# CycleGAN with Better Cycles

**Tongzhou Wang**
The Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA, 94704
simon.l@berkeley.edu
24438425

**Yihan Lin**
The Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA, 94704
linyh@berkeley.edu
3032523196

## Abstract

CycleGAN provides a framework to train image-to-image translation with unpaired datasets using cycle consistency loss [4]. While results are great in many applications, the pixel level cycle consistency can potentially be problematic and causes unrealistic images in certain cases. In this project, we propose three simple modifications to cycle consistency, and show that such an approach achieves better results with less artifacts.

## 1 Introduction

Image-to-image translation generates some of the most fascinating and exciting results in computer vision. Using generative adversarial networks (GANs), pix2pix gained a huge amount of popularity on Twitter with its edges to cats translation [2]. Trained using unpaired data, Cycle-Generative Adversarial Networks (CycleGAN) achieves amazing translation results in many cases where paired data is impossible, such as Monet paintings to photos, zebras to horses, etc. Image-to-image translation is also important in the task of domain adaptation. For safety reasons, robots are often trained in simulated environment and using synthesized data. In order for such trained robots to behave well in real life scenarios, one possible approach is to translate real life data, e.g., images, into data similar to what they are trained with using image-to-image translation techniques.

In this paper, we identify some existing problems with the CycleGAN framework specifically with respect to the cycle consistency loss, and several modifications aiming to solve the issues.

## 2 CycleGAN

CycleGAN is a framework that learns image-to-image translation from unpaired datasets [4]. Its architecture contains two generators and two discriminators as shown in Figure 1. The two image domains of interest are denoted as $X$ and $Y$. Generator $G$ takes an image from $X$ as input and tries to generate a realistic image in $Y$ that tricks discriminator $D_X$. Similarly, generator $F$ generates image in reverse direction and tries to trick discriminator $D_Y$.

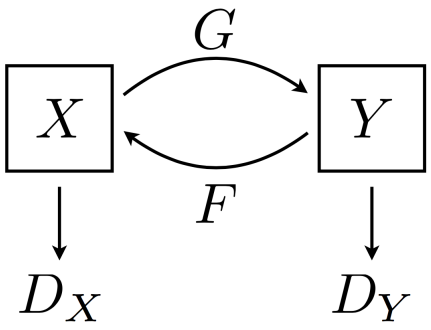



Figure 1: CycleGAN architecture.[1]

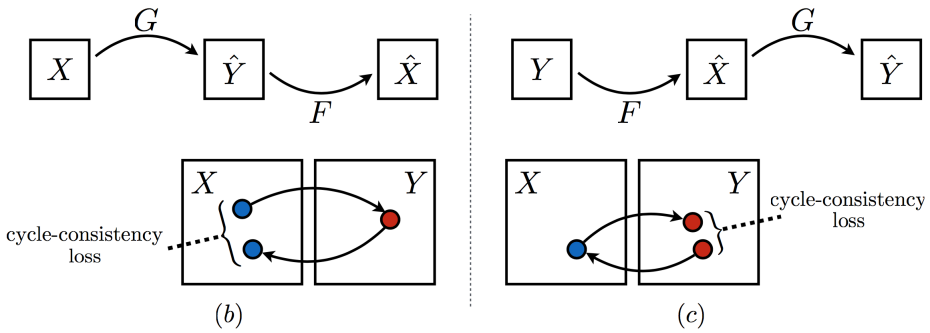



Figure 2: Cycle consistency.[2]

Similar to usual GAN settings, the discriminators encourage generators to output realistic images using the GAN loss:

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

In order to train with unpaired data, CycleGAN proposes the notion of cycle consistency. It asserts that given a real image $x$ in $X$, if the two generators $G$ and $F$ are good, mapping it to domain $Y$ and then back to $X$ should give back the original image $x$, i.e., $x \to G(x) \to F(G(x)) \approx x$. Similarly, the backward direction should also have $y \to F(y) \to G(F(y)) \approx y$. Figure 2 graphically shows the idea of cycle consistency, which is enforced through the following cycle consistency loss:

$$\mathcal{L}_{\text{cyc}}(G, F, X) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \tag{2}$$

Putting the two losses together, the full objective for CycleGAN is:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \tag{3}$$

$$+ \lambda \mathcal{L}_{\text{cyc}}(G, F, X) + \lambda \mathcal{L}_{\text{cyc}}(F, G, Y) \tag{4}$$

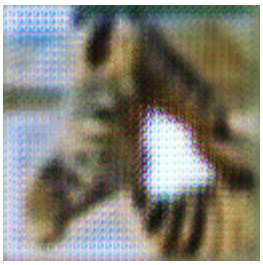
### 2.1 Effects of cycle consistency

On a high level, cycle consistency encourages generators to avoid unnecessary changes and thus to generate images that share structural similarity with inputs.

[1] Figure adapted from [4].

[2] Figure adapted from [4].

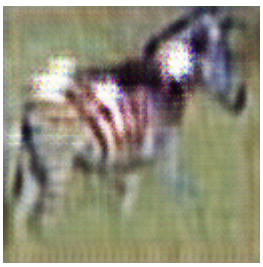
(a) Real zebra image.



(b) Generated horse image.

Figure 3: Generators quickly learn near-identity mapping at training epoch 3.

(a) Real zebra image.



(b) Generated horse image.

Figure 4: Generators quickly learn color mapping at training epoch 10.

**Guide training** During experiments, we observe that the cycle consistency guides training by quickly driving generators to output images similar to inputs with simple color mappings. As shown in Figure 3, the generator learns a near-identity mapping as early as training epoch 3 out of total 200. In Figure 4, we see that the generator learns to map yellow grass to green grass in zebra $\rightarrow$ horse translation at training epoch 10 out of 200. Upon inspecting the training dataset, we see that images from horse dataset generally have greener grass than those from zebra dataset. Because color mappings are often easily reversible, cycle consistency loss and GAN loss are particularly good at jointly guiding generators to output images with correct colors, which are crucial to whether they look realistic.

**Regularize** Cycle consistency can also be viewed as a form of regularization. By enforcing cycle consistency, CycleGAN framework prevents generators from excessive hallucinations and mode collapse, both of which will cause unnecessary loss of information and thus increase in cycle consistency loss.

**Unrealistic artifacts** Great as it is, cycle consistency is not without issues. Cycle consistency is enforced at pixel level. It assumes a one-to-one mapping between the two image domains and no information loss during translation even when loss is necessary. Consider the zebra $\rightarrow$ horse translation shown in Figure 5, the generator cannot completely remove zebra texture because of cycle consistency. In the shoe $\rightarrow$ edges translation shown in Figure 6, also due to cycle consistency, color of the boot must be (potentially unperceptibly) somehow encoded in the result edges image, which causes unwanted artifacts.

## 3 Better cycle consistency

Cycle consistency is great. But as shown above in Section 2.1, it is sometimes too strong an assumption and causes undesired results. In this section, we propose three changes to cycle consistency aiming to solve the aforementioned issues.

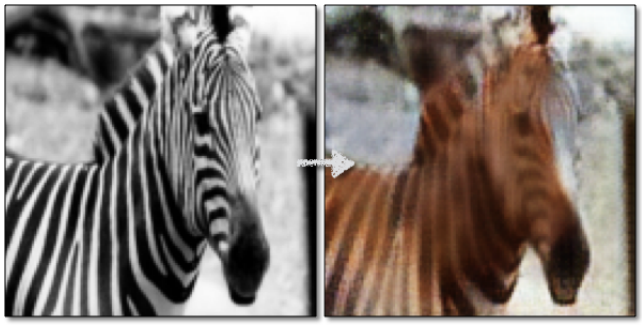


Figure 5: Unrealistic texture due to cycle consistency.

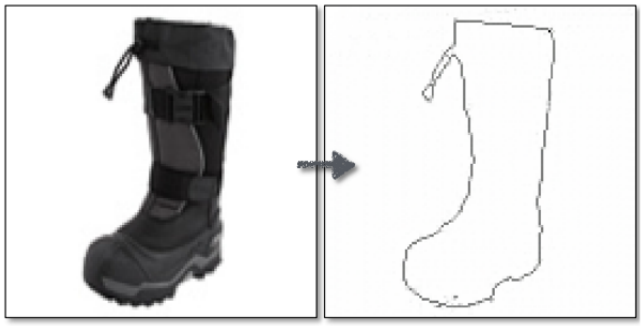


Figure 6: Generator must encoded color information in edges due to cycle consistency.[3]

### 3.1 Cycle consistency on discriminator CNN feature level

Information is almost always lost in the translation process. Instead of expecting CycleGAN to recover the original exact image pixels, we should better only require that it recover the general structures. For example, in the zebra-to-horse translation, as long as the reconstructed zebra image has realistic zebra stripes, be it horizontal or vertical, identical to original ones or not, the cycle should be considered consistent. We enforce this weaker notion of cycle consistency by including an $L1$ loss on the CNN features extracted by the corresponding discriminator, which hopefully has learned good features on the image domain we are interested in. Specifically, the modified cycle consistency loss for one direction is now defined as a linear combination of CNN feature level and pixel level consistency losses:

$$\tilde{\mathcal{L}}_{\text{cyc}}(G, F, D_X, X, \gamma) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\gamma \| f_{D_X}(F(G(x))) - f_{D_X}(x) \|_1 + (1-\gamma) \| F(G(x)) - x \|_1], \quad (5)$$

where $f_{D_{(\cdot)}}$ is the feature extractor using last layer of $D_{(\cdot)}$, and $\gamma \in [0, 1]$ indicates the ratio between discriminator CNN feature level and pixel level loss. This approach is similar to the deep perceptual similarities metric (DeePSiM) in GAN setting introduced in [1]. DeePSiM also uses a combination of pixel level distance and CNN feature level distance, where the CNN can be fixed, such as VGGNet, or trained, such as generator or discriminator.

In practice, we observe that during training, it is best that $\gamma$ vary with epoch. In particular, $\gamma$ should start low because discriminator features are not good at beginning, and gradually linearly increase to a high value close but not equal to 1 because some fraction of pixel level consistency is needed to prevent excessive hallucination on background and unrelated objects in the images.

### 3.2 Cycle consistency weight decay

As shown in Section 2.1, cycle consistency loss helps stabilizing training a lot in early stages but becomes an obstacle towards realistic images in later stages. We propose to gradually decay the weight of cycle consistency loss $\lambda$ as training progress. However, we should still make sure that $\lambda$ is not decayed to 0 so that generators won't become unconstrained and go completely wild.

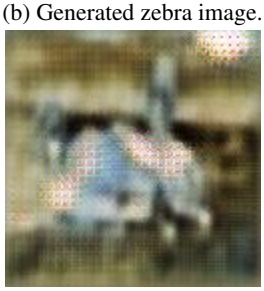
[3] Figure adapted from [4].

(a) Real horse image.

(b) Generated zebra image.

(c) Reconstructed horse image.

(d) Real horse image.



(e) Generated zebra image.

(f) Reconstructed horse image.

Figure 7: Color inversion effect observed at training epoch 6.

### 3.3 Weight cycle consistency by quality of generated image

Sometimes early in training, we observe cases where generated image is very unrealistic and cycle consistency doesn't even make sense. For instance, in Figure 7, the two generators, instead of trying to generate realistic images, learns color inversion mapping so that they can collectively decrease cycle consistency loss. In fact, once stuck in such local modes, the generators are unlikely to escape due to the cycle consistency loss. Therefore, enforcing cycle consistency on cycles where generated images are not realistic actually hinders training. To solve this issue, we propose to weight cycle consistency loss by the quality of generated images, which we obtain using the discriminators' outputs. Adding this change to Equation 5, we have the new cycle consistency loss:

$$\begin{aligned}&\tilde{\mathcal{L}}_{\text{cyc}}(G, F, D_X, X, \gamma) = \\ &\quad \mathbb{E}_{x \sim p_{\text{data}}(x)} \Big[ D_X(x) \Big( \gamma \| f_{D_X}(F(G(x))) - f_{D_X}(x) \|_1 + (1-\gamma) \| F(G(x)) - x \|_1 \Big) \Big]\end{aligned} \tag{6}$$

In particular, such a change dynamically balances GAN loss and cycle consistency loss early in training. It essentially urges the generators to first focus on outputting realistic image and to worry about cycle consistency later.

It is worthwhile to note that gradient of this loss should not be backward propagated to $D_{(\cdot)}$ because cycle consistency is a constraint only on generators.

### 3.4 Full objective

Putting the above proposed changes together, the full objective at epoch $t$ it:

$$\mathcal{L}(G, F, D_X, D_Y, t) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \tag{7}$$

$$+ \lambda_t \tilde{\mathcal{L}}_{\text{cyc}}(G, F, D_X, X, \gamma_t) + \lambda_t \tilde{\mathcal{L}}_{\text{cyc}}(F, G, D_Y, Y, \gamma_t), \tag{8}$$

where we suggest $\lambda_t$ to linearly decrease to a small value and $\gamma_t$ to linearly increase to a value close to 1.

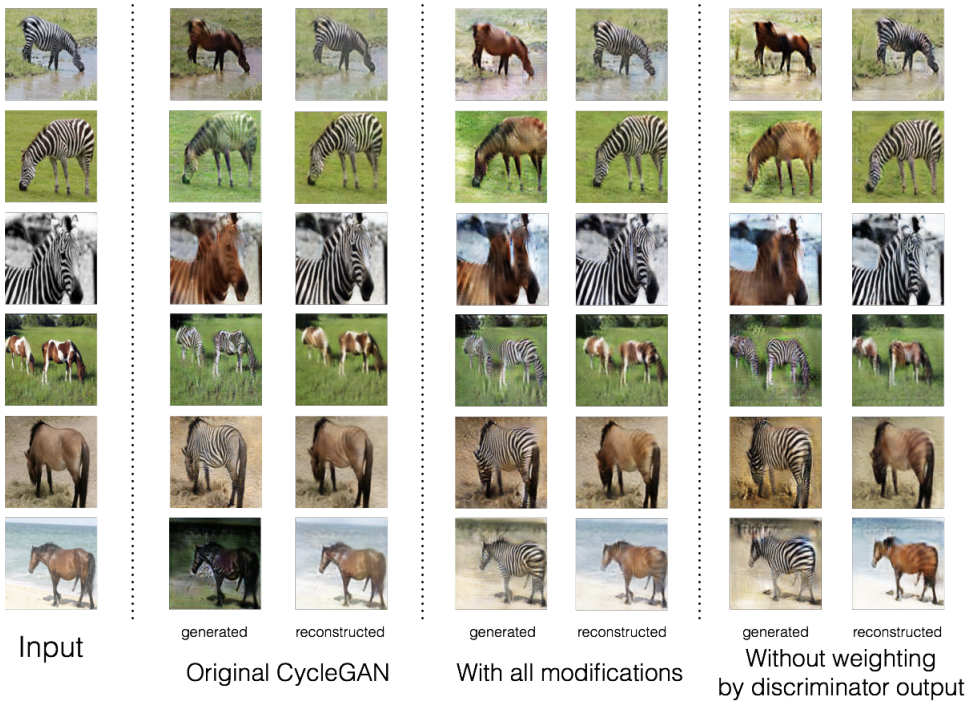



Figure 8: Comparison among original CycleGAN, CycleGAN with proposed modifications, and CycleGAN with proposed modifications except weighting cycle consistency by discriminator output on `horse2zebra` dataset. These images are hand picked from training set.


Figure 9: Failure case on `horse2zebra` dataset.

## 4 Experiments

We compare the proposed approach with original CycleGAN on `horse2zebra` dataset. In both experiments, we train with constant learning rate $0.0002$ for $100$ iterations and linearly decaying learning rate to $0$ for another $100$ iterations.

Figure 8 shows the comparison on training set among original CycleGAN, CycleGAN with proposed modifications, and CycleGAN with proposed modifications except weighting cycle consistency by discriminator output.

As we can see, our proposed changes achieve better results with less artifacts than original CycleGAN. Specifically, although the reconstructed images are not as close to the original inputs, the generated outputs generally look more realistic. However, weighting cycle consistency by discriminator output doesn't contribute much to the result quality. We suspect that this is due to that discriminators are jointly trained with generators. During entire training, the discriminator outputs mostly stays around a constant value, which we observe to be about $0.3$. Therefore, we believe that using pretrained discriminators will make this modification actually have positive effect. In Section 5, we will discuss the potential approach of pretraining and fine-tuning discriminators in greater depth.

Nonetheless, we still found some cases where our modifications relax the cycle consistency may be too much so that it allows the generators to hallucinate unwanted artifacts, such as zebra textures around the horse in Figure 9. We think better parameter tuning will alleviate such issue.

## 5 Future work

While our proposed approach achieves better results, there are still many exciting directions for us to explore, for example, how to tune the parameters, how to solve the one-to-many mapping problem. In this section, we describe several directions that we should investigate in future.

**Parameter tuning** With the proposed changes, training CycleGAN now has many more parameters to tune, e.g., when and how to change $\lambda_t$ and $\gamma_t$, which discriminator to use as feature extractor, etc. During experiments, we found that the result quality is very sensitive to different parameters. Due to time limit, we are only able to try a few combinations. Experimenting for better parameters is definitely an important and future work direction.

**Pretrain and fine-tune discriminators** Discriminators play an important role in two of three our proposed changes. However, since the discriminators are trained together with generators, they don't offer much in early stages. We believe that pretrained discriminators, either trained with CycleGAN task or initialized with pretrained CNN weights like AlexNet, along with fine-tuning, should give a considerable improvement over our current results. Moreover, because CycleGAN uses least-squares GAN, we in theory can over-train the discriminators and need not to worry about the diminishing gradients problem [3].

**One-to-many mapping with stochastic input** Another exciting direction is to feed stochastic input to the generators so that they are essentially one-to-many mappings. However, it remains unknown how to include stochastic input into the architecture while still properly enforcing cycle consistency or some other training guidance and regularization. We attempted adding a noise channel to input and modifying generators to output an extra channel of data, which is encouraged to have same distribution as the input noise channel with the help of discriminators. However, we weren't able to achieve comparable results with the original CycleGAN within same period of training, likely due to a full channel of noise being too much randomness.

**Generators with latent variables** We can consider the two image domains in CycleGAN translation task as sharing a latent space, and each generator as first mapping to latent space and then mapping to target domain. To incorporate this idea into CycleGAN framework, we can pick a certain layer in generator architecture as outputting latent variable, and think of the two generators as a pair of "mutual encoders/decoders" in the sense that encoding and decoding between a certain image domain and latent space are done in two different generators. Then, we can potentially enforce other notions of consistency, such as latent variable consistency and shorter cycle consistency (image domain $\rightarrow$ latent space $\rightarrow$ image domain).

**Single discriminator for both directions** Since the two discriminators do classification on different domains, we can potentially replace them with one network that classifies among three classes, two image domains and fake images. Therefore, the discriminator sees data from both domains. Because the generators almost always are near-identity mapping at early stage of training, such a discriminator may better drive the generators towards right directions.

## 6 Conclusion

In this project, we identify and analyze several issues in CycleGAN framework caused by the cycle consistency assumption. In order to solve these issues, we propose changes to cycle consistency: adding $L1$ loss on the CNN features extracted by the corresponding discriminator, decaying the weight of cycle consistency loss as training progresses, and weighting cycle consistency loss by the quality of generated images. Training on the `horse2zebra` dataset, We show that experiment results on `horse2zebra` improve obviously. Last but not least, we point out several exciting future work directions to investigate.

## References


[1] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems*, pages 658–666, 2016.

[2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[3] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016.

[4] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.