Cross-Domain Object Detection with YOLO

Michele D'Addetta
Politecnico di Torino
s257801@studenti.polito.it

Lorenzo Montoro
Politecnico di Torino
s266049@studenti.polito.it

Giorgio Giacalone
Politecnico di Torino
s267545@studenti.polito.it

Abstract

There are some papers about cross-domain classification, but only few of them are about object detection. In this paper, we present the results obtained and the methodologies adopted for cross-domain supervised object detection, taking inspiration from the paper made by Inoue et al [1]. The novelty of this paper is the use of multi-source approach on domains characterized by instance-level annotations, combined with images having only image-level annotations in a target domain. The classes to be detected in the target domain are a subset of those in the source domains. We start from a YOLO implementation using a pre-trained Darknet, and then we apply two deep learning procedures aimed to increase the performance of our model. Finally, we test our network on a subset of the target dataset, achieving an improvement of accuracy in detection of 5-15%, depending on the used method, with reference to the detections made with the pre-trained network.

1. Introduction

The idea behind this paper is to train a network implementing the YOLO algorithm to detect objects in a target domain, which has only information about objects in the images, but no information about where they are placed. So, our goal is to use instance-level annotations of source domains and try to generate them on the target one. To make this possible, we have used a Darknet implementation by Redmon et al [2] from scratch. Object detection is the process of autonomously recognizing objects (e.g. people, animals, cars etc.) in an image or video, not only classifying those objects correctly deciding their class, but also recognizing their positions and sizes in the input image. In fact, object detection is about assigning the right label (i.e. the class) and the right bounding box (i.e. a rectangular shaped frame) to an object. So, we define as image-level annotation the set of classes coupled with the objects contained in a certain image, without knowing where objects are. The knowledge of the class and the position of objects defines the instance-level annotation of an image. The position of an object is defined with a bounding box **b**, defined as *(xmin, xmax, ymin, ymax)*, where *xmin* and *xmax* are the coordinates in pixel of the leftmost and rightmost vertexes respectively. Same for *ymin* and *ymax*, that are the uppermost and lowermost vertexes of the box.

Right now most object detection networks and researches, focus on images from the real world obtaining great results, but object detection can be very useful also in other domains. The methods explained in this paper aim for detect objects in domains covered by few datasets with lack of annotations, i.e. painting or comic. This is very useful to improve performances on them and transfer knowledge from a well-known domain. For example, an automated museum guide will exploit the knowledge obtained from these techniques to recognize objects in form of statues, paintings, etc.

We start from a Darknet pretrained on images with instance-level annotations from a source domain, then we fine-tune it with the target domain. This approach seems the best one, but there are no instance-level annotations available in the target domain. To generate this information, we use the same methods applied in [1], that are *Domain Transfer* and *Pseudo Labeling*. To perform *Domain Transfer*, we use a CycleGAN implemented by Zhu et al [3], in order to transform data from source domains in the target one. Once the annotations are automatically created, we fine-tune the Darknet on them. The results obtained by the original paper applying this task achieve an improvement of 5 to 20 percentage points in terms of mean average precision, but the domain adaptation was made starting from a single source domain. The source was always Pascal VOC and it is used to train three networks in order to transfer instance-level knowledge to three different domains, that are Clipart, Comic and Watercolor. In our implementation, we want to know how much the results change if we use several sources and transfer all their information to the target one. All the sources have instance-level annotations, meanwhile the target dataset has only image-level annotations. The domains we take as source domains are Pascal VOC 2007 and 2012, Watercolor and Clipart. Comic, instead, is used as target domain.

2. Related Work

2.1. Supervised Detection: YOLOv3

Many methods can be used to realize object detection in images and video, such as Fast R-CNN, Faster R-CNN, YOLO. The first two use a similar approach: they first define some region of interest in the image, then they try to classify the objects into them. These two approaches are more accurate than YOLO, but they are slower. In fact, YOLO is faster and able to detect objects in real time, without a big latency between the request and the response. The approach of YOLO in the detection is quite different, and this is because YOLO splits the image in a grid and creates a set of bounding boxes in each of them. Then it regresses from each box to a box made by *(xmin, xmax, ymin, ymax, confidence)*, where the first four values have the same meaning explained before, and the confidence says the probability of having a correct prediction. Once this is done for each box, it predicts scores for the classes in the dataset, including the background class, and outputs them. YOLOv3 can learn instance-level annotations from a training dataset and then detect objects in an image in about 40 milliseconds, which makes it suitable for real-time detection.

2.2. Cross-Domain Object Detection

The target domains may have few or zero information about the position of the objects in the images. This may lead to an unfeasible training. The idea is to take instance-level annotations from a source domain and use them in the target one. So, the network learns from a well-known domain, that has big and several datasets of images coupled with instance-level annotations. But if it tries to predict objects in other domains' images, the results significantly worsen, because the features in the images are different and the network is not able to understand them. In order to increase performance, the network has to use knowledge of the source domain and has to extract patterns from target one, making the prediction more reliable. As we said before, we use the same approach, in terms of methods, of [1], where they take a real-world dataset as source and adapt it to a target one, which has only image-level annotations. In our implementation, instead, we use multi-domain source and one target domain. The domains used as source have all the instance-level annotations. We also have instance-level annotations of the target one, but we use them only for testing, in order to get the prediction precision. We want to know if the results get worse with multiple domains or if they can be used to make the network more robust.

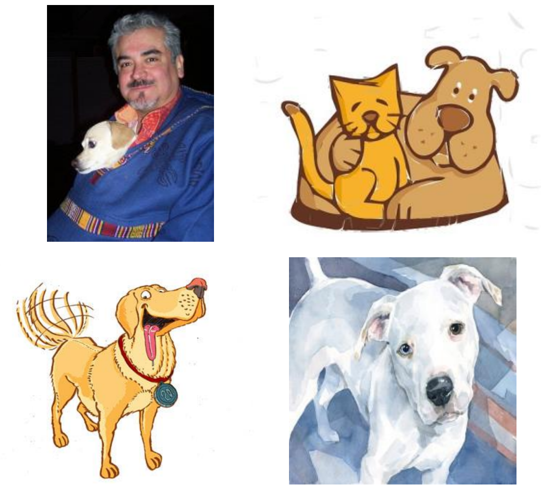
Figure 1: Samples of images from the four used datasets: Pascal VOC, Clipart, Comic, Watercolor (from upper left to lower right)

2.3. Domain Adaptation

There are lots of methods to perform the domain adaptation task. The one proposed here is the CycleGAN, a model able to translate images from a source to a target domain and vice versa. It uses two GANs, which are unsupervised generative models able to generate images similar to pictures of the set used as training. The GAN uses an implicit density distribution to generate data and uses a two-player approach. There are a generator and a discriminator in each of the GAN, that respectively tries to fool the discriminator creating real-like images and to distinguish between real and fake images. The CycleGAN generates a fake image from source to target and tries to reconstruct the image from the target to the source. The cycle-consistency losses are used to update the parameters in the network in order to create images from source to target.

3. Dataset

In our implementation, four datasets have been used. The source ones are Pascal VOC [4], Clipart and Watercolor. The target one is Comic. Comic and Watercolor datasets belong to BAM! dataset [5], which contains several domains, but some of them are not suitable for the detection, because they contain just one object placed in the center in most of the images, so the detection is not so challenging. All the datasets contain instance-level annotations, but as explained before, the target ones will be used only for testing. Examples of images from each dataset are shown in *Fig. 1*.

3.1. Pascal VOC

Pascal VOC is the biggest dataset among the used ones, and it is composed of images from real world. The dataset consists of two subparts made in different years, which are 2007 and 2012. The first one contains 9963 images, whereas the second one has 17125 images. Both datasets address 20 classes. This set is used to pre-train the Darknet and to transform the contained images in the target domain for the *Domain Transfer* phase.

3.2. Clipart

Clipart is a dataset made by drawings, pictures and cartoons-like images. It contains 1000 images belonging to 20 classes, that are the same of Pascal VOC. The images have been taken from CMPlaces dataset and correctly annotated. This set is used during the *Domain Transfer* phase.

3.3. Watercolor

Watercolor is a subpart of the BAM! dataset and contains 2000 paintings, which are made by objects of 6 classes. This set is used during the *Domain Transfer*.

3.4. Comic

Comic is a part of BAM! dataset too, and, as the name says, it has images taken from comics. The dataset is made of 2000 images, some of them are colored and others are black and white. The classes belonging to the dataset images are 6. This set is the target domain, so it is used as aim during the *Domain Transfer* and to perform the *Pseudo Labeling*.

Remark that both Comic and Watercolor have only 6 classes, specifically *bicycle*, *bird*, *car*, *cat*, *dog* and *person*. These classes are a subpart of the one of Clipart and Pascal VOC.

4. Proposed Method

In order to achieve the project goal, we use YOLOv3 as our object detector algorithm. It is implemented by the Darknet network. We pre-train it on Pascal VOC dataset so that it starts learning something about the images features and the information contained in them. However, if we try to detect objects directly on the target domain, the results are very poor, because it hasn't instance-level information about Comic to be used for training. In fact, source and target domains are from different distributions, and the accuracy of the prediction decreases significantly. To improve the model, we apply *Domain Transfer* and *Pseudo Labeling*. These are two methods also implemented by [1] able to increase the prediction accuracy in the target domain without having instance-level annotations about them. This is done transferring knowledge from the source domains to the target one. Our purpose is to implement a multi-source approach and see if the model can reach higher level of accuracy. If it works better, then this approach is extendible to all the unknown target domains starting from the knowledge of multiple sources, considered reliable and very detailed. The steps followed during the workflow are shown in *Fig. 3*.

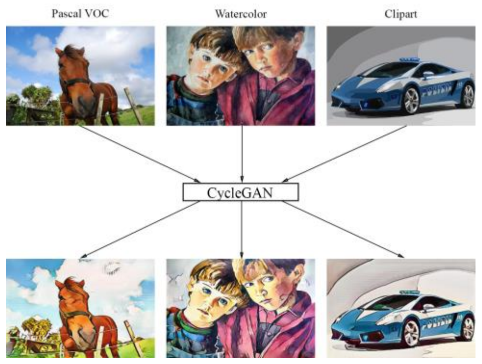

Figure 2: Examples of images transformed by Domain Transfer from the sources (top) to the target (bottom).

4.1. Pre-training

Darknet implementation is taken from [2]. Once loaded, the object detector needs to gain some knowledge about the source domains, so as starting point it has been pretrained over 30000 iterations with LR of 0.001 and using a step-down policy, that reduces the learning rate after 15000 and 25000 steps, with a γ equal to 0.1.

4.2. Domain Adaptation

Feature and output spaces address to the same task, namely to create instance-level annotations, but they have very different marginal distributions. If we plot domains in the features space, they appear in a very different way. The main idea is to take images from source domains and transform them in images similar to the target ones. Then their instance-level annotations can be used by the network to understand images from a new domain, learning their features using the original domain annotations. In order to generate new samples, that will have the same appearance of the target domain, we use a CycleGAN. Each generated image contains the same objects of the original, having a comic fashion.

The implementation is taken from [3] and it is trained for 20 epochs over the three source domains creating new samples. In the first ten epochs, the CycleGAN uses a learning rate of $2x10^{-4}$, then, in the last ten epochs, the learning rate is decreased linearly after each epoch. After the training, we obtain the parameters of the network, that will be used to transform all the source images. Generated

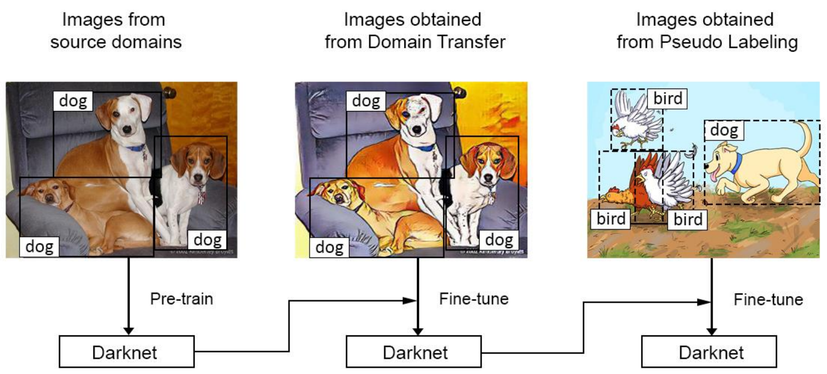
Images from source domains
Images obtained from Domain Transfer
Images obtained from Pseudo Labeling
dog
dog
dog
dog
dog
dog
bird
dog
bird
bird
Pre-train
Fine-tune
Fine-tune
Darknet
Darknet
Darknet

Figure 3: The steps in the workflow.

images have been stored and used to fine-tune the Darknet for 10000 iterations with a constant LR value of 10^{-5}. The training configuration used by the creators of [1] is different, but when we tried it the results were sub-optimal. In fact, the network didn't gain enough information about those generated images. Using our training configuration, the detector will try to extract as more instance-level annotations from the generated images as it can. The problem is that the quality of images generated by the CycleGAN is lower than the real target domains pictures, but after this phase, the Darknet has a deep knowledge of instance-level annotations. These can be used to classify the target domain images. Samples of generated images for each source dataset are shown in *Fig. 2*.

4.3. Pseudo-Labeling

So far, our object detector may predict images of target dataset, but the results are not so accurate. In fact, even if the Darknet is trained over images that look like Comic ones, the real target images are quite different.

In order to achieve better performance, Darknet needs to understand as much as possible the real data from the target domain, in order to classify images correctly.

Pseudo Labeling technique, also implemented by [1], is the choice made to help the model to gain more accuracy on target domain, by creating pseudo instance-level annotations for each image from target domain. The objective is to classify each image of Comic with the parameters obtained from the *Domain Transfer* and pick, for each class from its image-level annotation, the top-1 confident detection. In our implementation, we select the best prediction for each class present in a certain image, according to its image-level annotation. If an image contains more than one instance of a certain class *c*, we take the *k*-most confident predictions on *c*, where *k* is the minimum between the obtained predictions for *c* and the number of instances of *c* in the image. Remark that if the Darknet predicts an object belonging to a class that hasn't instances in a certain image, this prediction is discarded. In addition, we also implement the *PL* only taking the top-1 confident detection for each class in a certain image. More details will be explained in *section 5.1*. Finally, we save them as instance-level annotations and discard the least confident ones. The created pseudo-annotations are used to fine-tune the Darknet for 10000 iterations, using a learning rate of 10^{-5}.

5. Experiments and results

5.1. Evaluation metrics and test cases

In order to evaluate the accuracy of the detection on the target domain, we use Average Precision (*AP*) and its mean value (*mAP*).

We calculate the Average Precision (*AP*) for each class of the target dataset, and then we average these results to obtain *mAP*. As explained before the classes of the target domain, that is Comic, are six, and all of them are a subset of the source domains classes.

We evaluate three different cases:

- *Case 1*: we pre-train Darknet on Pascal VOC and we use the same parameters and network architecture to fine-tune during the *Domain Transfer*. Then, in order to execute the *Pseudo Labeling*, we define the pseudo-labels on the training images, discard the predictions of objects not in the set of Comic classes and we modify the last layer of the Darknet. In this way, the net will be able to classify only objects belonging to the six classes of Comic.
- *Case 2*: we pre-train Darknet on Pascal VOC, but, in this case, we change the last layer of the Darknet, and we use the pretrained weights architecture to fine-tune during the *Domain Transfer*. So, now during the *Pseudo Labeling* we use a network that only knows the classes of the target domain. The pseudo-annotations are generated using the technique explained before in the *Pseudo-Labeling* paragraph, taking the top-k predictions for each class.
- *Case 3*: we pre-train Darknet on Pascal VOC, and also in this case, we change the last layer of the Darknet,

and we use the pretrained weights architecture to fine-tune during the *Domain Transfer*. The pseudo-annotations that will be used during the fine-tuning are generated using the *Pseudo-Labeling*, but in this case, we take only the top-1 confident detection for each class for all the images. This is the approach used in [1].

Another parameter we keep in consideration, for the calculation of results, is the *IoU* (Intersection over Unit), which is the percentage of overlap between the bounding boxes of ground truth and the bounding boxes predicted by the Darknet. This parameter is used to determine a threshold to define if a predicted bounding box can be considered correct or not. In our case, we use a threshold for *IoU* of 50%, which means that if a detected bounding box overlaps more than the half of the ground truth bounding box and the detected class corresponds to the real one, then the predicted bounding box is considered correct.

In order to calculate *AP*, the detector predicts classes and bounding boxes on the test images, then it ranks all the predictions in descending order depending on the value of confidence of the prediction. Then, if the *IoU* is greater than 50% and the class is correct, the detection is considered as *True Positive (TP)*, otherwise there will be two possible classifications of the detection:

- *False Positive (FP)*, if the *IoU* is lower than 50% or the bounding box around the object is duplicated
- *False Negative (FN)*, if the *IoU* is greater than 50%, but the predicted class is not the real class

Once the predictions are ranked and classified in terms of *TP*, *FP* and *FN*, the precision is calculated for each class as:

$$AP(c) = \frac{\sum_{i=0}^{n} \frac{nTP}{i}}{n} \qquad (1)$$

where *n* is the number of objects predicted for the class *c*, *nTP* is the number of objects correctly detected until the *i-th* analysed detection.

The *mAP* value is computed as follows:

$$mAP = \frac{\sum_{c=0}^{C} AP(c)}{C} \qquad (2)$$

where *c* is a class of the dataset, *C* is the total number of classes in the target dataset and *AP(c)* is the Average Precision of class *c*.

We test the accuracy of the detection after Domain Transfer (DT) and then also after Pseudo Labeling (PL). As test set, we split the whole Comic dataset and use a part as test and the other during the fine-tuning of the *Pseudo Labeling*. As training set, we use the whole Pascal VOC and half of Clipart and Watercolor datasets. We leave the other two splits because we use them for testing in the intermediate results of the entire workflow.

5.2. Results and discussion

	AP for each class						
Method	bicycle	bird	car	cat	dog	person	*mAP*
Pre-train	25.7	6.6	9.5	8.4	7.4	21.0	13.1
Case 1							
DT	43.8	20.4	22.0	15.7	19.6	53.2	29.1
DT+PL	0.3	0.5	8.1	1.3	0.2	44.2	9.1
Case 2							
DT	34.2	13.5	19.6	13.5	13.9	43.3	23.0
DT+PL	6.7	10.4	21.5	11.1	17.9	40.4	18.0
Case 3							
DT	34.2	13.5	19.6	13.5	13.9	43.3	23.0
DT+PL	28.5	11.3	20.3	11.1	20.3	42.0	22.3
DT+PL1000iters	35.0	14.2	19.8	11.7	19.2	42.8	23.8

Table 1: *AP* results for each class and *mAP* in percentage.

The *Pre-train* row in the *Table 1* shows the *mAP* of the pretrained network using Pascal VOC and tested on the target dataset. As expected, the results are very poor, because the Darknet has no knowledge on the target domain.

As the *Table 1* shows, the results obtained using the first approach are very good for the *Domain Transfer*, meanwhile the *Pseudo Labeling* phase worsen. This is because the Domain Transfer is done using all the source classes and only in the Pseudo Labeling the last layer is changed. In fact, the pseudo labels cannot improve the detection as expected.

Analyzing the results of the case 2, we see that the *Domain Transfer* obtains quite good results, lower than the previous ones, but more reliable. In fact, case 1 predicts also objects of classes not in the target domain, and the *AP* on those classes is equal to 0. The real *mAP* in the first case was 8.7, but using only *AP* of the target classes, we obtain the result shown in the *Table 1*. The *mAP* decreases during the *Pseudo Labeling*, and we address this behavior to the usage of the top-k detections for each class in the image. These detections may be wrong, and therefore we also try the case 3, where we expect more confident and accurate top detections.

About the last case, the results given both in *Domain Transfer* and *Pseudo Labeling* are very similar, with a slight decrease after the PL. We expected higher *mAP* for the latter one, but it seems that the Darknet is overfit or our implementation of *Pseudo Labeling* may be not completely correct. Another possible cause of the deterioration may be due to the multiple source domains used during the Domain Transfer. In fact, the pseudo labels are generated after the fine-tuning of the *DT* phase, and these may not be correct because the annotations are not supervised, but automatically created.

Executing the *PL* fine-tuning, the loss value decreases during the iterations, but also the *mAP* decreases. Testing the net with weights of the Darknet after 1000 iterations of *PL* fine-tuning, the *mAP* value obtained is 23.8%. This is

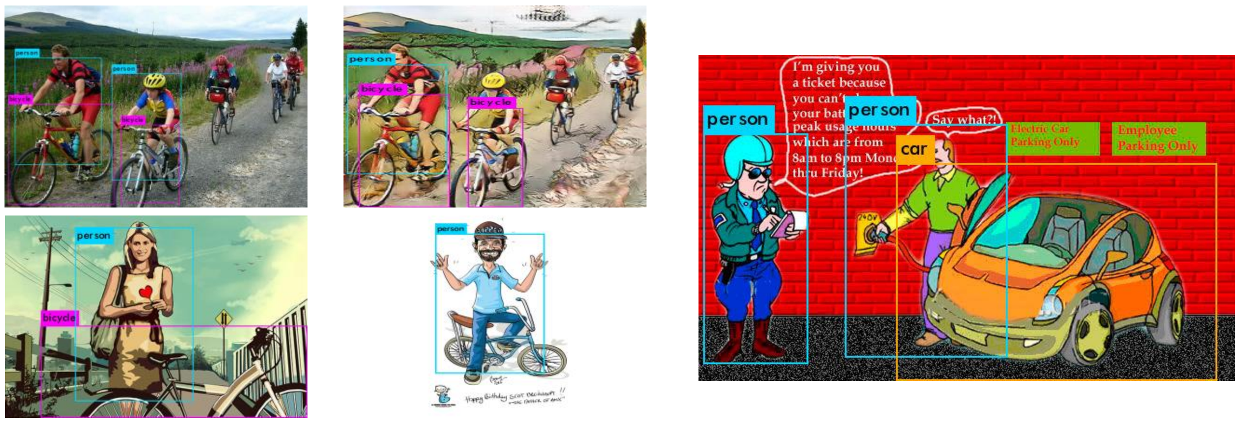

Figure 4: Examples of detection on images from Pascal VOC, generated images from Pascal VOC and images from Comic dataset

the highest computed during the whole experience, so we think that the best way to do *PL* with our implementation is to run the tuning only for few iterations. This is a big difference between our implementation and [1], which uses 10000 iterations.

Moreover, comparing our results and the [1] ones, the latter obtain a better *mAP*, and we attribute this to the multiple source domains or to a possible mistake in the implementation of *PL*. The *APs* for each class are shown in the last row of *Table 1*.

5.3. Graphs

The loss graphs for both CycleGAN and Darknet during the training are not reported because we had to stop often the training during the time. This is due to the session time of Google Colab, that every 10 hours disconnects the sessions and requires to create a new one, losing all data created in it. Even if the intermediate graph is stored and loaded in each session, it is cleared and a new one is created every time the training restarts.

6. Conclusion

Detection of objects in a domain that doesn't have instance-level annotations for training improve quite significantly using the proposed methods, so this approach can be considered a good workflow that can be adapted to any domain needed. In fact, we consider these techniques a very powerful tool, because of the flexibility of its usage among every possible domain. However, if results obtained with *Domain Adaptation* are similar to the ones obtained by [1], we can't say the same thing about *Pseudo-labeling*. So, in a future work, or having more time to spend on this project, we would like to inspect with more attention the implementation of *PL* in order to find possible problems and make it work as expected. In fact, in [1], the highest accuracy in detection is obtained with *DT* and *PL*, but we haven't been able to reproduce the same results.

References

[1] Naoto Inoue, Ryosuke Furuta, Toshihiko Yamasaki, Kiyoharu Aizawa. Cross-Domain Weakly-Supervised Object Detection through Progressive Domain Adaptation. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[3] Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. The IEEE International Conference on Computer Vision (ICCV), 2017.

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

[5] M. J. Wilber, C. Fang, H. Jin, A. Hertzmann, J. Collomosse, and S. Belongie. BAM! the behance artistic media dataset for recognition beyond photography. In ICCV, 2017.