Adversarial Domain Adaptation for Real-time Semantic Segmentation

Giulia D'Ascenzi
Politecnico di Torino
s287720@studenti.polito.it

Patrizio de Girolamo
Politecnico di Torino
s292497@studenti.polito.it

Carlos Rosero
Politecnico di Torino
s293625@studenti.polito.it

Abstract

Semantic segmentation is a crucial challenge in the field of computer vision. Traditional approaches rely on large models and real-world datasets, which respectively demand substantial resources and a labor intensive process, therefore not suitable for real-time applications. It is imperative to develop efficient, real-time models which can be trained using automatically-labeled synthetic datasets. In this paper, we combine adversarial learning using a lightweight model with a simple Fourier Transformation to narrow the domain gap and increase semantic segmentation performances. We evaluate our approach on the GTA5 to Cityscapes benchmark. [1]

1. Introduction

Semantic segmentation is the task of assigning a category label to each pixel of an input image. It is a challenging task in computer vision and it is beneficial to many applications, such as autonomous vehicles, virtual reality, and computer-aided diagnosis. Convolutional neural networks (CNNs), trained in a supervised fashion, are generally used to solve this task. The significant disadvantage with this approach is how costly it is: CNN-based models contain a high number of parameters, require numerous floating point operations (FLOPS), and rely on massive fine-annotated datasets. These drawbacks make these kinds of models unsuitable for real-time applications.

The first challenge to solve is the scalability of the annotation process: it requires a large amount of human effort in terms of time and money. Moreover, capturing scenes under varying circumstances, with different weather and visibility conditions, makes the task even more unfeasible. Given how expensive this is, photo-realistic simulation software and images from modern computer games can prove to be affordable alternatives, where the desired scenarios can be created and the labels generated through mainly automated processes. Despite this, directly transferring the knowledge acquired from the labeled source (synthetic) datasets to the unlabeled target (real-world) datasets produces a drop in prediction performance. This happens due to the domain shift between the real world and its simulation, despite how photo-realistic the latter may be. Therefore, domain adaptation needs to take place in order to reduce the performance deficit and take advantage of synthetic data generation. To tackle this problem, in this work we used an adversarial based architecture, where the model is trained through a two-player game between a generator, which produces the segmentation predictions for both the target and the source images, and a discriminator, which tries to distinguish whether the input comes from the source or the target domain. The goal of the generator is, in order to fool the discriminator, to generate similar data distributions in the output space for both types of images.

An additional way to decrease the domain gap between source and target domain is by aligning the low-level statistics between the source and target distributions. To do so, we adopted the Fourier Domain Adaptation technique proposed by Yang et. al. [22] as the first stage in our proposed pipeline, increasing the performances of the overall model.

The second challenge is to reduce the complexity and the number of parameters used in current networks, in order to achieve high performances in real-life applications. To do so, we used BiSeNet, designed by Yu et al. [23], as the generator for the domain adaptation task. This model simultaneously preserves spatial details of the input images and captures semantics. Also, to make the discriminator network lighter we implemented a faster and thinner variant, using depth-wise separable convolutions instead of classical 2D-convolutions.

In summary, in this work we propose an architecture that narrows the domain gap, first in the low-level statistics, then in the output space, increasing semantic segmentation performances in a lightweight and efficient manner.

In the following sections, we provide the context of current literature for our work, describe our method, and test it in a standard unsupervised domain adaptation benchmark (GTA5 to Cityscapes).

[1]Code available at: https://github.com/CRosero/aml-project-gpc

2. Related Work

2.1. Semantic segmentation

State-of-the-art approaches mainly rely on deep neural networks. The pioneers were Long et al. [15], who proposed to use a fully–convolutional network (FCN) for semantic segmentation. To date, numerous deep-learning approaches have utilized different techniques to increase the performances for this task, such as incorporating higher-level context [24], enlarging receptive fields [14] or fusing multi-scale features [25]. Unfortunately, there are many drawbacks to standard deep learning models for semantic segmentation, which hinder real-life applicability in many domains.

First of all, convolutional neural networks for semantic segmentation need to be trained on a large amount of densely labeled images. These are usually annotated manually through an expensive and time-consuming process. To tackle this, one recent approach is to use large-scale synthetic datasets based on rendering, such as GTA5 [17], where partially automated processes can be used to annotate the data. Unfortunately, a model trained on synthetic data, despite how photo-realistic it may be, does not generalize well to real-world datasets, due to the domain shift between the datasets. It is thus necessary to perform domain adaptation to narrow the performance gap.

The second challenge is to overcome the complexity and the number of parameters of the current networks without worsening their performances, but making them suited for real-life applications that demand a higher time efficiency and a lower usage of resources.

2.2. Unsupervised Domain Adaptation

Domain adaptation (DA) is a machine learning paradigm that aims to learn a model using a labeled source domain, which can then also generalize to a different, but related, target domain. The difference in data distribution between domains makes it difficult to transfer knowledge from one to the other. Thus, the main task of these algorithms is to reduce the domain discrepancy between the source and the target dataset.

Domain adaptation can be supervised, semi-supervised or unsupervised depending on the presence of target annotations during training. This work uses an Unsupervised Domain Adaptation (UDA) approach, thus relying on a labeled source dataset and an unlabeled target dataset. Under this approach, different strategies have been proposed. Some follow discrepancy-based methods, minimizing the domain distribution discrepancy, and measuring it e.g. with the MMD (Maximum Mean Discrepancy) [19]. Another approach used in recent research, and also followed by this paper, is to tackle the problem using Adversarial Learning as done by Tsai et al. [20]. This method constructs a minimax game between a discriminator, trained to distinguish whether the output comes from the source or target dataset, and a generator, which tries to emit features that confuse the discriminator. Jointly training these two components leads to final results that are less sensible to domain shift, with distributions closer to each other across the two domains.

Other alternatives are to use adversarial discriminative models as done by Ganin et al. [8] or to reduce the domain gap using image-to-image translation and style transfer. Yang et al. [22] used the Fourier Transform to translate images in the source domain to the style of the target domain. Li et al. [13] instead used a bidirectional learning framework to learn alternatively the image translation model and the segmentation adaptation model, while promoting each other.

2.3. Real-Time Semantic Segmentation

The majority of semantic segmentation applications require high inference speed, but many deep-learning-based semantic segmentation models, due to their large architectures and numerous parameters, are unable to accomplish this. To accelerate inference speeds, approaches that have been followed include cropping or resizing the input images [21], pruning the channels of the network [2], or dropping the last stage in pursuit of an extremely tight framework [16]. These approaches fail in maintaining spatial details. To remedy this phenomenon, the U-Shape structure [18] was proposed. However, it reduces the speed of the model due to the increase of computation on high-resolution feature maps. Moreover, most spatial information lost in the pruning or cropping cannot be recovered. Due to these observations, this work uses the Bilateral Segmentation Network (BiSeNet), proposed by Yu et al. [23]. It adopts a multi-path framework, combining the low-level details and high-level semantics, to increase inference speed with only a slight decrease in segmentation performance.

3. Method

Given the source dataset S (e.g., synthetic data) with segmentation labels Y_S and the target dataset T (e.g., real data) with no labels, the goal is to train a network for real-time semantic segmentation using S and test it on T. The network used is BiSeNet (section 3.1) and the goal is to make its performances to be as close as the upper bound got by the model trained on T with ground truth labels. To address the problem of the domain shift between the two datasets, we implemented unsupervised adversarial domain adaptation comparing the performances, the number of parameters and the floating point operations (FLOPS) when using a fully convolutional discriminator and a lightweight one (section 3.2). Lastly, we increased the domain alignment and the adaptation performances using the Fourier Domain

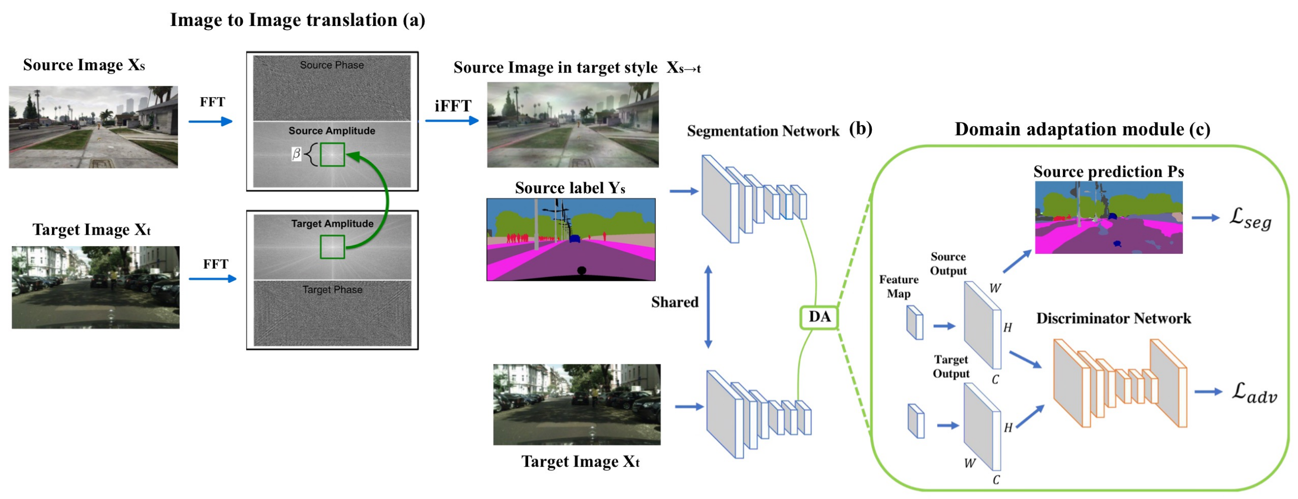


Figure 1. *Overview of the proposed architecture.*The input images from source and target domains pass through three phases: (a) image to image translation applying FDA [22] on the source images (b) segmentation done by the BiSeNet network [23] (c) domain adaptation following the approach proposed by Tsai et al. [20].

Adaptation technique (section 3.3). An overview of the full architecture of our model is described in Figure 1.

3.1. BiSeNet

The task of performing semantic segmentation accurately in real-time speed is challenging due to high computational cost of the deep-learning methods in terms of memory and time. To perform real-time semantic segmentation in this project, we used BiSeNet [23]. BiSeNet captures semantics and preserves spatial details of the input images, simultaneously entrusting them to two sub-networks, respectively to the context path and to the spatial path.

The spatial path is composed of only three convolutional layers and extracts feature maps that are 1/8 of the initial image dimension. The context path utilizes a lightweight convolutional neural network model. In the original paper Xception [4] was chosen, while in this work we use ResNet18. The context path down-samples the feature map rapidly and obtains large receptive fields, which encodes high level semantic context information. On the tail, a global average pooling is added to provide the maximum receptive fields with global context information. Afterward, an Attention Refinement Module is used to refine the last two layers of the backbone that are then multiplied by the output of the average pooling layer. Finally, the results of the two paths are intelligently combined using a Feature Fusion Module, since they are different in level of feature representation.

3.2. Unsupervised Adversarial Domain Adaptation

In order to minimize the domain gap between the source and target dataset, we perform unsupervised adversarial domain adaptation on the output level, as presented by Tsai et al. [20]. This method consists of using two networks: a segmentation network, as the generator G that emits the pixel predictions, and a discriminator D, that tries to distinguish whether the original input image belongs to the source or to the target domain. The intuition behind this process it that even if input images belongs to two different domains, their segmentation should share strong similarities.

Having two images, X_s (with annotations Y_s) and X_t (unlabeled) sampled from the source dataset S and the target dataset T, the generator G predicts the segmentation outputs P_s and P_t . The generator G is optimized using a segmentation loss L_{seg} computed on P_s and Y_s using equation 2. The discriminator D takes as input the segmentation P_s and P_t and predicts whether they belong to the target or to the source domain. It uses an adversarial loss L_{adv} on the target prediction P_t and backpropagates gradients to the segmentation network G, encouraging the latter to generate similar segmentation distributions. The adaptation task can be therefore formulated as:

$$\mathcal{L}\left(X_s, X_t\right) = \mathcal{L}_{seg}\left(X_s\right) + \lambda_{adv}\mathcal{L}_{adv}\left(X_t\right) \tag{1}$$

with λ_{adv} serving as the weight to balance both losses. The segmentation loss in (1) is the cross-entropy loss computed using the predictions P_s and the ground-truth labels Y_s:

$$\mathcal{L}_{seg}\left(I_s\right) = -\sum_{h,w}\sum_{c\in C} Y_s^{(h,w,c)} \log\left(P_s^{(h,w,c)}\right). \tag{2}$$

On the other hand, the L_{adv} in (1) is defined as:

$$\mathcal{L}_{adv}\left(I_t\right) = -\sum_{h,w} \log\left(\mathbf{D}\left(P_t\right)^{(h,w,1)}\right). \tag{3}$$

It measures how the discriminator was fooled by the segmentation generated by G. It is used to train the network to improve the quality of the segmentation by maximizing the probability of the target prediction being considered as source prediction.

Finally, the discriminator D is trained using the loss L_d, that can be written as:

$$\begin{aligned}\mathcal{L}_d(P) = -\textstyle\sum_{h,w}(1-z)\log\left(\mathbf{D}(P)^{(h,w,0)}\right)\\ +z\log\left(\mathbf{D}(P)^{(h,w,1)}\right)\end{aligned} \tag{4}$$

where $z = 0$ is a target domain sample while $z = 1$ is a source domain sample.

Lighter Discriminator: In this work we tested and compared the performances of two types of discriminators. Initially, we used a Fully Convolutional discriminator, as Tsai et al. [20] do, whereby each layer performs 2D convolutions. This model suffers from having a huge number of parameters and floating point operations. Therefore, to solve this issue and make our overall Domain Adaptation algorithm more resource-efficient, we tested our algorithm using a Lightweight Convolutional discriminator, replacing the 2D convolutions with depth-wise separable convolutions as done in [10]. This namely results in a faster and smaller model. The main difference between the two convolutions is that 2D convolutions are performed over multiple input channels, whereas in depth-wise convolution, each channel is kept separate, decreasing computation significantly. The difference in terms of number of parameters and FLOPS are shown in Table 1 and Table 2. The lightweight convolutional discriminator has over 14 times less parameters and less FLOPS than the fully convolutional one.

Table 1. *Fully Convolutional Discriminator*: Architecture, total parameters and total number of Floating Point Operations required. Input: batch of 4 images with size 1024x512.

Layer	Output Shape	Param
Conv2d	[4, 64, 512, 256]	19,520
LeakyReLU	[4, 64, 512, 256]	–
Conv2d	[4, 64, 512, 256]	19,520
LeakyReLU	[4, 64, 512, 256]	–
Conv2d	[4, 64, 512, 256]	19,520
LeakyReLU	[4, 64, 512, 256]	–
Total params:		2.7 M
Total FLOPS:		61.8 G

3.3. Improving the performances: FDA

The Fourier Domain Adaptation (FDA) algorithm was presented by Yang et al. [22]. It is a fast image transformation technique to perform domain alignment between source and target images, without requiring any additional training. For this reason, the primary stage of our algorithm consists of applying it on the source dataset input image, as shown in Figure 1. We expect to further reduce the domain gap between the two datasets through this, leading to an increase in overall domain adaptation performance.

Table 2. *Lightweight Convolutional Discriminator*: Architecture, total parameters and total number of Floating Point Operations required. Input: batch of 4 images with size 1024x512.

Layer	Output Shape	Param
DW Conv2d	[4, 19, 512, 256]	323
Conv2d	[4, 64, 512, 256]	1,280
LeakyReLU	[4, 64, 512, 256]	–
DW Conv2d	[4, 19, 512, 256]	323
Conv2d	[4, 64, 512, 256]	1,280
LeakyReLU	[4, 64, 512, 256]	–
DW Conv2d	[4, 19, 512, 256]	323
Conv2d	[4, 64, 512, 256]	1,280
LeakyReLU	[4, 64, 512, 256]	–
Total params:		191 K
Total FLOPS:		4.36 G

Having $X_s \in \mathcal{R}^{H\times W\times 3}$ as source image and $X_t \in \mathcal{R}^{H\times W\times 3}$ as target image, the discrepancy between the low-level statistics of the two is reduced by replacing the low-level frequencies of the target image into the source image. More specifically, first we compute the Fourier Transform $\mathcal{F}$ of the two images using the Fast-Fourier Transformation for each channel of the RGB images.

$$\mathcal{F}(x)[m,n] = \sum_{h,w} x(h,w)e^{-j2\pi(\dfrac{h}{H}m + \dfrac{w}{W}n)}, j^2 = -1 \tag{5}$$

From the two transformations $\mathcal{F}$ can be extracted the amplitude $\mathcal{F}^A$and the phase $\mathcal{F}^P$ components.

Then, we denote as M_β a mask, whose size is the same as the source image and whose value is zero except for the center region where $\beta \in (0, 1)$. Here we assume the center of the image is (0,0).

$$M_\beta(h,w) = \mathbf{1}_{(h,w)\in[-\beta H:\beta H, -\beta W:\beta W]} \tag{6}$$

M_β is used to extract the low frequency part of the target image amplitude, which then is swapped with the corresponding area of the source image amplitude. The phase of the source image is kept unaltered.

$$\mathcal{F}^A(x^{s\to t}) = M_\beta \circ \mathcal{F}^A(x^t) + (1 - M_\beta) \circ \mathcal{F}^A(x^s) \tag{7}$$

$$\mathcal{F}^P(x^{s\to t}) = F^P(x^s) \tag{8}$$

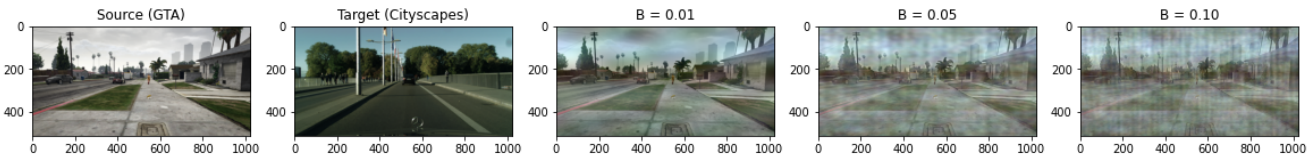


Figure 2. Effect of different values of β when applying FDA on two images sampled from the source and the target domains. Increasing β decreases the domain shift between the transformed source image and the target one, but introduces artifacts.

Finally, applying the inverse Fourier transformation to map $\mathcal{F}^A(x^{s\to t})$ and $\mathcal{F}^P(x^{s\to t})$ back to the image space, we get the transformed image $x^{s\to t}$.

The transformed image $x^{s\to t}$, or "source image in target style" as coined by the authors, will contain the same semantic content of the original one, but will have a style more similar to the target, showing a smaller domain gap.

Hyperparameter β : It denotes the size of the spectral neighborhood to be swapped. Setting $\beta = 0$ is equivalent to not applying any swapping, $x^{s\to t}$ would be the same as the original x^s. On the other hand having $\beta = 1$ leads to completely replace the source amplitude with the target one.

As proposed by the authors in [22], we set $\beta \leq 0.15$. In Table 4 we show the results got by varying it. Also, Figure 2 describes how images change when FDA is applied with a different β.

4. Experiments

4.1. Datasets

The synthetic-to-real benchmark used to test our model is GTA5 to Cityscapes.

GTA5 [17] is composed of 24,966 images with a resolution of 1914x1052, extracted from the "Grand Theft Auto V" videogame. The dataset annotations are compatible with the 19 semantic classes used when evaluating on Cityscapes.

Cityscapes [5] contains 5,000 images with high quality pixel-level annotations as well as 20,000 images with coarse annotations for a total of 25,000 images. The resolution is 2048x1024. The images were acquired in 50 cities, primarily in Germany.

Due to hardware constraints, we used a subset of each dataset. It contains only 500 images for training and only 250 images for evaluation/testing. In the case of Cityscapes, this subset contains images for which fine annotations are present.

Given the relatively small subset at our disposal, we test different data augmentation methods in an attempt to increase the diversity of the training set. These include horizontal flipping, blurring and the combination of both. Their corresponding effects on the results can be seen in Table 3.

4.2. Implementation Details

To train the adaptation model, the segmentation network and discriminator are trained jointly. Training was performed for 50 epochs, with a batch size of 4. Images are cropped from their original resolutions to 1024x512 and normalized. We implemented our network using the PyTorch framework and training it using the free resources offered by the Google Colab service.

Segmentation Network: BiSeNet was trained using the Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9, a weight decay of 10^4, an initial learning rate of $2.5 * 10^{-2}$ decreased by the polynomial decay power 0.9 (as mentioned in [3]). ResNet-18 [9], pretrained on ImageNet [11], was chosen as the backbone of the context path. We preferred it over more performing, but more computationally expensive alternatives, such as ResNet-101, due to hardware limits.

Discriminators: D is trained using the Adam optimizer [12] with a learning rate of $1*10^{-4}$, exponential decay rates set to 0.9 and 0.99. As mentioned before, we tested two discriminators: the Fully Convolutional (D_{FC}) and the Light Weight Discriminator (D_{Light}). D_{FC} consists of 5 convolutional layers with kernel 4x4 and stride 2, where the channel number is {64, 128, 256, 512, 1}, respectively. Except for the last layer, each convolutional layer is followed by a Leaky ReLU with slope 0.2. The last convolutional layer is followed by an up-sample layer that re-scales the output to the original dimensions. D_{Light} follows the same architecture of the D_{FC}, but replaces the convolutional layers with depth-wise convolutions, consisting of a convolution of kernel size 4x4 and stride 2, applied separately on each channels followed by a point-wise convolution with kernel size 1.

In regards to the adaptation task, the λ_{adv} is set to 0.001, as suggested in the original paper [20]. When applying FDA, we tested the performances with different β (0.1, 0.05, 0.01), keeping $\beta \leq 0.15$, as suggested by Yang et al. [22].

Metrics: To asses the semantic segmentation quality of our approach, we rely on accuracy and mIoU. The accuracy is defined as,

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

with TP standing for True Positives, TN for True Negatives, FP for False Positives and FN for False Negatives. In the case of semantic segmentation, accuracy calculated globally across all classes and represents the percent of pixels correctly classified in the image (correct predictions / total predictions).

The disadvantage with this metric is how misleading it can be for an imbalanced dataset, as Cityscapes and GTA5 are (see Table 5 and Figure 1 and 2 in [5]), since it does not distinguish between the numbers of correctly classified pixels of different classes, favoring the majority classes [7].

Therefore, we also report the mIoU [1], mean Intersection over Union, and use this as the main metric to asses semantic segmentation quality. For an individual class, IoU is defined as follows:

$$IoU = \frac{TP}{TP + FP + FN}$$

It is the intersection of the inferred segmentation and the ground truth, divided by the union [6]. The mean IoU, mIoU,

$$mIoU = \frac{\sum_{c \in C} IoU_c}{|C|}$$

is the arithmetic mean of the IoU of all classes. Through this we compensate dataset imbalance and acquire a more accurate performance metric.

4.3. Results

In this section we report the results of our method on the $GTA5 \rightarrow Cityscapes$ benchmark.

Table 3. *Upper Bound:* performances of BiSeNet trained fully supervised with the target dataset varying the type of augmentation applied.

Augmentation	Accuracy(%)	mIoU(%)
None	79.3	47.3
Blur	79.1	47.0
Horizontal Flipping	79.8	49.7
Horizontal Flipping and Blur	79.7	**49.9**

Upper Bound: Table 3 shows the results obtained training BiSeNet using the target dataset Cityscapes (with the corresponding annotations). As previously mentioned, due to the small amount of training data, and to make our model more robust, we employed image augmentation to create modified versions of available input images, thereby increasing the model's ability to generalize. Since the dataset used is Cityscapes, which contains images sampled from a video taken with a car's front-facing camera, the image augmentation methods used need to be consistent with image perturbations that a car could encounter. On the other hand, to not undermine the performances of the model, we used actions that lead to faint variations of the input images. For these reasons, the augmentation techniques chosen are: (1) horizontal flipping, (2) Gaussian Blur and (3) the composition of the two. The augmentation is applied with probability $p = 50\%$ and the parameters used for the Gaussian blur are kernel size 9 and standard deviation σ used for creating the kernel chosen randomly from range $(1, 2)$.

As shown in Table 3, the best results are obtained with the combination of horizontal flipping and Gaussian blur. Therefore we applied these same augmentations in the other steps as well. The established upper bound for the subsequent network, trained on a synthetic dataset and then tested on Cityscapes, is $mIoU = 49.9\%$.

Transfer Knowledge without Domain Adaptation: As explained in section 1, the goal is to use a synthetic dataset to train the network and then to transfer the knowledge in a real-life domain, maintaining high performance. We therefore firstly trained BiSeNet with the configuration described in section 4.2 and only with samples from the GTA5 dataset, testing it directly on Cityscapes without domain adaptation. Due to the domain gap between the two datasets, the performance on Cityscapes drops to a final mIoU of 13.4%, significantly lower with respect to the 49.9% obtained training only with Cityscapes (Table 4). For this reason, we compulsorily need domain adaptation techniques to get a model trained on the source domain to generalize also in the target domain.

Domain Adaptation: Firstly, we compared the quantitative results obtained using the Fully Convolution Discriminator and the Lightweight version. As shown in Table 4, the lightweight discriminator achieves a $mIoU$ of 24.4%, an increase of 0.2% compared to the fully convolutional variant. This validates our strategy, since we are able to obtain comparable results while consuming considerably less resources.

Increasing Performance with FDA: As seen in Table 4, using the lightweight discriminator with FDA a $mIoU$ of 27.4% was achieved, improving performance by 3.0%. We tested three different values of β for FDA and the results obtained only differed by, at most, a $mIoU$ of 0.8%, demonstrating the robustness of FDA with respect to the choice of β , as described in [22]. We also tested if the presence of the Gaussian Blur augmentation would worsen the results, but as seen in Table 4, it actually increases performance by 0.5%.

Despite these improvements, there is still a considerable $mIoU$ difference of 22.5% when we compare our best result in the domain adaptation task to the upper bound, determined by BiSeNet trained and tested on the Cityscapes dataset using ground-truth labels. When we observe the individual $mIoU$ results in Table 4, we see that for certain

Table 4. *IoU of each individual class and final mIoU*. All numbers are in terms of %. ”Fully conv” stands for model trained on the source dataset with domain adaptation using a fully convolutional discriminator. ”LW” stands for model trained on the source dataset with domain adaptation using the lightweight discriminator.

Method	Road	Sidewalk	Building	Wall	Fence	Pole	Light	Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle	mIoU %
Target only	95.5	69.4	84.8	23.2	25.4	32.5	29.1	45.9	86.9	48.2	88.6	56.8	30.2	85.4	15.5	23.0	25.9	27.1	53.3	**49.9**
No domain adaptation	4.7	11.8	47.2	2.2	11.5	6.5	0.8	0.0	67.9	3.6	57.8	13.7	0.0	22.9	2.9	1.1	0.0	0.0	0.0	13.4
Fully Conv	73.9	24.5	67.9	18.4	10.2	16.1	7.4	0.1	77.0	12.3	63.0	22.4	0.1	53.1	8.8	1.3	0.0	2.6	0.0	24.2
LW Conv	65.1	25.8	62.2	13.3	11.0	17.0	7.8	0.2	79.1	21.5	62.8	27.0	1.0	54.5	10.4	3.2	0.0	0.4	0.0	24.4
LW Conv, FDA ($\beta = 0.01$)	73.8	30.6	70.2	23.7	7.0	16.3	5.9	1.4	78.0	28.0	69.6	28.2	1.4	58.4	10.6	5.3	0.0	2.9	0.0	27.0
LW Conv , FDA ($\beta = 0.05$)	72.5	27.9	72.5	16.5	10.7	17.1	6.2	0.6	77.0	24.9	65.7	27.3	8.2	59.2	8.7	2.7	0.0	5.9	0.0	26.6
LW Conv, FDA ($\beta = 0.10$)	78.8	30.6	73.8	18.5	10.4	15.4	2.6	**1.4**	76.2	22.7	65.9	29.8	**3.5**	64.3	11.8	5.7	**0.0**	9.3	**0.0**	**27.4**
LW Conv , FDA ($\beta = 0.10$, no blur)	80.4	33.8	72.8	19.6	6.3	18.5	5.1	1.3	74.1	19.2	67.8	29.0	4.5	61.0	9.6	3.3	0.0	3.4	0.0	26.9

Table 5. Percentage of images containing a specific label in the GTA5 training dataset and the Cityscapes validation dataset.

Label	GTA5 images (%)	CS images (%)
Road	99.4	96.4
Sidewalk	96.2	92.8
Building	100	98.4
Wall	95.8	40.4
Fence	84.8	38
Pole	99.8	98
Light	72.2	56.8
Sign	**54.2**	**94.4**
Vegetation	99	97.2
Terrain	96.6	47.2
Sky	99.8	86.8
Person	93.4	78
Rider	**13.4**	**49.2**
Car	89	94.8
Truck	66.2	17.6
Bus	20	16.4
Train	**3.4**	**4.8**
Motorcycle	15.6	19.2
Bicycle	**0.8**	**68.8**

classes such as ”Train”, ”Bicycle”, and ”Sign”, the metric is close or equal to 0.0%. As seen in Table 5 and mentioned in section 4.2, when inspecting the subsets available to us, the unbalanced class distribution in the datasets and between the datasets becomes very apparent. Some classes are rarely represented in the dataset, e.g. the class ”Train” is found in only 3.4% of the images of the source dataset and in only 4.8% of the images of the target dataset. This leads to a mere 25.9% with the fully supervised model, while all the domain adaptation models achieve an IoU of 0.0%. On the other hand, the classes like ”Sign”, ”Rider” and ”Bicycle” are not as present in the source dataset as they are in the target dataset. The ”Bicycle” class appears in just 0.8% of the training images while in 68.8% of the target images. The IoU drops from 53.3% in the fully supervised model to 0.0% in all the domain adaptation models. We believe these imbalances play an important factor in the 22.5% performance decrease.

In Figure 3, it is possible to visualize the predictions for a sample image from the Cityscapes dataset using the different approaches we tested, such as what BiSeNet predicts when no domain adaptation takes place or what the output is when using the lightweight discriminator with FDA.

5. Conclusion

In this paper, we have combined adversarial learning using a lightweight real-time model, with a a simple Fourier Transformation, to narrow the domain gap and increase semantic segmentation performance. We evaluated our approach on the synthetic-to-real GTA5 to Cityscapes benchmark, achieving a mIoU of 27.4%. Possible options for future work are to validate this approach using the complete datasets and an higher number of training epochs, use a different segmentation model or attempt a further combination with another domain adaptation method.

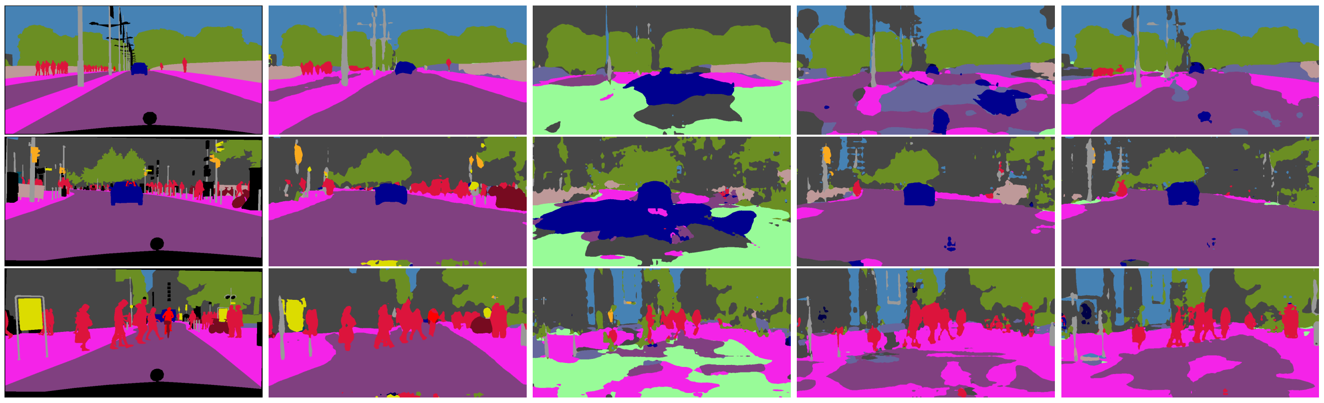
Acknowledgement

Our code has benefited from the official implementation code of AdaptSegNet (https://github.com/wasidennis/AdaptSegNet) and FDA (https://github.com/YanchaoYang/FDA). Also, the model of BiSeNet has been borrowed by the pytorch implementation of https://github.com/ooooverflow/BiSeNet. Furthermore, we would like to acknowledge the support and guidance of our tutor Antonio Tavera.

References

[1] tf.keras.metrics.MeanIoU. 6

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pat-*



((a)) GT ((b)) Only Target ((c)) No DA ((d)) LW no FDA ((e)) LW FDA

🟪 road	⬛ building	🟫 fence	🟧 light	🟩 vegetation	🟦 sky	🟥 rider	⬛ truck	⬛ train	🟥 bicycle
🟪 sidewalk	🟪 wall	⬜ pole	🟨 sign	🟩 terrain	🟥 person	🟦 car	⬛ bus	🟦 motocycle	⬛ others

Figure 3. *Visual Comparison.* (a) Ground Truth of the input image from Cityscapes. Then segmentation produced by: (b) the model trained only on Cityscapes, (c) the model trained on GTA without domain adaptation, (d) the model trained on GTA with domain adaptation, without FDA and finally (e) is the output of the model trained on GTA with domain adaptation, with FDA (Light-Weight discriminator, $\beta = 0.10$). Down is reported the legend of the colors for the annotations.

tern Analysis and Machine Intelligence, 39(12):2481–2495, 12 2017. 2

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 4 2018. 5

[4] Francois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017-Janua, pages 1800–1807. IEEE, 7 2017. 3

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223. IEEE, 6 2016. 5, 6

[6] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136, 1 2015. 6

[7] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 7 2012. 6

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. In *Advances in Computer Vision and Pattern Recognition*, volume 17, pages 189–209. 2017. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:770–778, 12 2015. 5

[10] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 4 2017. 4

[11] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 6 2009. 5

[12] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 12 2014. 5

[13] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional Learning for Domain Adaptation of Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:6929–6938, 4 2019. 2

[14] Wei Liu, Andrew Rabinovich, and Alexander C. Berg. ParseNet: Looking Wider to See Better. pages 1–11, 6 2015. 2

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 847–856, 11 2014. 2

[16] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. ENet: A Deep Neural Network Architecture

for Real-Time Semantic Segmentation. (4):239–244, 6 2016. 2

[17] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9906 LNCS:102–118, 8 2016. 2, 5

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *IEEE Access*, 9:16591–16603, 5 2015. 2

[19] Ilya Tolstikhin, Bharath K. Sriperumbudur, and Bernhard Schölkopf. Minimax estimation of maximum mean discrepancy with radial kernels. *Advances in Neural Information Processing Systems*, (Nips):1938–1946, 2016. 2

[20] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to Adapt Structured Output Space for Semantic Segmentation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7472–7481. IEEE, 6 2018. 2, 3, 4, 5

[21] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Realtime Semantic Image Segmentation via Spatial Sparsity. 12 2017. 2

[22] Yanchao Yang and Stefano Soatto. FDA: Fourier Domain Adaptation for Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4084–4094, 4 2020. 1, 2, 3, 4, 5, 6

[23] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. BiSeNet: Bilateral Segmentation Network for Real-Time Semantic Segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11217 LNCS, pages 334–349. 2018. 1, 2, 3

[24] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated Residual Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017-Janua, pages 636–644. IEEE, 7 2017. 2

[25] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid Scene Parsing Network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2017-Janua, pages 6230–6239. IEEE, 7 2017. 2