Images Using Adversarial Learning". In: *IEEE Access* 7 (2019), pp. 3046–3061.

[32] George Papandreou et al. "Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1742–1750.

[33] Nasim Souly, Concetto Spampinato, and Mubarak Shah. "Semi and weakly supervised semantic segmentation using generative adversarial network". In: *arXiv preprint arXiv:1703.09695* (2017).

[34] Stephan R. Richter et al. "Playing for Data: Ground Truth from Computer Games". In: ed. by Bastian Leibe et al. Vol. 9906. LNCS. Springer International Publishing, 2016, pp. 102–118.

[35] German Ros et al. "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes". In: *CVPR*. 2016, pp. 3234–3243.

[36] Tatiana Tommasi et al. “A deeper look at dataset bias”. In: *Domain Adaptation in Computer Vision Applications*. Springer, 2017, pp. 37–55.

[37] A Torralba and AA Efros. “Unbiased look at dataset bias”. In: *CVPR*. IEEE Computer Society. 2011, pp. 1521–1528.

[38] Boqing Gong, Fei Sha, and Kristen Grauman. “Overcoming dataset bias: An unsupervised domain adaptation approach”. In: *NIPS Workshop on Large Scale Visual Recognition and Retrieval*. Vol. 3. Citeseer. 2012.

[39] Aditya Khosla et al. “Undoing the damage of dataset bias”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 158–171.

[40] Ashish Shrivastava et al. “Learning from simulated and unsupervised images through adversarial training”. In: *CVPR*. 2017, pp. 2107–2116.

[41] Xingchao Peng and Kate Saenko. “Synthetic to real adaptation with generative correlation alignment networks”. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1982–1991.

[42] Xiaolong Wang and Abhinav Gupta. “Generative image modeling using style and structure adversarial networks”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 318–335.

[43] Jun-Yan Zhu et al. “Generative visual manipulation on the natural image manifold”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 597–613.

[44] Daniel Jiwoong Im et al. “Generating images with recurrent adversarial networks”. In: *arXiv preprint arXiv:1602.05110* (2016).

[45] Yaroslav Ganin and Victor Lempitsky. “Unsupervised Domain Adaptation by Backpropagation”. In: *International Conference on Machine Learning*. 2015, pp. 1180–1189.

[46] Yaroslav Ganin et al. “Domain-adversarial training of neural networks”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2096–2030.

[47] Eric Tzeng et al. “Simultaneous deep transfer across domains and tasks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4068–4076.

[48] Judy Hoffman et al. “FCNs in the wild: Pixel-level adversarial and constraint-based adaptation”. In: *arXiv preprint arXiv:1612.02649* (2016).

[49] Yang Zhang, Philip David, and Boqing Gong. “Curriculum domain adaptation for semantic segmentation of urban scenes”. In: *ICCV*. 2017, pp. 2020–2030.

[50] Yi-Hsuan Tsai et al. “Learning to adapt structured output space for semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7472–7481.

[51] Yuhua Chen, Wen Li, and Luc Van Gool. “Road: Reality oriented adaptation for semantic segmentation of urban scenes”. In: *CVPR*. 2018, pp. 7892–7901.

[52] Yang Zou et al. “Unsupervised domain adaptation for semantic segmentation via class-balanced self-training”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 289–305.

[53] Yuhua Chen et al. “Learning Semantic Segmentation from Synthetic Data: A Geometrically Guided Input-Output Adaptation Approach”. In: *arXiv preprint arXiv:1812.05040* (2018).

[54] Yiheng Zhang et al. “Fully convolutional adaptation networks for semantic segmentation”. In: *CVPR*. 2018, pp. 6810–6818.

[55] Swami Sankaranarayanan et al. “Unsupervised domain adaptation for semantic segmentation with GANs”. In: *arXiv preprint arXiv:1711.06969* (2017).

[56] Judy Hoffman et al. “CyCADA: Cycle-Consistent Adversarial Domain Adaptation”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018.

[57] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[58] Mark Everingham et al. “The Pascal Visual Object Classes (VOC) challenge”. In: *International journal of Computer Vision* 88.2 (2010), pp. 303–338.

[59] Matteo Biasetton et al. “Unsupervised Domain Adaptation for Semantic Segmentation of Urban Scenes”. In: *Proceedings of the Int. Conference of Computer Vision and Pattern Recognition Workshop (CVPRW): Workshop on Autonomous Driving (WAD)* (2019). URL: http://lttm.dei.unipd.it/paper_data/semanticDA/.