Domain Adaptive Visual Object Detection

Fereshteh Feizabadi
s274475@studenti.polito.it

Mahtab Niknahad
s273284@studenti.polito.it

Uditi Ojha
s270049@studenti.polito.it

Andrea Zappavigna
s278042@studenti.polito.it

Abstract

Is it possible to perform visual object detection in a variety of unlabeled image domains? The goal of this project is to study the visual object detection task and understand what it takes to transfer task knowledge from a source domain with instance-level annotation to an unlabeled target one. Even though state-of-the-art deep object detectors have become extremely precise and faster than ever, all of them record a decrease in the performance in domain shifted scenarios: that is when the target visual domain is different from the training one. In fact, this is still a novel task, and in our work we start from the results of previous related work and we progressively take steps to achieve our goal. Starting from a fully supervised single-stage object detector, which is pre-trained on the source domain, we propose two different domain adaptation strategies by fine-tuning the detector model on different artificially generated samples. In practice, we propose two different pixel-level domain adaptation strategies that will allow us to translate images from the source to the target visual domain. At the end, we test our model to achieve an improvement of approximately 6 to 14 percentage in terms of mean average precision(mAP) when compared to the baseline model, and approximately 14 percentage behind the ideal case. Code is available at https://github.com/zappavignandrea/ssd

1. Introduction

Object detection is a fundamental task which deals with the identification of specific objects and localization of object instances within the image. It is improving rapidly due to the use of Convolutions Neural Networks (CNN's). The detectors which usually have the best performance are the fully supervised ones (R-CNN, Fast R-CNN[4], Faster R-CNN[12], YOLO[11]). These detectors normally learn from images with instance-level annotations, such annotations includes a label (the object class of that instance) and a bounding box (the location of the instance).

Although this kind of object detection has achieved high performance in a natural image domain it is simply unfeasible to prepare the annotations for a large number of images in different domains, it is time consuming and costly too.

The objective is to tackle cross-domain object detection through domain adaptation: in this work, we are working to improve the performance of visual object detection task in different unlabeled domains by applying the technique of pixel level adaption, which is essentially a shift in representation.

In order to better understand and study key components of domain adaptive visual object detection, we first define the tasks as follows: We start by using a fully supervised object detector, in our case, the Single Shot MultiBox Detector [2], trained on images with instance-level annotation in the source domain. The goal is to detect images as accurately as possible in the target domain by using only instance-level annotations and images from the source domain. At this point, the easiest solution would be to fine-tune the FSD (fully supervised detectors) model on images in the target domain, but we do not have access to instance-level annotations in the target domain. Instead, we present two strategies to automatically generate images by performing domain transfer (DT), this is used to generate images that look like those in the target domain from images in the source domain accompanying instance-level annotations. In this way, we effectively use information from both domains to improve the model performance.

The first example of style transfer is achieved by image-to-image translation methods from unpaired examples such as Cycle-Consistent Adversarial Networks or CycleGANs [8], following the reference paper[10]. Using this method , we can generate the transferred images offline.

After that, we propose a different pixel-level domain adaptation strategy using Adaptive Instance Normalization or AdaIN [6] which is a method that allows to perform arbitrary style transfer in real-time. Using this method, we can generate the domain transferred images online.

In our work we first try to replicate the performance of

baseline and the domain transfer from the reference paper [10], we analyze and show the performances of our baseline and domain transfer methods keeping in mind the different training configuration. Throughout the paper we call **DT1** and **DT2** to refer to the two domain adaptation methods.

2. Related works

Our experiments are based on related works, that consist of previous attempts to interface with domain adaptation problem. In this section, we describe the most relevant ones for this work.

2.1. Visual Object Detection

Object detection has been around for a long time and is now an important research topic in computer vision. The development of convolutions neural networks has greatly advanced the performance of object detection. CNN-based detectors can be mainly divided into two categories: one-stage detectors and two-stage detectors. Although one-stage detectors such as YOLO [11] and SSD [2] have notably higher efficiency and have become popular paradigms over time, two-stage detectors like Fast R-CNN [4], Faster R-CNN [12] and Mask R-CNN [5] are still widely adopted for their higher precision. Faster R-CNN [5] is a classical two-stage object detector and is commonly used as a baseline for domain adaptation. SSD[2] is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature re-sampling stages and encapsulates all computation in a single network, this pipeline setting also means that it has real-time detection ability. This makes SSD[2] easy to train and straightforward to integrate into systems that require a detection component. In this work, we choose SSD300[2] as the baseline detector in all our experiments for its efficiency and scalability.

2.2. Unsupervised Domain Adaption

Unsupervised domain adaptation (UDA) is a task used for learning domain invariant models, where pairs of an image and annotation are available in the source domain while only images are available in the target domain. Previous works for UDA in image classification is mostly distribution-matching-based, in which features extracted from two domains are made to closely resemble each other using the maximum mean discrepancy (MMD)[1] or a domain classifier network. Although current distribution-matching-based methods are applicable, it is primarily challenging to fully align the distribution for tasks that require structured outputs, such as object detection. This is because it is essential to keep the spatial information in the feature map. Our framework employs style transfer and fine tuning it to avoid this problem.

2.3. Domain Adaption Methods

In the past few years there has been many research in domain adaptation for object detection. There are many approaches used for domain adaption, here we describe few of them.

2.3.1 Adaptation via self supervision

Self-supervised learning has gained popularity because of its ability to avoid the cost of annotating a large scale datasets. There are number of papers which proposes to use self supervised tasks in order to bridge the domain gap in cross domain analysis [7][3]. A self supervised task can, in fact, be applied in the same way on supervised and unsupervised images therefore allowing the network to learn something also from unsupervised data. This methods usually involves two tasks using the source domain(labeled) and the target domain(unlabeled), there is joint training between this two tasks. This joint training lead the network to extract useful features from the unsupervised data, thus learning something about the target domain too.

2.3.2 Adaptation via adversarial feature alignment

Adversarial domain alignment is one of the most popular domain adaptation technique which uses deep neural network. The most effective way to achieve domain transfer is based on making predictions on features that cannot discriminate between the source domain(labeled) and the target domain(unlabeled). The focus is on learning features that combine discriminativeness and domain invariance. There are various methods such as Domain Adversarial training of neural network [7], Domain generalization by solving jigsaw puzzles [8], Self-supervised domain adaptation for computer vision [11] and various others.

2.3.3 Pixel level adaptation strategy

There are various pixel level domain adaptation techniques. One such techniques is, unpaired Image-to-Image translation using Cycle-Consistent Adversarial Networks known as CycleGAN [8]. CycleGAN[8] uses a cycle-consistency loss to enable training without the need for the paired data. In other words, it translates from one domain to other without the need for one to one mapping between the source and the target domain. On the other hand , AdaIN [6] is a method that allows to perform arbitrary style transfer in real time, it can extract the style from any style image and apply it over a content image. The advantage of AdaIN[6] w.r.t. CycleGANs[8] is in the fact that it extracts a different style from every image. As a result you can obtain greater variability in the translated dataset. The main disadvantage is that it works on the style level only, without performing

semantic translations. In this work, we have used both these techniques to implement domain adaptation.

3. Dataset

As explained before, our objective is to replicate the results achieved in the reference paper [10], for this reason the datasets we used are the same. For source domain, the natural images in PASCAL VOC [9] were used, overall including 20 classes. In particular, we use VOC2007 and VOC2012 trainval splits for a grand total of 16551 images. Clipart1k[10], that was first introduced in the target paper, were used for images in the target domain. It includes the same 20 classes and 1000 images and 3165 instance-level annotations.

4. Methods

In this section we describe the methodological side of our work, from the first to the last step. Section 4.1 deals with our first method, setting a baseline for performance. Section 4.2 discuss domain transfer, further fine tuning the baseline. Section 4.3 explains implementation of our domain adaptation choice which is AdaIN[6].

4.1. SSD Model

The SSD[2] approach is based on a feed-forward convolution's network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detection. The early network layers are based on a standard architecture used for high quality image classification (truncated before any classification layers), which is the base network, then there are some auxiliary structure to produce detection. Figure 1 shows the a comparison between two single shot object detection models SSD[10] and YOLO[11].

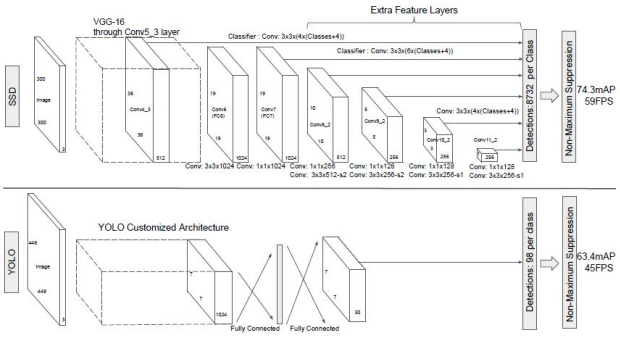
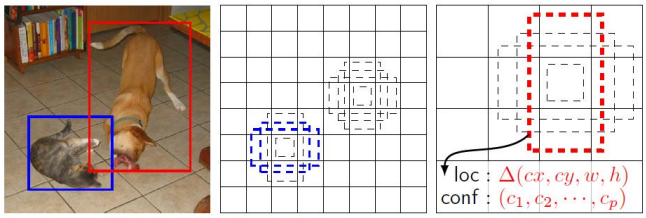


Figure 1. A comparison between two single shot object detection models: SSD[10] with a 300*300 input size significantly outperforms its 448*448 YOLO [11] counterpart in accuracy on VOC2007[9] test while also improving the speed.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Figure 2. Single Shot MultiBox Detector framework.

4.1.1 SSD Training

The key difference between training SSD[2] and training a typical detector that uses region proposals, is that ground truth information needs to be assigned to specific outputs in the fixed set of detector outputs.Once this assignment is determined, the loss function and back propagation are applied end-to-end. Training also involves choosing the set of default boxes and scales for detection as well as the hard negative mining and data augmentation strategies. Figure 2 shows the SSD[2] framework.

4.1.2 Training objective

The SSD[2] training objective is derived from the MultiBox objective, but is extended to handle multiple object categories.

$$x_{i\,j}^{p} = \{1, 0\}$$

is an indicator for matching the i-th default box to the j-th ground truth box of category p. In the matching strategy above, we can have

$$\sum_{i} x_{i\,j}^{p} >= 1$$

.The overall objective loss function is a weighted sum of the localization loss (loc) and the confidence loss (conf):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

where N is the number of matched default boxes. If N = 0, then the loss is 0. The localization loss is a Smooth L1 loss between the predicted box (l) and the ground truth box (g) parameters.Similar to Faster R-CNN [12], we regress to offsets for the center (cx,cy) of the default bounding box (d) and for its width (w) and height (h). The confidence loss is the softmax loss over multiple classes confidences (c). The weight term alpha is set to 1 by cross validation. We have use SSD300[10] as our baseline FSD in all our experiments.

4.2. CycleGAN: DT1

The goal of the domain transfer is to generate images that look like the target domain from the source domain.

To achieve this goal, we are using an unpaired image-to-image translation method called CycleGAN [8] . With CycleGAN[8], the goal is to learn the mapping functions between two image domains X and Y with unpaired examples. In practice, a mapping G : X $\rightarrow$ Y and an inverse mapping F : Y $\rightarrow$ X are jointly learned using CNN. We train CycleGAN[8] to learn the mapping functions between the source domain and the target domain. Once the mapping functions are trained, we convert images in the source domain that are used in the pre-training and obtain domain-transferred images that accompany instance-level annotations. Using these images, we fine tune our baseline.
The goal is to learn mapping functions between two domains X and Y given training samples

$$\{x_i\}_i^N = 1$$

The overall objective loss function are : adversarial losses for matching the distribution of generated images to the data distribution in the target domain; and cycle consistency losses to prevent the learned mappings G and F from contradicting each other.

Adversarial Loss:

$$L_{GAN}(G, D_Y, X, Y) = E_y \sim_{p\ data}(y)[logD_y(y)]$$

$$+E_x \sim_{p\ data}(x)[log(1 - D_y(G(x))$$

Cycle Consistency Loss:

$$L_{cyc}(G, F) = E_x \sim_{p\ data}(x)[||F(G(x)) - x||1]$$

$$+E_y \sim_{p\ data}(y)[||G(F(y)) - y||1]$$

4.3. AdaIN : DT2

AdaIN [6] is a method that allows to perform arbitrary style transfer in real-time, at the heart of the model is a novel adaptive instance normalization layer that aligns the content of the source images with the style of target images. This layer effectively combines the content input x and the style input y by transferring feature statistics, specifically the channel wise mean and variance, so that the values of x and y match.

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

The model adaptively computes the scaling $\sigma(y)$ and shift $\mu(y)$ across spacial locations. A decoder network then generates the final stylized image by inverting the AdaIN[6] output back to the image space.

In order to use AdaIN[6] as a domain adaptation strategy, it is ideal to apply the translation online so that we can exploit the advantages of the high variability. In practice, we used the translation model to stylize your images right before sending them to the detector.

In our implementation, we decide with probability p if we apply the style transfer each time a source data batch is loaded, that is for every iteration. For each of the 32 content images of each batch, we then select a random target image for the style. In this way we do not really consider any possible style for each source image, but choosing the style randomly at each iteration will still give a good variability while keeping the number of iterations equal.

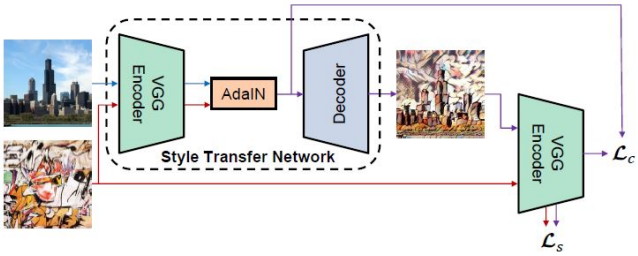


Figure 3. An overview of style transfer algorithm. AdaIN[6] uses a VGG-19 network to encode the content and style images. An AdaIN[6] layer is used to perform style transfer in the feature space. A decoder is then used to invert the output to the content image space.

5. Experiments

In section 5.1 we describe the implementation details of the baseline and its result. In section 5.2 we describe the implementation details of the cycleGAN[8]. In section 5.3 we describe the implementation details and result with the cycleGAN[8] augmented images . In section 5.4 we describe the implementation details and results with the AdaIN[6] transferred images. In section 5.5 we describe the ideal case.

5.1. SSD Baseline

Training: We train our SSD300[2] model in the source domain with instance-level annotations for 60k iteration and different Learning Rate. The input images were resized to 300 × 300 for SSD300 [2]. The IoU threshold, also known as Jaccard Index, for NMS (0.5) and the confidence threshold for discarding low confidence detections (0.01) were employed. We followed the original paper on other hyper-parameters of SSD300[1] unless specified.

Testing: We test our SSD[2] trained model in the target domain, which is a different domain with respect to the source domain that our model was trained on. The result of the tests is shown in Table 1.

5.2. CycleGAN

We trained CycleGAN[8] with a learning rate of 1e-5 for the first 10 epochs and a linear decaying rate to 0 over the next 10 epochs.

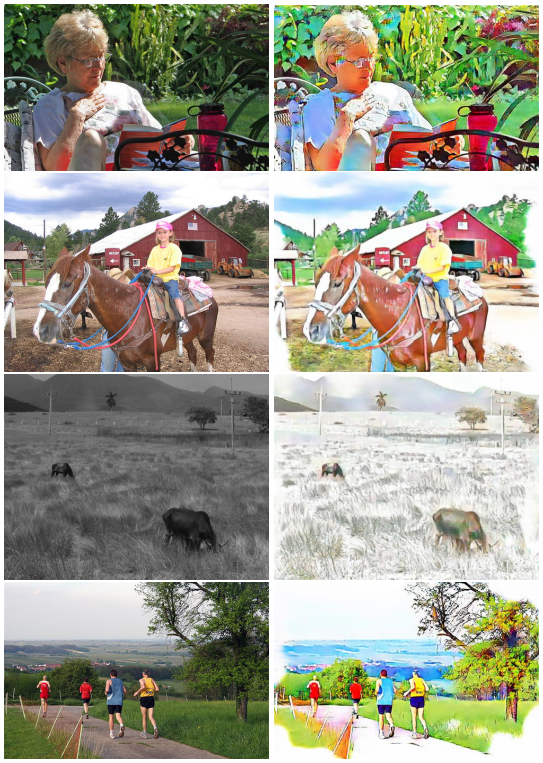


Figure 4. Examples of source domain images in target domain obtained by CycleGAN[8]

Training: We are using the images of the source domain without any annotations, as CycleGAN[8] is fully unsupervised.

Testing: Source domain images are transferred in the target domain. Fig.3 shows the example images generated by DT. Visibly, the perfect mapping is not accomplished in this experiment as the representation gap between a natural image domain and the other domains is too wide as compared with the gap between synthetic and real images tackled in recent studies. CycleGAN[8] seems to transfer color and texture while keeping most of the edges and semantics of the input image.

5.3. SSD Model: Augmented Images of CycleGAN

In this setting, we fine tune the baseline model of SSD[2] and train it with the augmented images that we got from CycleGAN[8], the images are similar to the target domain and after training our model for 10k iteration with learning rate 1e-5 we test our model on the target domain and the result is showed in Table 2. There is an increase in performance in compare with our baseline model. Unfortunately, we are not able to reach the reference paper results[10].

5.4. SSD Model: Style Transferred Images through AdaIN

In this setting , we fine-tune the baseline model of SSD[2] and train it using the style transfer images of AdaIN[6] model, after training our model for 10k iteration with learning rate 1e-5, p=0.5 and training a batch of 32 images. We test our model on the target domain and the result is showed in Table 2. We see there is a increase of performances compared with the baseline and the previous DT1 model, moreover by adopting this style transfer we were able to improve on the target paper DT only score.

5.5. Ideal Case

In this setting, we had access to the instance-level annotations for the training split of the target domain dataset. We simply fine-tune the baseline model of SSD[2] using these instance'level annotations. This experiment was to confirm the weak upper-bound performance of the model. We test our model on the testing split of the target domain dataset and the result is showed in Table 2.

6. Results on Clipart1k

The results of our implementation based on SSD300[2] are shown in Table 2. It shows the comparison of average precision (AP) for each class and mean average precision (mAP) among other methods against our baseline and the comparable reference paper implemtentation.

The performances of our work are examined using two different configurations. (i) DT1: fine-tuning the baseline by using images obtained through cycleGAN[8] domain transfer.(ii) DT2: fine-tuning the baseline by styling images online using AdaIN[6].

We observe that SSD300[2] perform not so well in the baseline as it was not trained to adapt on the target domain, it is however in line with the reference paper result. DT1 provides an improvement of 6.09 percentage points improvement from the baseline SSD300[2] in terms of mAP. Finally, DT2 provides an improvement of 13.39 percentage in terms of mAP. Most importantly, AdaIN[6] slightly outperforms the reference paper DT implementation. In the end, our result confirm that both domain adaptation methods gives better performances.

7. Conclusion

In this paper we are able to show that it is indeed possible to perform visual object detection in a unlabeled target domain. We were able to achieve improvements by using two different domain adaptation methods, both offline (DT1) and online (DT2). Experimental results demonstrated that our domain adaptation methods show decidedly better performance when compared with the baseline, and are at least comparable to other existing implementations.

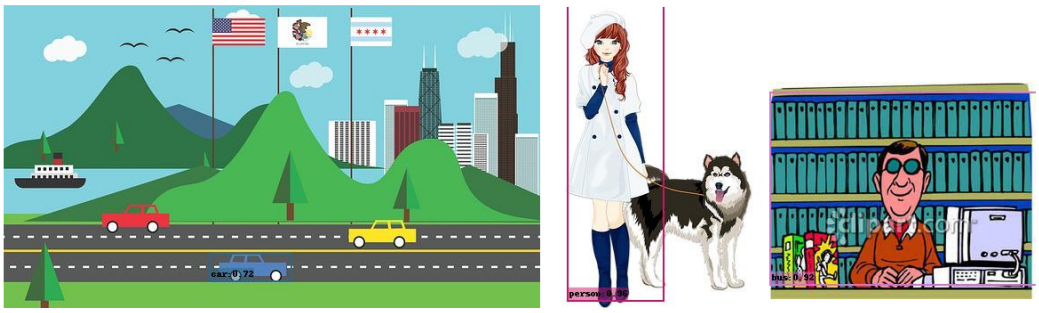


Figure 5. Example outputs for our baseline in the target domain. We see in the first image it fails to perform detection, in the second image only the person is detected and in the last it fails again to perform detection.

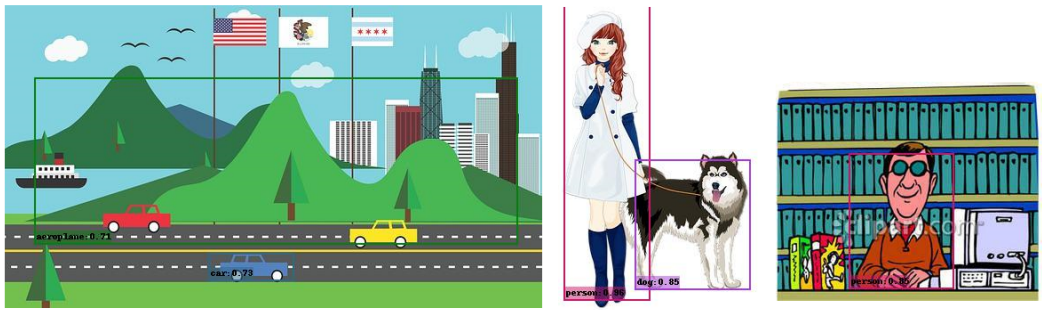


Figure 6. Example outputs for our DT1 in the target domain. We see in the first image it is able to detect the blue car while the red car is detected wrong and it fails to detect the yellow car, in the second image this time it also detects the dog along with the person and in the last image it detects the person.

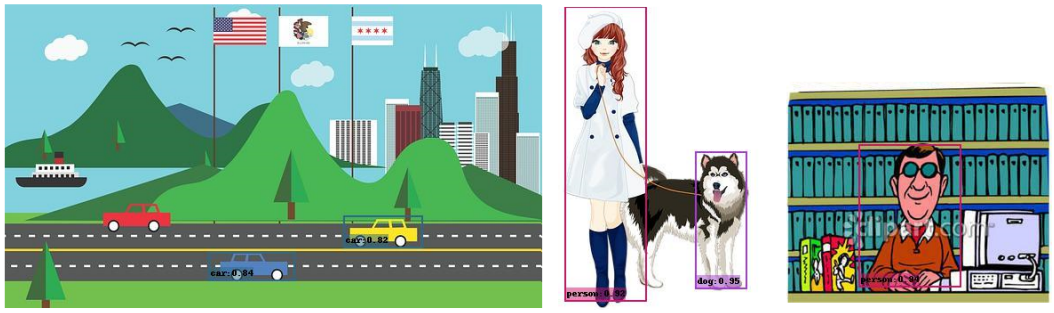


Figure 7. Example outputs for our DT2 in the target domain. We see in the first image it is able to detects this time two cars correctly , in the second image and third image it detects same like DT1, but with greater confidence score.

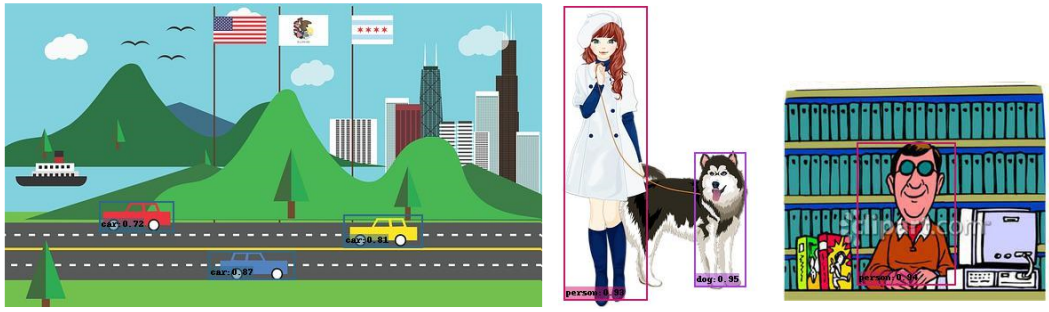


Figure 8. Example outputs for our ideal case in the target domain. We see here in all the images the detection is correct and the confidence score both are high.

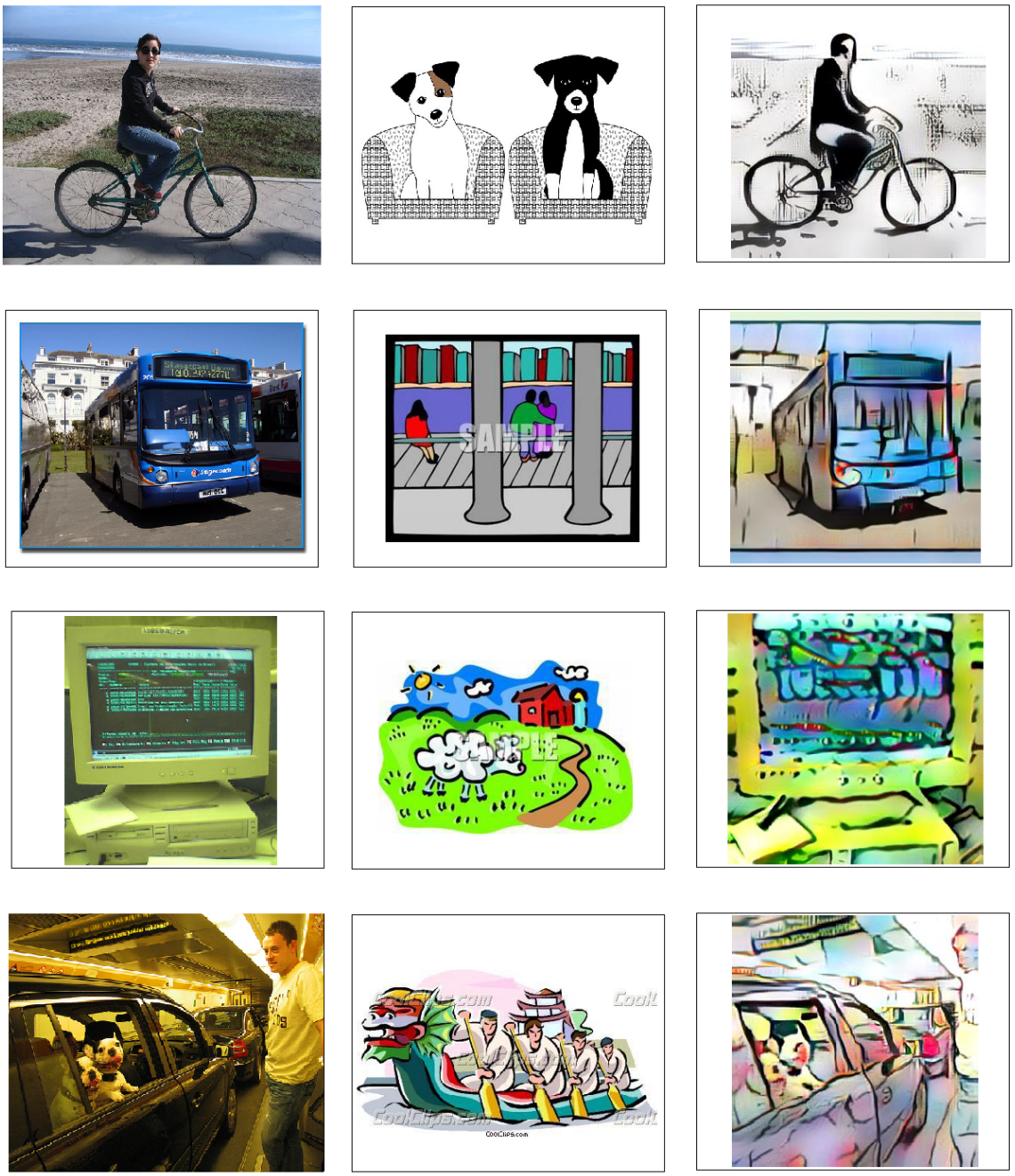


Figure 9. Style Transferred Images through AdaIN training, from Content and Style Images[6]

Name	Learning Rate	Gamma	LR Steps	Max Iteration	mAP
Paper	1e-3	0.1	60k	80k	26.8
Setup 1	1e-3	0.1	40k	60k	25.6
Setup 2	3e-4	0.1	40k	60k	24.5
Setup 3	1e-3	0.1	[40k, 50k]	60k	25.9

Table 1. Experimental results of Test in the Clipart1K dataset with different learning rate for SSD300[2] as the baseline.

Class	Reference Baseline	Baseline	DT1	DT2	Ideal
aeroplane	19.8	19.24	23.48	29.65	49.61
bicycle	49.5	48.68	56.83	63.79	69.58
bird	20.1	20.52	22.93	22.83	40.67
boat	23.0	17.96	20.81	24.83	50.10
bottle	11.3	14.62	18.01	20.37	32.49
bus	38.6	34.43	35.65	46.60	66.61
car	34.2	30.53	38.61	44.49	57.55
cat	2.5	11.64	6.97	5.02	16.29
chair	39.1	44.25	46.48	49.48	53.37
cow	21.6	10.91	42.60	54.82	69.20
table	27.3	27.53	29.86	40.54	61.76
dog	10.8	13.07	9.30	21.06	27.16
horse	32.5	25.36	29.81	34.49	55.60
motorbike	54.1	54.43	57.77	74.64	77.55
person	45.3	39.85	54.18	62.36	73.81
plant	31.2	26.24	27.90	33.13	40.50
sheep	19.0	19.98	18.78	18.39	43.18
sofa	19.5	25.29	37.54	45.77	55.76
train	19.1	19.85	29.54	38.46	64.38
tv	17.9	22.89	32.02	54.24	66.71
mAP	**26.8**	**25.86**	**31.95**	**39.25**	**53.60**

Table 2. Comparison of all the methods in terms of mAP using SSD300[2] as the baseline FSD in Clipart1k.

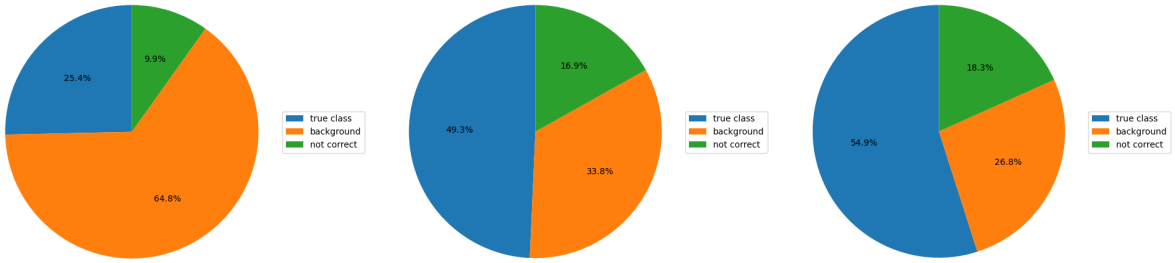


Figure 10. shows the example of the error analysis in the Clipart1k test set. Comparing the baseline DT1 and DT2, we observe that fine-tuning the FSD on images obtained by DT1 and DT2 improves the detection performance, especially in less-confident detections (confidence is equal or greater than 0.1) . The analysis was performed on person category in test set; The true class indicates the correct output labeling on the images, The background class shows the percentage of images which was not recognized by the detector and Not correct class presents the number of images with the wrong bounding boxes.

References

[1] M. J. R. B. S. A. Gretton, K. M. Borgwardt and A. Smola. A kernel two-sample test, 2012. JMLR, 13(Mar). 2

[2] W. L. et al. Ssd: single shot multibox detector, 2016. Lect. Notes Comput. Sci. 1, 2, 3, 4, 5, 7, 8

[3] S. B. B. C. F. M. Carlucci, A. D'Innocente and T. Tommasi. Domain generalization by solving jigsaw puzzles, 2019. 2019, pp. 2229–2238, Accessed: Nov. 25. 2

[4] R. Girshick. Fast r-cnn. in: Proceedings of the ieee international conference on computer vision, 2015. pp. 1440–1448. 1, 2

[5] G. G. D. P. G. R. He, K. Mask r-cnn. in: Proceedings of the ieee international conference on computer vision, 2017. pp. 2961–2969. 2

[6] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017. ArXiv170306868 Cs , Jul. 2017, Accessed: May 15, 2020. 1, 2, 3, 4, 5, 7

[7] L. X. J. Xu and A. M. López. Self-supervised domain adaptation for computer vision tasks, 2019. IEEE Access , vol. 7. 2

[8] P. I. J.-Y. Zhu, T. Park and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2017. in 2017 IEEE International Conference on Computer Vision (ICCV) , Oct. 2017, pp. 2242–2251. 1, 2, 4, 5

[9] C. K. W. J. W. M. Everingham, L. Van Gool and A. Zisserman. The pascal visual object classes (voc) challenge, 2010. IJCV, 88(2). 3

[10] T. Y. N. Inoue, R. Furuta and K. Aizawa. Cross-domain weakly-supervised object detection through progressive domain adaptation, 2018. in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 1, 2, 3, 5

[11] D. S. G. R. F. A. Redmon, J. You only look once: Unified, real-time object detection, 2016. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788. 1, 2, 3

[12] H. K. G. R. S. J. Ren, S. Faster r-cnn: Towards real-time object detection with region proposal networks. in: Advances in neural information processing systems, 2015. pp. 91–99. 1, 2, 3