Domain Adaptive Visual Object Detection

Leonardo Di Giovanna
Politecnico di Torino
s270195@studenti.polito.it

Giuseppe Moscarelli
Politecnico di Torino
s270240@studenti.polito.it

Gaetano Riccardo Ricotta
Politecnico di Torino
s276937@studenti.polito.it

Andrea Vara
Politecnico di Torino
s270327@studenti.polito.it

Abstract

In the object detection task a performance decay is registered when we want to identify objects belonging to a different target domain than the source domain on which the model has been trained. In this paper we try to deal with the Domain Adaptation problem. At first we replicate and analyze a domain transferring technique proposed in literature: this technique exploits a CycleGAN implementation for unpaired image-to-image translation. Later, we propose an alternative solution based on arbitrary style-transferring with AdaIN: it involves the online transferring of the domain target images styles to the source images. The AdaIN approach allows us to reach an high stylization variability due to the possibility to extract a different style from each image: this high variability is the main advantage of AdaIN approach w.r.t. the CycleGAN solution. Another advantage of our solution w.r.t. CycleGAN is the fact that there is no need to use a model specifically inferred from our domain and target images: it is possible to use a pre-trained architecture without losing results quality.

1. Introduction

The object detection task is based on the identification of certain objects belonging to a predefined set of known classes. In literature, different solutions were proposed for the above-mentioned task: among these, [10, 8, 9] are the most well-known. The detectors analyzed in these papers need a huge amount of instance-level annotated images during the training phase. With *instance-level annotated image*, we mean an image on which each class instance is associated with a bounding box and a label.

The difficult in finding a huge amount of images, combined with the fact that these images have to be instance-level annotated, makes the object detection solutions inapplicable in the majority of the cases. In contexts where it is not possible to find sufficient training data for training a detector on a particular domain, it may be reasonable to think about using a model previously trained on a different domain: however, it was shown that, the good performances, achieved applying the model on data belonging to the same data distribution from which the training sample are drew, decay when you try to apply the above strategy.

In order to overcome the performance gap due to the domain shift, different Domain Adaptation solutions were proposed. Domain Adaptation is a transfer learning subcategory, whose goal is to fit a model trained on specific source domain on a specific target domain. In order to achieve sufficiently accurate results, in this paper we focus on domain adaptation settings on which the possession of samples belonging to the target domain is required, although in small numbers and without instance-level annotations: in [6] some solutions of this type are analyzed. In our paper, we propose an alternative solution based on online style transferring: this alternative method, based on an AdaIN layer [4], allows us to perform arbitrary transferring of the target domain images styles, in an online fashion during the training, over the source domain images. Besides our solution, but also in order to prove its effectiveness, we propose a comparison among 3 models: the model trained on the data belonging to the source domain, the model obtained replicating the technique exposed in [6] and the the model obtained implementing our solution. As the experiments points out, our solution allows us to achieve comparable if not better results than those obtained replicating the domain transferring technique analyzed in [6].

2. Related Work

The paper [6] analyzes Domain Adaptation techniques from a source domain to different target domains. It proposes incremental improvements on a model trained on data belonging to the source domain. In particular, the authors described a novel task called *Cross-Domain Weakly Super-*

vised Object Detection: the followed strategy consists of two step of adaptation on a baseline model. The strategy could be described as follows:

1. the authors started from an SSD [8] implementation and performed the training using images belonging to the source domain; the model obtained in this phase was called *baseline SSD model*
2. in the next step, a CycleGAN [12] implementation was exploited by the authors in order to obtain a mapping function from the source domain to the target domain; the obtained function made possible the transformation of the source domain images into target domain images; inside the paper, this process is referenced as *domain transferring* or with the *DT* acronym
3. the images obtained in the previous step were exploited for the baseline SSD model finetuning
4. in the final step, the authors obtained instance-level annotations for domain target images through a procedure called *pseudo-labelling* (PL)
5. the obtained instance-level annotated images were used for a further finetuning step

The performances of the model obtained after each finetuning step were compared to the ones obtained applying different supervised and unsupervised techniques. Since in this paper we are interested in analyzing the settings on which there are no instance-level annotations, we focus on analyzing the performances obtained by the model finetuned with the data coming from the DT step.

3. Our work

3.1. Baseline detector

Initially, our work focused on building a baseline model: this was achieved by training an SSD [8] implementation using the source domain images.

3.2. Domain-transferring with CycleGAN

Among the techniques described in [6], we chose to replicate the DT technique: in order to accomplish this, an unpaired image-to-image translation method called CycleGAN [12] is employed. The CycleGAN objective is to learn the mapping functions between two image domains X and Y that do not need to be related in any way. Formally, a mapping $G : X \rightarrow Y$ and an inverse mapping $F : Y \rightarrow X$ are jointly learned. G and F should be inverses of each other: this structural assumption is applied by training both the mapping G and F simultaneously, and adding a cycle consistency loss that encourages $F(G(x)) \approx x$ and $G(F(y)) \approx y$, where $x \in X$ and $y \in Y$. The cycle consistency loss captures the intuition that if we translate from one domain to the other and back again we should arrive at where we started (Figure 1). In order to match the distribution of generated images to the data distribution in the target domain, also adversarial losses are applied to both mapping functions. The adversarial losses are used to learn mappings such that the translated images cannot be distinguished from images belonging to the arrival domains. The combination of the cycle consistency loss and the two adversarial losses defines the objective of CycleGAN. We train CycleGAN to learn the mapping functions between the source domain, X, and the target domain, Y. Once the mapping functions are obtained, we use F in order to convert the source domain images to target domain images. These domain transferred images, along with their original annotations, are used for the SSD finetuning.

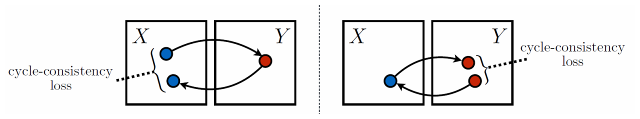


Figure 1: forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

3.3. Online style-transferring using AdaIN

The technique based on domain-transferring has several similarities with the one proposed in this paper: only a limited amount of samples is needed and there is no need of instance-level annotations for the domain target images. Specifically, our proposed solution plans to execute a style transferring from the target domain images to the images belonging to the source domain: this transferring was executed online during the finetuning of the baseline model. The style transferring was achieved by exploiting a style transfer network implementation that uses an AdaIN layer [4].

The style transfer network, whose architecture is shown in Figure 2, is composed of an encoder, an AdaIN layer and a decoder. The encoder is a truncated pre-trained VGG-19 network: it is used to encode both content and style images. The AdaIN layer simply adjusts the mean and variance of the encoded content image to match those of the encoded style image. The decoder network inverts back the normalized content representation, obtained as output of the AdaIN layer, to the starting content image space.

Formally, we can describe the encoder as a function f which takes as input an image x and produces a representation $f(x)$ in a feature space. The AdaIN layer takes as input two samples $x, y \in R^{C \times H \times W}$ and aligns the per-channel means and variances of the former to the per-channel means and variances of the latter:

$$AdaIN(x,y) = \sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \mu(y) \tag{1}$$

In the above equation, $\mu, \sigma \in R^C$. Given a content image c and a style image s, their encoded representations are actually feature maps that can be denoted as $f(c)$ and $f(s)$. Considering $f(c)$ and $f(s)$ as input of the AdaIN layer, the output t is the following:

$$t = AdaIN(f(c), f(s)) \tag{2}$$

Modelling the decoder as a function g, we can express the style transfer network output as:

$$T(c,s) = g(t) \tag{3}$$

There is no need to train the decoder in order to fit our specific problem since, as [4] points out, its reconstruction ability is not affected by the fact that specific styles are seen during training. For this reason, we decided to use a pretrained model also for the decoder part.

An important aspect of this style transfer network is that is possible to control the stylization grade tuning the α:

$$T(c,s,\alpha) = g((1-\alpha)f(c) + \alpha t) \tag{4}$$

The α parameter is introduced only in the test phase. It denotes the grade of stylization that has to be applied over the content image: when α tends to 0 the decoder tends to reconstruct the original image; when α tends to 1, the decoder tends to reconstruct the normalized image.

The style transfer network in test setting is used as building block of the detector architecture: we exploited it to stylize source images right before sending them to the detector during the finetuning phase, in an online fashion. The overall architecture for the finetuning is shown in Figure 3. During finetuning, at each iteration i, a content batch C_i, containing source domain images, and a style batch S_i, containing target domain images, are drawn: the two batches have the same dimension N. We introduce an hyperparameter called *transfer ratio* (*TR*) in order to tune the amount of content batch images that has to be transformed by the style transfer network before feeding the detector. The *transfer ratio* hyperparameter expresses the probability that the style-transferring is applied over a content batch image. Considering a (content, style) images pair $(C_{i,j}, S_{i,j})$ with $j \in \{0, 1, ..., N\}$, the detector network input becomes a random variable $X \in \{T(c,s,\alpha), c\}$ described by the following mass distribution function:

$$p(x, TR) = TR^{\delta(x, T(C_{i,j}, S_{i,j}, \alpha))}(1-TR)^{\delta(x, C_{i,j})} \tag{5}$$

Choosing different values for both the *transfer ratio* and the α hyperparameters allowed us to perform different experiments: the obtained results were compared to the ones obtained by finetuning the *baseline SSD model* with the images obtained through the *DT* step.

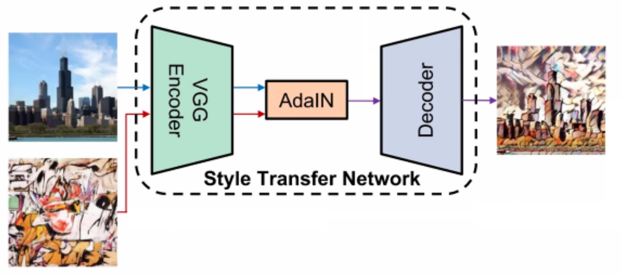


Figure 2: AdaIN architecture

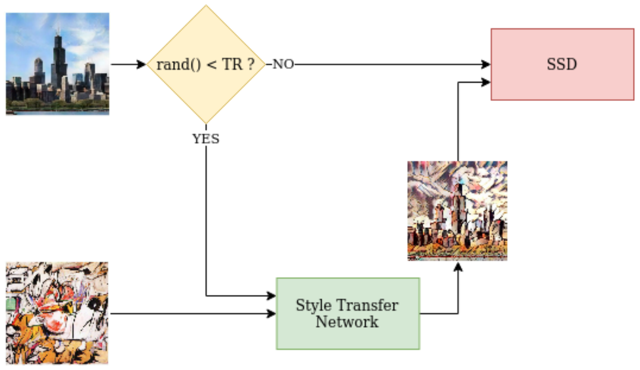


Figure 3: Overall architecture

4. Datasets

The following sections detail the datasets used in the different phases. Some images examples drawn from these datasets are shown in Figure 4.

4.1. Pascal VOC2007 e VOC2012

Pascal VOC2007 e VOC2012 are two datasets containing instance-level annotated natural images. These images contains objects that belongs to 20 different classes. In this paper, the images belonging to these datasets define the source domain. In particular, the datasets splits used are the followings:

- the VOC 2007 trainval split containing 5011 images
- the VOC 2012 trainval split containing 11540 images
- the VOC 2012 trainval split containing 11540 images

The two trainval splits were used during the training/finetuning phases of the *baseline SSD model* and during *CycleGAN model* training. The test split was used for the *baseline SSD model* testing.

Figure 4: On the left, some samples belonging to VOC 2007/12 are shown. On the right, some samples belonging to ClipArt1k are shown.

4.2. ClipArt1k

ClipArt1k is a dataset that was built in order to accomplish the work proposed in [6]. This dataset is composed by 1000 instance-level annotated images containing objects that belong to the same 20 classes present in the source domain images. In this paper, the images belonging to this dataset define the target domain.

The dataset is composed of two splits: train and test. In turn, each split is composed of 500 images. The train split was used in order to perform the *CycleGAN model* training and the online style transferring of our proposed solution. The test split was used in order to perform:

- the *baseline SSD model* test
- the test of the *baseline SSD model* finetuned using the domain-transferred images
- the test of the model obtained performing the finetuning of the *baseline SSD model* with our proposed solution

5. Implementations

5.1. Baseline SSD model

In order to perform the *baseline SSD model* training, we use the Pascal VOC 2007 and 2012 trainval splits. We started from the PyTorch implementation [7] and we performed some changes. These latter changes brought to the implementation [1]. As base architecture, SSD300 with VGG16 as a core network was used. For the *baseline SSD model* training, the specifications provided in [8] with some variations were followed, as specified in [6]. Some of the used settings are the followings:

- the input images were resized to 300×300
- The IoU threshold for NMS was set to 0.45
- the confidence threshold for discarding low confidence detections was set to 0.01
- learning rate was set to 10^{-3}
- momentum was set to 0.9
- weight decay was set to 5×10^{-4}
- batch size was set to 32
- iterations number was set to 120000

5.2. CycleGAN training

CycleGAN was trained in order to obtain a mapping function between the source domain images and the target domain images. The training was executed using the merge between the Pascal VOC 2007 and 2012 trainval splits as first dataset. The ClipArt1k train split was used as second dataset.

The training was performed for 20 epochs in total: for the first 10 epochs a learning rate of 1.0×10^{-5} was used; for the following 10 epochs, a linear decay rate to 0 was set. Following the specification proposed in [12], we set the value of λ to 10 and the batch size to 1. The original implementation of CycleGAN, available at [11], was exploited as a starting point, but we performed on it some modifications available at [2].

5.3. Baseline SSD model finetuning using domain-transferred images

Using the CycleGAN model obtained in the previous step, we performed a domain transferring of the Pascal VOC 2007 and 2012 trainval splits to the ClipArt1k domain. The obtained images were used in order to perform the finetuning step of the *baseline SSD model*. Some examples of domain-transferred images are shown in Figure 5. The finetuning was performed setting the learning rate to 1.0×10^{-5} and the iterations number to 520 (equivalent to about an epoch, considering a batch size of 32).

5.4. Baseline SSD model finetuning using style-transferred images

The results obtained in the previous step were compared to the ones obtained in this last step. Specifically, in this step we performed the finetuning of the *baseline SSD model* using the original Pascal VOC 2007 and 2012 trainval splits

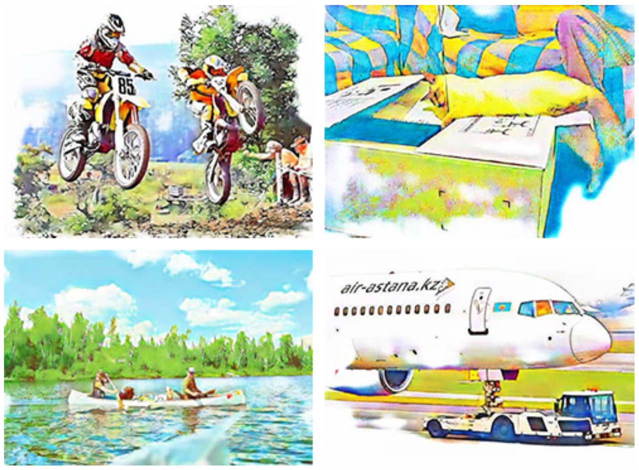


Figure 5: In this image, some results of the CycleGAN application over the Pascal VOC 2007/12 dataset are shown.

images: in this setting, we performed an online style transferring exploiting the ClipArt1k train split images. In order to perform the style transferring in an online setting, we started from the implementation [5] of the style transfer network described in [4]. The latter implementation was readapted [3] in order to allows the online style transferring during the *baseline SSD model* finetuning. The target domain images were resized to 512×512. During the style transferring, we decided to enable the source domain images color preservation on the style transfer network. Different configurations for the $transferratio$ and α hyperparameters were explored.

6. Results

In this section we provide the results obtained performing the different steps. The results were derived by testing the models using the ClipArt1k test split.

6.1. Baseline SSD model

The model obtained in this phase achieved results that can be viewed observing the entry *Our Baseline SSD* in Table 1. The first entry of the latter Table shows the results obtained by the *baseline SSD model* in [6]. As you can see, our baseline, with an mAP of 25.4, achieved results comparable to the solution proposed in the paper. In terms of single classes, we registered a significant performance improvement in bike, cat, sofa, train instances classification. On the contrary, the performances achieved by the *Author's Baseline SSD* model of [6] were better over bus, car, cow, person and sheep instances.

6.2. Baseline SSD model finetuning using domain-transferred images

By performing test over the *baseline SSD model* finetuned using the domain transferred images, we registered a significant performances decay with respect to the ones achieved in the corresponding step in [6]: specifically, our model reached an mAP value of 30.7, whereas the value reached in the paper is 38.0. In detail, the results are shown in Table 2, on the entry *Our FT* and *Author's FT* respectively. Given the substantial results difference, in order to try to overcome the performance gap, we decided to perform a second finetuning step with an higher iterations number (from 520 to 1100): however, as it is shown on the *Our FT 2* entry of the aforementioned table, we registered only a slight mAP improvement, not yet comparable with the *Author FT* mAP. It would have been possible to try to further increase the iterations number but we preferred to go no further, since the results in [6] were obtained stopping the finetuning after a single epoch.

The performance gap may be due to the different strategies followed for the CycleGAN training. In fact, as we can see in the Figure 6, the images built using the CycleGAN model [6] are closer to the target domain images than the ones produced by our CycleGAN model.

6.3. Baseline SSD model finetuning using style-transferred images

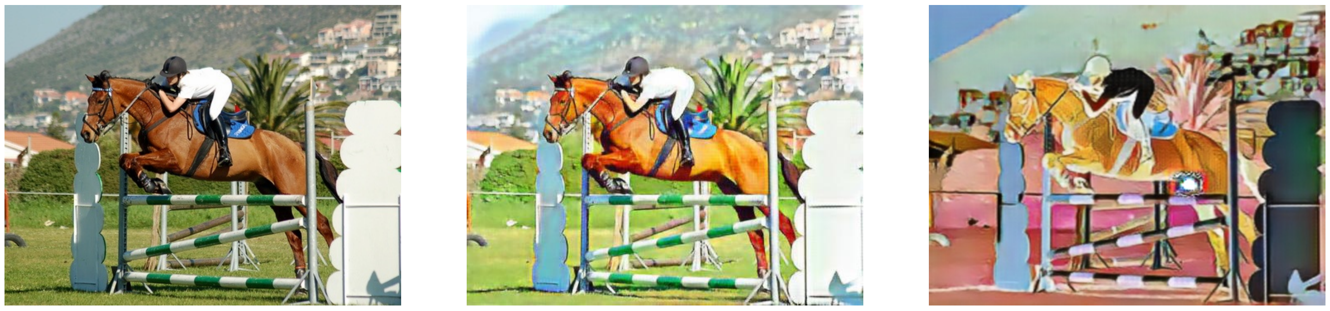
The results obtained in this step are summed up in the Table 3. The *baseline SSD model* finetuning was performed trying different combinations of the iterations number ($\#It$), transfer ratio (TR) and α parameters. We recall that, the transfer ratio hyperparameter expresses the probability that the online style transferring over a specific source sample is applied: so, it can assume values between 0.0 and 1.0. The α parameter defines the degree of sample stylization: it can assume a value between 0.0 and 1.0, where 0.0 tells the network to reconstruct to original sample without any application of the new style, whereas 1.0 tells the network to apply the maximum of stylization, as described in [4].

6.3.1 Notation

In the following discussion, whenever we talks about a specific configuration, this will be denoted by the triple $(\#It, TR, \alpha)$; whenever a configuration is represented by a parameters couple, the iterations number will be assumed to be 520.

6.3.2 Results details

We chose to use the same iterations number (520) for both the finetuning performed after the domain transferring step



(a) (b) (c)

Figure 6: The image (a) is a sample extracted from Pascal VOC 2007. The image (b) shows the result of the application of our CycleGAN model over the image (a). The image (c) shows the result of the application of the CycleGAN model used in [6] over the image (a).

Model	aereo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Author's Baseline SSD	19.8	49.5	20.1	23.0	11.3	38.6	34.2	2.5	39.1	21.6	27.3	10.8	32.5	54.1	45.3	31.2	19.0	19.5	19.1	17.9	26.8
Our Baseline SSD	18.9	56.6	19.2	19.9	10.1	26.5	27.0	10.5	42.7	10.3	24.9	14.5	30.9	48.5	35.0	30.4	9.1	24.2	26.5	22.5	25.4

Table 1: The table shows the results of the tests applied over the two Author's Baseline SSD and Our Baseline SSD model. The results obtained by the Author's Baseline SSD model are extracted from [6].

Implementation	aereo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Author's FT	23.3	60.1	24.9	41.5	26.4	53.0	44.0	4.1	45.3	51.5	39.5	11.6	40.4	62.2	61.1	37.1	20.9	39.6	38.4	36.0	38.0
Our FT	20.2	48.9	20.4	23.4	24.5	39.0	39.7	3.9	45.2	20.4	38.2	15.0	33.8	57.7	49.3	34.5	13.0	28.5	30.8	28.6	30.7
Our FT 2	22.0	54.8	21.0	25.0	23.9	40.6	39.9	3.2	46.2	24.0	35.7	12.7	35.0	56.1	51.8	36.8	15.8	30.3	27.8	30.0	31.6

Table 2: In this table the results of the tests performed over the models obtained by finetuning the baseline SSD model with the domain-transferred images are shown. The results shown for the model built with the finetuning performed by the authors of [6] are extracted from the same paper. For the two provided implementations, identified by the entry "Our FT" and "Our FT 2", the training has been performed for 520 and 1100 iterations respectively.

Configuration			aereo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
# It	TR	α																					
520	0.5	0.5	24.1	56.8	20.6	21.1	23.0	41.6	39.1	5.2	47.1	15.6	38.2	11.5	32.9	59.6	49.5	32.8	6.6	31.1	37.3	31.6	31.3
520	1.0	0.5	24.2	54.9	21.3	25.2	24.7	48.5	41.6	3.8	46.5	21.8	39.5	12.8	32.2	61.6	50.4	36.2	6.3	28.8	37.1	35.8	32.7
520	0.5	1.0	24.0	56.7	22.2	27.6	28.0	53.4	42.1	2.9	49.2	38.3	43.4	15.0	34.4	70.5	54.7	43.0	8.1	34.3	46.8	41.1	36.8
520	1.0	1.0	24.5	56.8	23.5	33.5	26.0	62.9	45.7	4.2	48.5	40.6	41.5	21.2	31.0	69.9	55.1	45.6	9.8	36.0	47.1	40.9	38.2
1040	1.0	1.0	25.3	59.7	23.8	34.0	25.4	65.0	45.8	4.6	46.8	39.7	40.6	19.8	29.1	73.6	57.1	47.6	8.8	37.8	46.9	39.5	38.5

Table 3: In this table the results of the tests performed over the models obtained by performing the finetuning of our baseline SSD model with the VOC online style-transferred images.

and the one performed together with the online style transferring: this choice was made in order to compare the two techniques. All the tried combinations led to comparable if not better results than the ones obtained with our implementations in the previous steps. In particular, the following considerations can be made:

- the $(0.5, 0.5)$ configuration achieved an mAP value equal to 31.3, already higher than 30.7 achieved by *Our FT* model (Table 2) and comparable to 31.6 achieved by *Our FT 2* model (Table 2)
- the α parameter plays a decisive role in the quality of performance of the produced model; starting by the observation of the results obtained from the tests on the model produced with the $(0.5, 0.5)$ configuration, changing the transfer ratio parameter value from 0.5 to 1.0 resulted in an mAP increase of 1.4 while changing the α parameter value from 0.5 to 1.0 resulted in an mAP increase of 5.5

- the $(1.0, 1.0)$ configuration achieved the best results among all the other configurations with 520 iterations; the resulting mAP value, equal to 38.2, slightly overcame the mAP value obtained by the author's model [6] and obtained by finetuning their baseline SSD model with the CycleGAN output images; this author's finetuned model achieved the performance shown in Table 2 in the entry *Author's FT*
- the $(1040, 1.0, 1.0)$ configuration achieved the best results with an mAP value equal to 38.5; however, these latter results do not differ much from the results obtained with the $(520, 1.0, 1.0)$ configuration; therefore, we decided not to go any further with the number of iterations.

7. Conclusion

In this paper, we performed the training of an SSD model upon the images belonging to the source domain. We followed two different strategies in order to fit the model on the target domain distribution: the first strategy, based on the work described in [12], was already exploited in literature as domain adaptation technique in [6]; the second strategy, based on [4], allowed us to define a new unsupervised domain adaptation framework exploiting online style transferring. This new framework is based on the possession of few samples belonging to the target domain and it does not make the possession of instance-level annotated images a prerequisite. As we can see, the online style transferring-based domain adaptation allowed us to achieve comparable if not better results than the ones obtained using the domain transferring technique.

References

[1] L. Di Giovanna, G. Moscarelli, G. R. Ricotta, and A. Vara. High quality, fast, modular reference implementation of SSD in PyTorch. `https://github.com/ekoops/SSD`, 2021.

[2] L. Di Giovanna, G. Moscarelli, G. R. Ricotta, and A. Vara. Image-to-image translation in Pytorch. `https://github.com/ekoops/pytorch-CycleGAN-and-pix2pix`, 2021.

[3] L. Di Giovanna, G. Moscarelli, G. R. Ricotta, and A. Vara. Unofficial pytorch implementation of 'Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization'. `https://github.com/ekoops/pytorch-AdaIN`, 2021.

[4] X. Huang and S. Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. *arXiv e-prints*, page arXiv:1703.06868, Mar. 2017.

[5] N. Inoue. Unofficial pytorch implementation of 'Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization'. `https://github.com/naoto0804/pytorch-AdaIN`, 2018.

[6] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa. Cross-Domain Weakly-Supervised Object Detection through Progressive Domain Adaptation. *arXiv e-prints*, page arXiv:1803.11365, Mar. 2018.

[7] C. Li. High quality, fast, modular reference implementation of SSD in PyTorch. `https://github.com/lufficc/SSD`, 2018.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. *arXiv e-prints*, page arXiv:1512.02325, Dec. 2015.

[9] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. *arXiv e-prints*, page arXiv:1612.08242, Dec. 2016.

[10] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv e-prints*, page arXiv:1506.01497, June 2015.

[11] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Image-to-image translation in Pytorch. `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`, 2017.

[12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv e-prints*, page arXiv:1703.10593, Mar. 2017.