Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Wang Hao,515021910060 Wang Xiao,515021910141 Qiang Zhiwen,515030910367

Abstract—In this paper, we study the problem of image to image translation with unpaired image sets. $CycleGAN$ is introduced to solve this problem. Compared with ordinary adversarial networks, more constraint is added in $CycleGAN$, the cycle-consistency. With the help of it, implementation with pretty good results can be achieve. Furthermore, we make some changes to the $CycleGAN$ to try to improve the image translation results in some occasions We add some skips to improve the stability of the adversarial network.

Index Terms—Image translation, Adversarial networks, Cycle-consistency.

1 Introduction

Image translation is a very popular topic nowadays, especially with the help of neural networks. Most of the image translation is about one-to-one mapping, which in other words, pairwise images are used to train the model. However, such pairwise images are not always available. And most of the time, we are only provided with groups of pictures of similar feature, (for instance, a group of paintings from the same painter or a group of images that depict apples) and what we can do with such groups of images? The unpaired image translation is our goal. With the help of the approach we present below, we can achieve collection style transfer, object transfiguration, season transfer, etc. Such achievements are based on a group of images obtaining certain features from another group of images.

In the paper which presents the above idea, its goal is to implement an unpaired training. This training is to learn a mapping $G : X \longrightarrow Y$ (X, Y are two groups of images respectively, and X is the source domain, Y is the target domain) such that in ideal case, the generated image $G(X)$ has the same distribution of the images in Y. And in this way, the generated image has some features of the images in Y.

The basic idea of implementation is to use adversarial networks. The generator network tries to produce images from the desired distribution, and the discriminator network needs to distinguish whether an image is a real image from the target domain or a fake image generated from the source domain. And in this way they compete with each other, finally we will get a desired generator. Since there is large capacity, a network can map the same set of input images to any random permutation of images in the target domain, any of the learned mappings are able to satisfy such distribution, which in this way we can hardly get the desired output images. And there are occasions when the mode collapses, which means all input images map to the same output image and the optimization process fails to make progress. As a result, more constraint needs to be added. The cycle-consistency, if we transform from source distribution to target and then back again to source distribution, we should get samples from our source distribution. And with the help of this, successful translation can be achieved.

The overall goal of our project is to capture special characteristics of one image collection and figuring out how these characteristics could be translated into another image collection, all in the absence of any paired training examples. And to do this, we try to use the $CycleGAN$ method which is proposed in the paper. However, the proposed method is quite primitive, there can be many efforts to improve the performance such as adding some skips into the training, etc. Our final result should be several generators which can translate some specific types of images into another type such as convert horse to zebra, etc.

Below are roughly all the goals we need to achieve in our project:

- Read and fully understand the paper.
- Set up the tensorflow working environment.
- Select a small dataset to train a sample generator to test the method.
- Use a server to train a complete generator to test the performance of the proposed method.
- Go through the source code and try to make some improvements to faster the training process or make some improvements to overcome some failure cases mentioned in the paper.

1.1 Related Work

The principle of Generative Adversarial Networks (GANs) [1] is to create an adversarial loss that forces the generated images to be indistinguishable from real images. This idea has been adopted by many recent methods like text2image, image inpainting and future prediction. But all this work require paired training examples, which refer to the concept of Image-to-Image Translation.

The idea of image-to-image translation is to employ a nonparametric texture model[2] on a single input-output training image pair. Some similar ideas have been applied to various tasks such as generating photographs from sketches or from attribute and semantic layouts, obvious these kind of methods still need he paired training examples.

Another way to perform image-to-image translation is neural style transfer[3], which synthesizes a novel image by combining the content of one image with the style of another image, but this kind of single sample transfer methods do

not perform well in the transfiguration of painting to photo, etc.

As a result of the shortcomings of image-to-image translation, several other methods tackle the unpaired Image-to-Image Translation. Like CoGAN[4] and cross-modal scene networks[5] use a weight-sharing strategy to learn a common representation across domains. Another line of concurrent work encourages the input and output to share certain content features even though they may differ in style. Nevertheless, unlike the above approaches, our formulation does not rely on any task-specific, predefined similarity function between the input and output, nor do we assume that the input and output have to lie in the same low-dimensional embedding space.

Another concept related to the idea of unpaired Image-to-Image translation is cycle consistency, which use transitivity as a way to regularize structured data. Recently, higher-order cycle consistency has been applied by many researches[6]. Zhou et al[7] and Godard et al use a cycle consistency loss as a way of using transitivity to supervise CNN training. Yi et al[8] independently uses a similar objective for unpaired image-to-image translation.

1.2 Proposed Work in Assigned Research Paper

The basic module is as follows.

- Domain: X, Y
- Two mappings: $G : X \longrightarrow Y \&\& F : Y \longrightarrow X$
- Two adversarial discriminators: D_X and D_Y
- Adversarial Loss
  We apply adversarial losses to both mapping functions. For the mapping function $G : X \longrightarrow Y$ and its discriminator D_Y, the adversarial loss is:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[logD_Y(y)]$$
$$+\mathbb{E}_{x \sim p_{data}(x)}[log(1 - D_Y(G(x))]$$

  where G tries to generate images $G(x)$ that look similar to images from domain Y , while D_Y aims to distinguish between translated samples $G(x)$ and real samples y. G aims to minimize this objective against an adversary D that tries to maximize it. We introduce a similar adversarial loss for the mapping function $F : Y \longrightarrow X$ and its discriminator D_X as well.
- Cycle Consistency Loss
  Adversarial losses alone cannot guarantee that the learned function can map an individual input x_i to a desired output y_i. To make sure the mapping function works in a meaningful way, we argue that the learned mapping functions should be cycle-consistent: for each image x from domain X, the image translation cycle should be able to bring x back to the original image, i.e.$x \longrightarrow G(x) \longrightarrow F(G(x)) \approx x$. Similarly, there is also: $y \longrightarrow F(y) \longrightarrow G(F(y)) \approx y$. We can incentivize this behavior using a cycle consistency loss:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{y \sim p_{data}(x)}[||F(G(x)) - x||_1]$$
$$+\mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1].$$

- Full Objective

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y)$$
$$+\mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda\mathcal{L}_{cyc}(G, F),$$

  where λ controls the relative importance of the two objectives. We aim to solve:

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

The model works by taking an input image from domain D_X which is fed to our first Generator X. The new generated image is then fed to another Generator Y which converts it back into an image $Cyclic_X$ from our original domain D_X. And as we discussed in above paragraph, this output image must be close to original input image to define a meaningful mapping that is absent in unpaired dataset.(see **Figure 1**)

Two inputs are fed into each discriminator (one is original image corresponding to that domain and other is the generated image via a generator) and the job of discriminator is to distinguish between them, so that discriminator is able to defy the adversary and reject images generated by it. While the generator would like to make sure that these images get accepted by the discriminator, so it will try to generate images which are very close to original images in Class D_Y. (Basically, this is what GAN does).(see **Figure 2**)

1.3 Limitations of Proposed Work

There are two main limitations. First of all, it is not very easy to get a steady adversarial network, that is to say, even if the input dataset and neural network parameters are the same, the final generator can be different because of the initial state for the network contains random factors. And **Figure3** illustrate it (two generators use the same parameters of the neural network and dataset, but there are minor differences in the output pictures.)

Besides, this method does good on texture translation. But it is not very easy to achieve shape translation. **Figure 4** describes it, our goal is to achieve the orange and apple translation. But it is not easy to change the shape of apples petiole.

2 Proposed Research Work

Since $CycleGAN$ uses unpaired datasets to train models and all parameters of CNN are randomly initialized, it can sometimes be very unstable. In other words, it may fail to extract and translate proper features of a given photo. One failure case we observed during training is that there appeared high contrast colors between input and generated images, e.g. white became to black. One possible explain is that as a result of the diversity of both datasets, there might be both white and black photos of the same object, and the initialized CNN may happen to transform a white photo into a black one which happens to get a close match in the other domain. Thus, the loss of the discriminator can be very small so that the generator believes it did the right job and keeps it that way.

What we want here is actually not to lose too much characters of the original photo and only change those parts

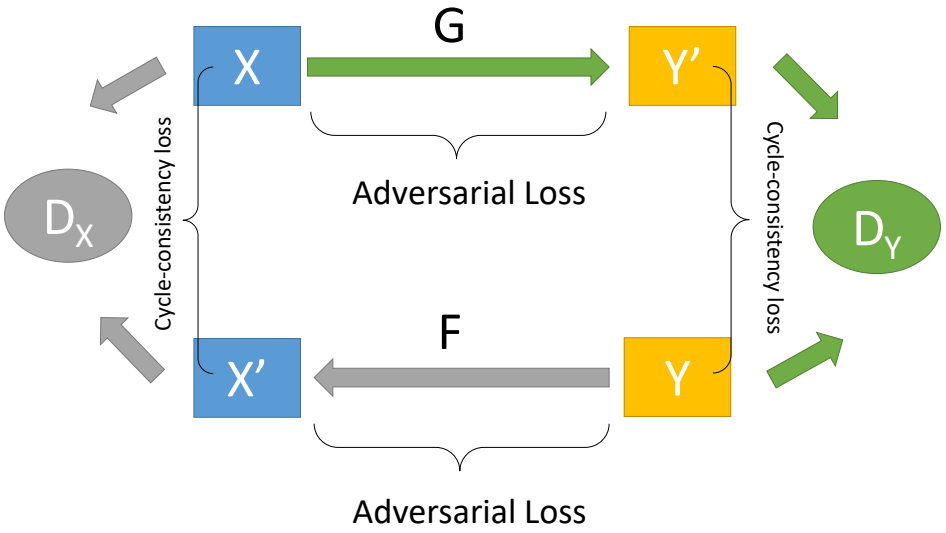

Fig. 1. An illustration of the full objective.

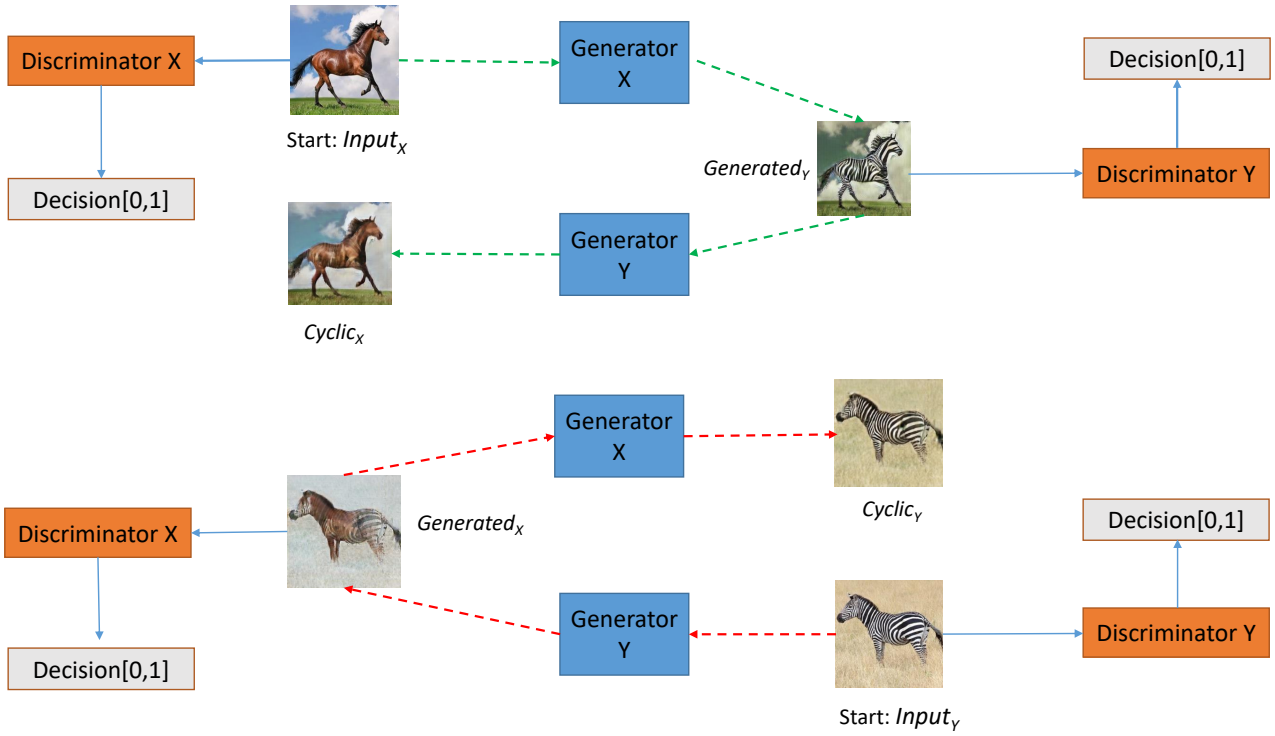

Fig. 2. Pipeline for the $CycleGAN$.

Fig. 3. Example of unstable result

Fig. 4. Example of poor texture translation

which need to. To achieve that, we managed to add a "skip" option in the original $CycleGAN$. In detail, if "skip" is set to be true, then before the generator gets the final output, we add CNN's final layer's output with the original photo first. And after that, we use the final tanh activation function to get the generated photo. As you can see, it works kind of like the residual network, and thanks to the short cut between the input and output, we can make sure that the generated photo keeps as many features as possible. During test time, we found this method can indeed serve our purpose.

Furthermore, as training steps go larger and larger, the model will be more and more stable. Thus, we can gradually decrease the weight of the original input in the addition step to faster the process and get models as good as the native $CycleGAN$.

3 METHODOLOGY

The flow chart of the whole project has been illustrated in the proposed work part, so is each part's function. Here, we will explain the detailed implementation of each step.

First thing to say, this project is implemented with Tensorflow and Python.

3.1 Generator

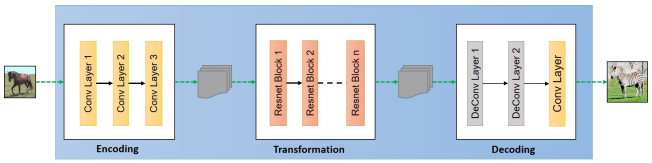

Fig. 5. Generator for the $CycleGAN$

Since two generators work in a same way, I will take Generator X as an example.

Generator X(see **Figure 5**) consists of 3 parts: encoder, transformer and decoder.

Following are the parameters we have used for the mode.

$$batch\_size = 1 \# batch\_size$$

$$pool\_size = 50 \# pool\_size$$

Encoder:

For simplicity, the input has been fixed as $[256, 256, 3]$. The first step is extracting features from an image through a convolution network. A convolution network takes an image as input, and then uses filters to move over the image to extract out features by one stride each time. For the first layer, each filter is $7 * 7$ and the stride is $1 * 1$. The activation function is relu. Outputs shape will be $[256, 256, 64]$ (by padding). Then, the output of the first layer will be passed to the second layer, and so on.

The hyperparameters of the second layer is:$\#filters = 128, filter = 3 * 3, stride = 2 * 2, activation functionisrelu, output = [128, 128, 128]$.

The hyperparameters of the third layer is: $\#filters = 256, filter = 3 * 3, stride = 2 * 2, activation functionisrelu, output = [64, 64, 256]$.

So, the input of the transformer will be $[64, 64, 256]$, meaning 256 features vectors of size $64 * 64$ each.

Transformer:

This part is to transform feature vectors of an image from domain X to domain Y. To do so, we have used 6 layers of resnet blocks.

Resnet block can be summarized as follows, which contains a direct channel from the input to the output and two convolution layers. The reason for using resnet network(see

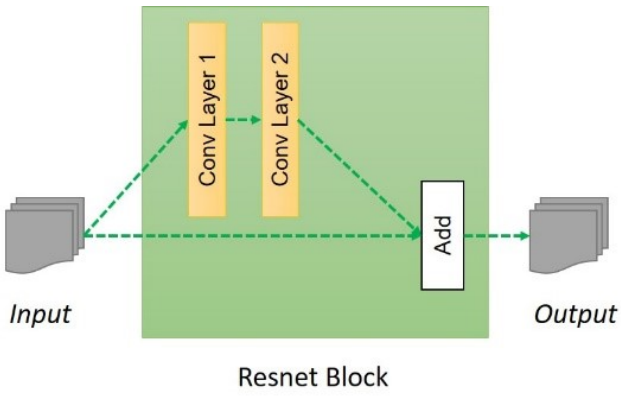

Fig. 6. Resnet for the $CycleGAN$

Figure6) is to ensure properties of input of previous layers are available for later layers, so that the output do not deviate much from original input, otherwise the characteristics of original images will not be retained in the output like the size and shape of the object and results can be very abrupt.

All filters size is $3 * 3$ and stride is $1 * 1$. Output is $[64, 64, 256]$.

Decoder:

Decoding step is exactly opposite of encoding step. We will rebuild an image from those feature vectors gained before, and this is done by applying three deconvolution layers which uses reversed parameters of encoding step.

3.2 Discriminator

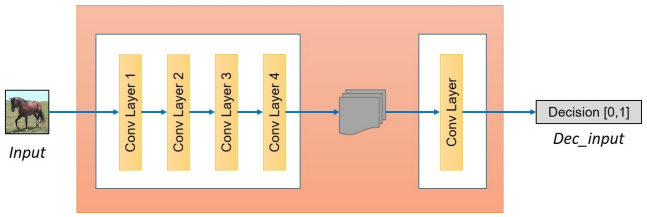

Fig. 7. Discriminator for the $CycleGAN$

Discriminator(see **Figure 7**) also needs to extract features, so its basically a convolution network.

Next step is deciding whether these features belongs to that particular category or not. For that we add a final convolution layer which produces a 1-dimensional output.

Hyperparameters:

$$\#filters + filtersize + stride + activation function$$

$$Layer1 : 64, 4 * 4, 2, leakyrelu(slope = 0.2)$$

$$Layer2 : 128, 4 * 4, 2, leakyrelu(slope = 0.2)$$

$$Layer3 : 256, 4 * 4, 2, leakyrelu(slope = 0.2)$$

$$Layer4 : 512, 4 * 4, 2, leakyrelu(slope = 0.2)$$

$$decision = 1, 4 * 4, 1, leastsquaregan$$

All filters are initialized by $tf.random\_normal\_initializer()$ function with $Gaussian's\, mean = 0$ & $standard\, deviation = 0.02$

3.3 Loss

We use least square gan loss, to meet our goal, the loss function must satisfy:

Discriminator X must be trained such that recommendation for images from domain X must be as close to 1. So, discriminator X would like to minimize $(DiscriminatorX(x) - 1)^2$.

Also, since discriminator X should be able to distinguish generated and original images, it should predict 0 for images produced by the generator X. Discriminator X would like to minimize $(DiscriminatorX(GeneratorY \rightarrow X(y)))^2$.

Moreover, generator X should eventually be able to fool the discriminator Y about the authenticity of its generated images. This can be done if the recommendation by discriminator Y for the generated images is as close to 1 as possible. So, generator X would like to minimize $(DiscriminatorY(GeneratorX \rightarrow Y(x)) - 1)^2$.

And the most important cycle loss captures that we are able to get the image back using another generator and thus the difference between the original image and the cyclic image which is $[F(G(x)) - x] + [G(F(y)) - y]$ should be as small as possible.

PS: weights of both cycle losses are 10 compared with other losses.

4 Implementation

We have implemented our $CycleGAN$ with the help of Googles Tensorflow framework. By using several tensorflows API which are implemented in python, we successfully constructed the generator and discriminator according to the methodology part. Datasets we used to train our model come from $BerkeleysEECS$ website. Besides, we have also tried to use the $Wiki\_Crop$ dataset to train a face to face translation model.

5 Results

We have successfully achieved $CycleGAN$. And using corresponding method mentioned above to re-implement it. **Figure 11**are part of what we have achieved.

So as we can see, this adversarial network can achieve good results of what we have expected.

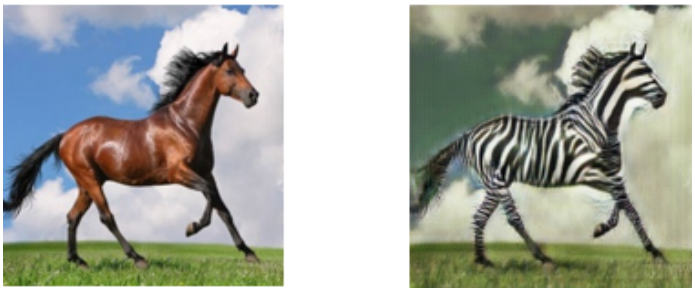

Fig. 8. A horse → zebra translation

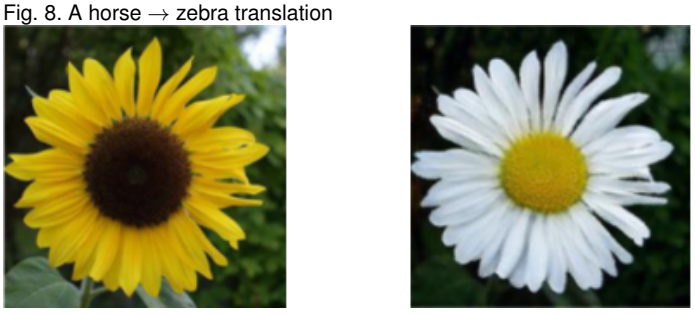

Fig. 9. A daisy → sunflower translation

Fig. 10. A painting → photo translation

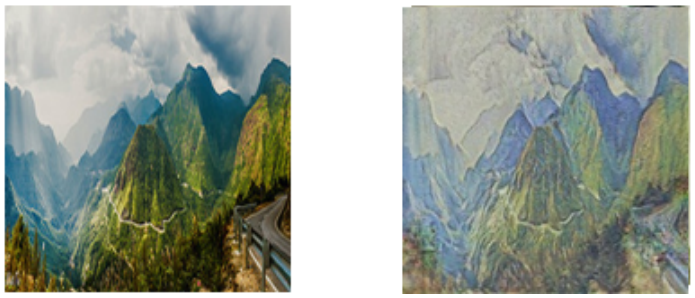

Fig. 11. A photo → painting translation

6 Conclusion and Future Direction

In this paper, we study the problem of image to image translation with unpaired image input. And we use $CycleGAN$ which is presented in the original paper to reimplement it. Furthermore, we make some changes to the $CycleGAN$ in order to try to improve the image translation results. And we successfully achieve image translation with unpaired image sets. With the improvements we made, the output images seem to be of better quality to some extent.

Since there are many limitations to this model as mentioned above. Our future work will mainly focus on improving the stability of the GAN network and shortening the time for training.

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Ben- gio. Generative adversarial nets. In NIPS, 2014. 2, 3, 4, 7
[2] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In SIGGRAPH, pages 327340. ACM, 2001. 2, 3
[3] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. CVPR, 2016. 3, 8, 9, 15, 16
[4] M.-Y. Liu and O. Tuzel. Coupled generative adversar- ial networks. In NIPS, pages 469477, 2016. 3, 6, 7
[5] Y. Aytar, L. Castrejon, C. Vondrick, H. Pirsiavash, and A. Torralba. Cross-modal scene networks. arXiv preprint arXiv:1610.09003, 2016. 3
[6] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement op- tical flow. In ECCV, pages 438451. Springer, 2010. 3
[7] T. Zhou, P. Krahenbuhl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d- guided cycle consistency. In CVPR, pages 117126, 2016. 2, 3
[8] Z. Yi, H. Zhang, T. Gong, Tan, and M. Gong. Dual- gan: Unsupervised dual learning for image-to-image translation. In ICCV, 2017. 3