# **Review on Paper** - *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*

Kangyan Xu

Dec 2020

**Abstract**

In this paper, the author introduced a method to translate images from a source domain X to a target domain Y, without learning from paired examples. Two loss function are used to achieve good performance, the Adversarial Loss and the Cycle Consistency Loss.

## 1 Introduction

What is Image-to-Image translation? The idea is to learn a mapping using a training set, so that an input image from a source domain X can be translated to a target domain Y using this mapping. However, it's not quite practical (though ideal) to use paired images for this training. Suppose that you see the famous *Impression, Sunrise* in gallery, you know its style is Impressionism, but the scene the painter facing (water, boats, cloud, sun) is long lost, you cannot get such a photo to consist a pair of images for learning a mapping from Impressionism Painting to Realistic Photo. Fortunately, you can get many paintings with the same Impressionism style and many photos containing beautiful scenes, though unpaired. Both image collections have their own special characteristics, so we can learn them to realize our mapping.

## 2 Ideas

We want to train a mapping $G: X \to Y$, with the output $\hat{y} = G(x)$ indistinguishable from the real $y$ by an *adversary* trained to distinguish $\hat{y}$ from $y$. So the mapping $G$ can translate domain $X$ to domain $\hat{Y}$, with $\hat{Y}$ distributed identically to $Y$. However, this is not enough. Nothing guarantees this mapping will be exactly the one we want, there lack too many constraints.

The idea of *cycle consistent* is introduced here. That is, we have not only the mapping $G: X \to Y$, but also the mapping $F: Y \to X$, both are bijections. The ideal circumstance is $F\big(G(x)\big) = x$ (forward one) and $G\big(F(y)\big) = y$ (backward one), so we add *cycle consistent loss* to train the mapping $G$ and $F$.

With both *adversary loss* and *cycle consistent loss*, we have the *objective* for our translation.

# 3 Further Implementation

According to the paper, the translation approach builds on the "*pix2pix*" framework of Isola et al., which uses a conditional generative adversarial network to learn a mapping. The network architecture is adopted from Johnson et al.

Two *adversarial discriminators* are introduced, $D_X$ aims to distinguish $\{x\}$ from $\{F(y)\}$ and $D_Y$ to distinguish $\{y\}$ from $\{G(x)\}$.

## Adversarial Loss

Define: $L_{GAN}(G, D_Y, X, Y) = E[\log(D_Y(y))] + E[\log(1 - D_Y(G(x)))]$
where $G(x)$ translates $x$ to domain $Y$ and $D_Y$ aims to distinguish $y$ from $G(x)$.
$G$ tries to minimize the objective while $D_Y$ tries to maximize it.

Similarly we have adversarial loss for mapping $F$. So we have:

1) $min_G max_{D_Y} L_{GAN}(G, D_Y, X, Y)$
2) $min_F max_{D_X} L_{GAN}(F, D_X, Y, X)$

## Cycle Consistency Loss

Define: $L_{cyc}(G, F) = E\left[\left\|F\big(G(x)\big) - x\right\|_{l1}\right] + E[\left\|G\big(F(y)\big) - y\right\|_{l1}]$
The author has tried replacing the $L_1$-norm and see no improvements.

## Full Objective

Loss: $L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda \cdot L_{cyc}(G, F)$

Optimal: $G^*, F^* = \arg min_{G,F} max_{D_X, D_Y} L(G, F, D_X, D_Y)$

## Further Details

In the training, $L_{GAN}$ is modified using least-squares loss, that is:

For $L_{GAN}(G, D, X, Y)$
1) Training $G$ to minimize $E\left[(D\big(G(x)\big) - 1)^2\right]$
2) Training $D$ to minimize $E[(D(y) - 1)^2] + E\left[(D\big(G(x)\big))^2\right]$

Besides, to reduce model oscillation, using previous created images instead of latest generated ones.

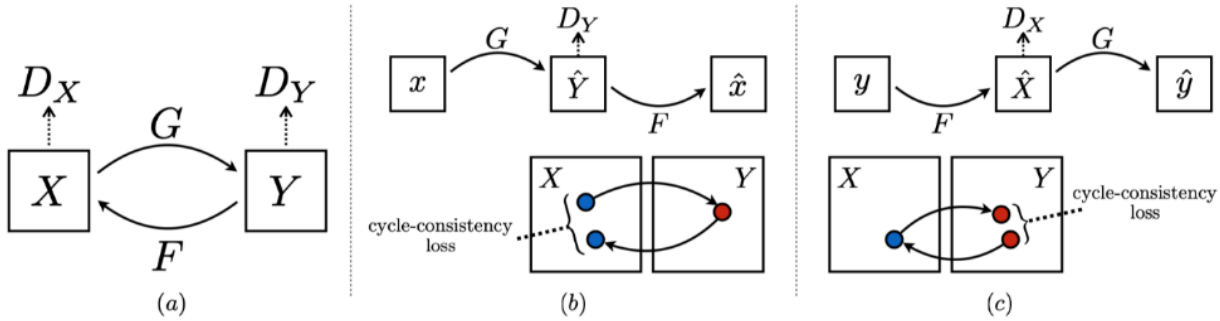

Figure 1: model illustration (from original paper)

## 4 Results

The author compares this method with other methods and studies the necessity of both types of losses.

Especially in the cycle consistency loss, remove one direction (forward: $E\left[\left\|F\left(G(x)\right)-x\right\|_{l1}\right]$ or backward: $E[\left\|G\left(F(y)\right)-y\right\|_{l1}]$) will cause training instability and model collapse.

## 5 Applications

There are multiple applications, including *Collection style transfer*, *Object transfer*, *Season transfer*, *Photo generation from paintings*. Especially in the last one, an additional loss is introduced to preserve the color from inputs:

Loss: $L_{identity}(G,F) = E[\|F(x)-x\|_{l1}] + E[\|G(y)-y\|_{l1}]$

The author also mentions some failure in the application, such as minimal changes are made to the input images, or changes are applied to wrong areas in the image.

## 6 My Lab

The author provides *PyTorch* implementations (*CycleGAN*) and its runnable on macOS.

There are multiple *datasets* for downloading (creating a dataset is also possible). Moreover, *pretrained models* are also available.

First, I use the *monet2photo dataset* and *pretrained monet2photo CycleGAN model* for the experiment. All the images have size $256 \times 256$, the training set has 1074 Monet images (trainA) with 6289 photos (trainB).

```
(base) [XuKangyan:...ytorch-CycleGAN-and-pix2pix]$ python test.py --dataroot datasets/monet2photo/testA --name monet2photo_pretrained --model test --no_dropout --gpu_ids -1
----------------- Options ---------------
               aspect_ratio: 1.0
                    batch_size: 1
          checkpoints_dir: ./checkpoints
                    crop_size: 256
                      dataroot: datasets/monet2photo/testA           [default: None]
                dataset_mode: single
                    direction: AtoB
              display_winsize: 256
                        epoch: latest
                         eval: False
                      gpu_ids: -1                                    [default: 0]
                    init_gain: 0.02
                    init_type: normal
                     input_nc: 3
                      isTrain: False                                 [default: None]
                    load_iter: 0                                     [default: 0]
                    load_size: 256
             max_dataset_size: inf
                        model: test
                 model_suffix:
                   n_layers_D: 3
                         name: monet2photo_pretrained                [default: experiment_name]
                          ndf: 64
                        netD: basic
                        netG: resnet_9blocks
                          ngf: 64
                   no_dropout: True                                  [default: False]
                      no_flip: False
                         norm: instance
                    num_test: 50
                 num_threads: 4
                   output_nc: 3
                       phase: test
                  preprocess: resize_and_crop
                 results_dir: ./results/
               serial_batches: False
                       suffix:
                      verbose: False
----------------- End -------------------
dataset [SingleDataset] was created
initialize network with normal
model [TestModel] was created
loading the model from ./checkpoints/monet2photo_pretrained/latest_net_G.pth
---------- Networks initialized -------------
[Network G] Total number of parameters : 11.378 M
-----------------------------------------------
creating web directory ./results/monet2photo_pretrained/test_latest
processing (0000)-th image... ['datasets/monet2photo/testA/00010.jpg']
processing (0005)-th image... ['datasets/monet2photo/testA/00060.jpg']
processing (0010)-th image... ['datasets/monet2photo/testA/00110.jpg']
processing (0015)-th image... ['datasets/monet2photo/testA/00160.jpg']
processing (0020)-th image... ['datasets/monet2photo/testA/00210.jpg']
processing (0025)-th image... ['datasets/monet2photo/testA/00270.jpg']
processing (0030)-th image... ['datasets/monet2photo/testA/00320.jpg']
processing (0035)-th image... ['datasets/monet2photo/testA/00390.jpg']
processing (0040)-th image... ['datasets/monet2photo/testA/00440.jpg']
processing (0045)-th image... ['datasets/monet2photo/testA/00490.jpg']
```

Figure 2: model shows parameters

Figure 3 shows three typical results by the model which transfers Monet images (testA) to photos. As we can see, in 00010 the clouds are not quite real, but the sea and mountain are more real. In 00270 the generated image looks like a real photo, while in 00280 there seems no change between two images.

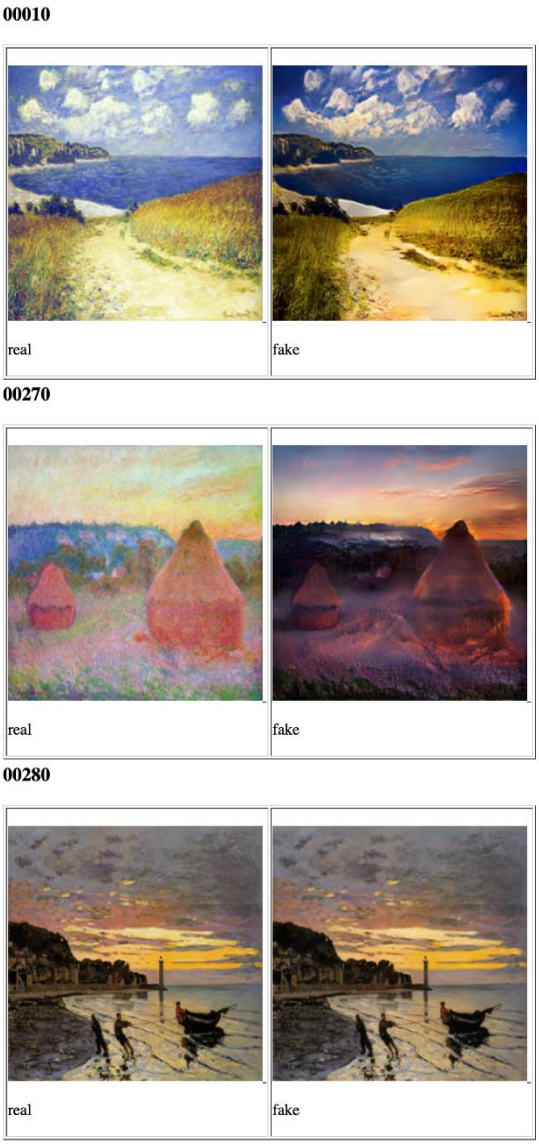


Figure 3: several model results

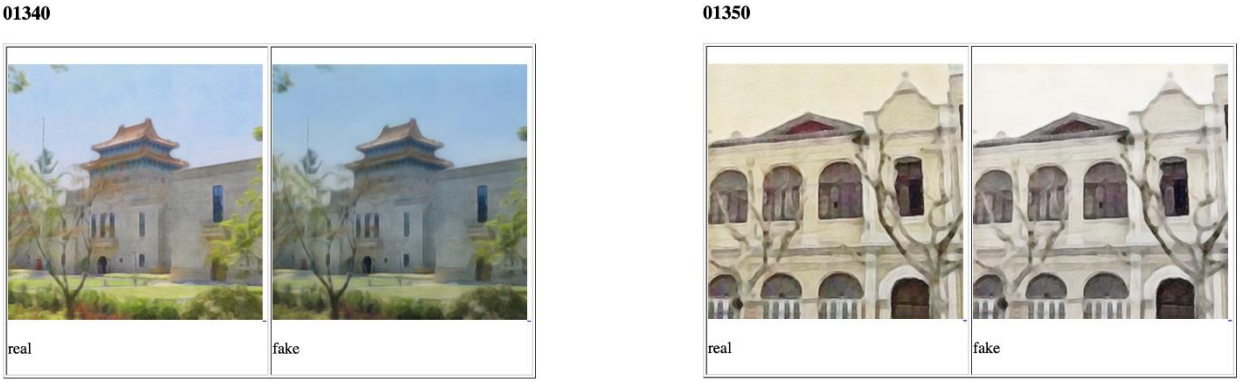


Figure 4: model results of “oil paintings”

Actually these two “oil paintings” are created by software using real photos. After the model transformation, the brush marks are eliminated.

Second, I tried to train the model using datasets on my local machine but receives no feedback (it takes too long to train). Reducing the size of training set or tuning the learning rate makes no progress.

The author trains the model using *horse2zebra* dataset and finishes training an epoch in ~90s, with a batch size of 16.

```
dataset [UnalignedDataset] was created
The number of training images = 1334
initialize network with normal
initialize network with normal
initialize network with normal
initialize network with normal
model [CycleGANModel] was created
---------- Networks initialized -------------
[Network G_A] Total number of parameters : 11.378 M
[Network G_B] Total number of parameters : 11.378 M
[Network D_A] Total number of parameters : 2.765 M
[Network D_B] Total number of parameters : 2.765 M
-----------------------------------------------
Setting up a new session...
```

Figure 5: prepare for training

Third, trying *Collection style transfer* on my photos using pre-trained CycleGAN model.


Figure 6: Collection style transfer

Forth, trying *horse to zebra* transformation.

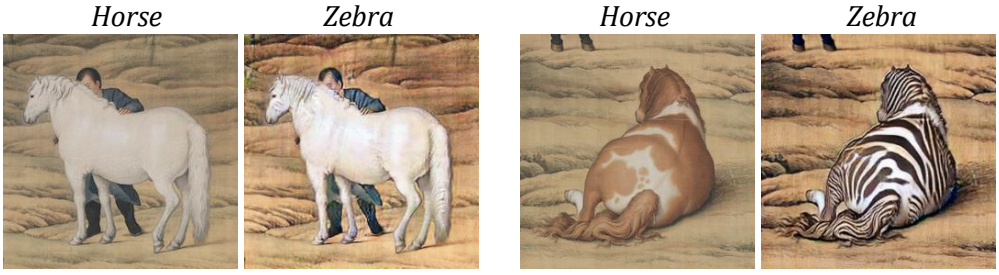


Figure 7: Horse to Zebra transfer

The horse image used here comes from an ancient painting, it’s interesting to find that there is almost no change to the white horse while the brown one is perfectly transferred to zebra.

When there are both white horse and brown horse in the image, only the brown one is successfully transferred to zebra.
This implies that the model can be affected by the color of the horse.
The model performs well in detecting the horse in the image though the author points out that this may fail under certain circumstances.

Figure 8: Horse to Zebra transfer

Fifth, trying *orange to apple* transformation.

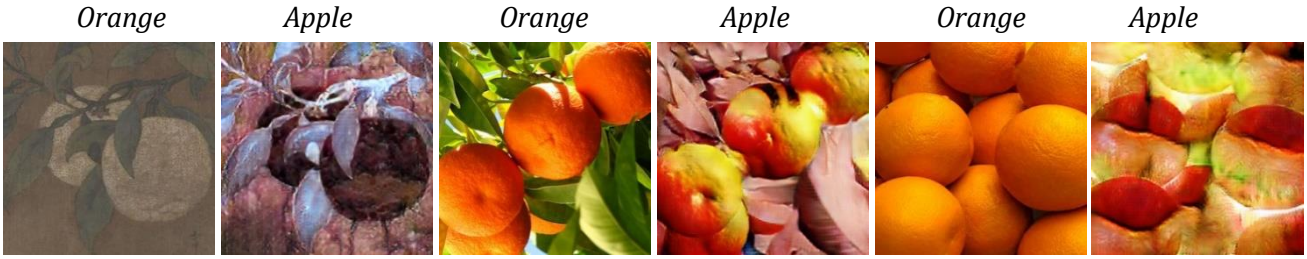


Figure 9: Orange to Apple transfer

The model of *Orange to Apple* transfer performs very bad. Although the model can detect the shape of oranges and distinguish them from leaves, the transfer is to replace the orange's texture with apple's, so the generated apple image looks not real and has a wrong change in the color (leaves in second pair).

## 7 Conclusion

The *CycleGAN* model performs well in Collection style transfer (after all, it's a translation from photos with high details to paintings with low details). The inverse transformation (e.g. *monet2photo*) is much more difficult to deceive people's eyes (the author says *CycleGAN* can fool participants on around a quarter of trial!). In the Object transfer which emphasizes on object's texture, the *Horse to Zebra* transfer is good (though affected by color. The horse and zebra's texture are not complex and covers large area, so the transfer is not too hard). The *Orange to Apple* transfer is bad, given the object is small and the texture replacement requires more accuracy.

There are some other applications, which are not implemented in my lab, and due to the limitation of my computer there is no successful local model training.

The key ideas of this paper and the *CycleGAN* model:
First, it only requires **unpaired image sets** for learning.
Second, it is **based on GAN** with the needs of adversarial loss as its objective.
Third, it uses **Cycle Consistency** to prevent the learned mappings $G, F$ from contradicting each other.

*The End*