Final: CycleGAN and DualGAN on Artistic Image-to-Image Translation

Maxwell Omdal

May 2019

Abstract

Image-to-image translation, often referred to as style transfer or image domain mapping is a technique that often uses generative adversarial networks to recreate one image in the "style" of another. In this paper, I explore different techniques for style transfer and outline the advantages of CycleGAN and DualGAN. Then, training a model with an implementation of each system is detailed. The paper focuses on qualitative results framed for the specific scenario of a studio looking to reduce their work load with incorporation of style transfer.

1 Image-to-image translation: An introduction

Image-to-image translation is a problem in computer science that has recently had major breakthroughs. The goal of an image-to-image translation is to take an image of one domain, and render it in the "style" of another domain. The problem I propose is as follows:

> An art studio does not have the resources to employ several artists, but they have customers who are always requesting paintings of different styles from a reference image. How can the studio produce new stylistically pleasing images from the given reference image without hiring artists experienced in different art styles?

One of the greatest challenges with such a translation is producing an image that is of the same context as the input. A common problem with methods for translation is an output image that does not seem to correspond to the original image. Since our problem is about customized artwork, it is of significant value to produce an output that corresponds to the input for the sake of the customer.

The proposed problem will be treated as a thought problem instead of a real-life issue to solve. The purpose of framing this as so is to create a clear connection to the applications of the outlined generative adversarial networks and outline the merits and issues that come along with several different techniques for image-to-image translation.

2 Generative adversarial networks

A generative adversarial network (GAN), is a system for generating mappings between two domains. Two neural networks play distinct roles in a zero sum game. One net acts as the "generator" sometimes colloquially referred to as the "artist". The other net judges how well the generated content matches a particular domain, and is often referred to as the "generator". The two adversarial nets work against each other [3]. The artist tries to fool the judge with output that passes for a genuine member of a class or domain, while the judge tries to reject the output of the artists.

Applications of GANs are continually expanding, but many practical uses have already been discovered. Some uses include generating realistic training data to improve classification performance [8], image-to-image translation [5, 9, 11, 8], increasing the resolution of images [6], and synthesizing images from text descriptions [10].

In this paper, I focus on a special implementation of GAN that uses more than two networks to generate forward and backward mappings. Thus an image can be translated between domains in either direction. There are many such GAN systems that have such a system. In this paper, I will compare different GAN implementations and describe why I think CycleGAN is the best implementation for our proposed problem.

3 Cycle-consistent adversarial networks

In Zhu et al. [11], a way to expand upon the pix2pix framework designed for isola et al. [5] is described. A CycleGAN works similarly, however, it also attempts to minimize what is called the cycle-consistency loss. It also avoids the need for paired image examples in its training set . So let there be two functions, $G(x)$ and $F(y)$. G takes as input x, and generates a result, $\hat{y}$. F takes as input y, and generates a result $\hat{x}$, such that $F(G(x)) \approx x$. Thus, four neural networks (two pairs of discriminator and generator) are required to train two models. The cycle loss is calulated as follows:

$$L_{cyc}(G, F) = E_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + E_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1] \quad (1)$$

The original implementation of CycleGan is from Zhu et al.'s [11] paper, and is written in both Torch and PyTorch. Since, there have been many implementations, including a Keras implementation [8]. For this paper, the original Author's PyTorch version is used on Google's Cloud Platform with a Tesla K80 24GB GPU, and our models are trained over 200 epochs.

4 DualGAN

DualGAN [9] is the most similar to CycleGAN. In Yi et al., the concept of reconstruction loss of a twice translated image is proposed. This method is described as more or less identical to the cycle-consistency loss in CycleGAN. Like

CycleGAN, DualGAN is also able to perform well with small, unpaired, unlabeled data sets, reliant on the reconstruction loss. Aside from the calculation of their loss functions, DualGAN and CycleGAN are very similar and perform with similar results. You can tell their similarity by viewing the fundamental shared calculation in the differences between reconstruction/cycles:

$$l_A^d(u,v) = D_A(G_A(u,z)) - D_A(v) \tag{2}$$

$$l_B^d(u,v) = D_B(G_B(v,z^{'})) - D_B(u) \tag{3}$$

If you segment the cycle loss calculation, you will see these are essentially the same. Both loss functions find the difference between the original and the reconstruction for each cycle.

5 Misrepresenting Losses

CycleGAN and DualGAN both rely on cycled losses to base the generator's performance. This performs quite well at ensuring the model does not produce output that defers too much from the input. However, the final reconstruction $F(G(x))$ can be deceptively accurate. Outlined in Chu et al. [2], a rendered reconstruction looks so similar to the real one because each cycle hides information about the original image in the output. This becomes abundantly obvious when looking at maps to satellite reconstructions. How could the model know the color of building roofs if not through steganography? This creates two problems: it makes it harder for us to understand quantitatively how well the model is performing, and it tells us there is noise in our image that only exists because it makes the image reconstruction decipherable by the model. For this reason, we consider the appeal of the output image to an individual more so than the calculated reconstruction error.

6 Similar Techniques

6.1 Conditional Adversarial Networks

Generative adversarial networks have been shown to be quite powerful with content generation. Image generation is a common representation of its performance and capabilities. A conditional adversarial network requires a pairing of images A and A', where A' represents the desired translation to make up its training data set [5]. In Isola et al. [5], the example of satellite maps is used, because it is easy to find satellite images and generated road maps to use as A and A'.

6.2 Image Analogies

An Image Analogy [4] is a method for translating an image from a pair of images with the applied filter of interest. So given three images, A, A' and B,

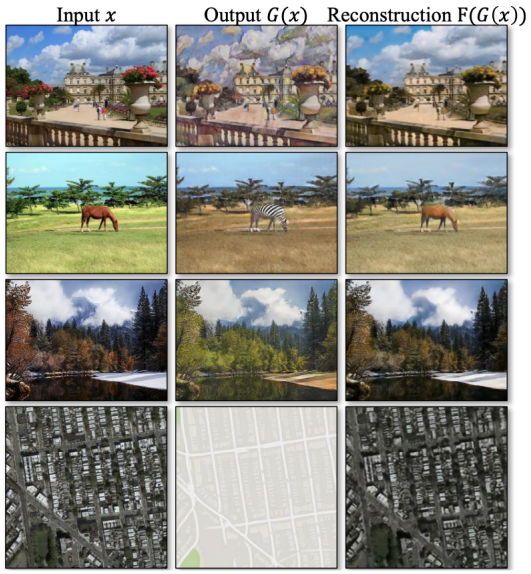

Figure 1: [11] Reconstructions of images from original. When mapping from images of higher entropy to images of lower entropy, the cycle loss ensures that information describing the original image is hidden in the output. This is how the reconstruction is able to so accurately render the original input. Originally made by Zhu et al.

where A' is the result of filtering A, we would have the following relationship: $A : A' :: B : B'$. Unlike other outlined techniques, image analogies are not adversarial networks. Image analogies have very compelling results in a wide array of applications that would be similar to that of CycleGAN and/or DualGAN including texture synthesis, super-resolution, and artistic style transfer. Image Analogies were one of the first techniques for style transfer and has many distinct advantages such as its requirement for only one paired set of images.

6.3 StarGAN

All the discussed systems thus far have been for one-to-one mappings from one domain to another. StarGAN [1] is unique in that it can generate many domain mappings from a single generator. This could be applicable to our proposed problem because it would reduce training time, and we can assume that ultimately we will have to produce many domain mappings for outputting images of different styles. However, such a generator would require that it be re-trained any time a new mapping is desired. This makes it less flexible for our current application. Another advantage of CycleGAN previously mentioned is its ability to preserve the context, which is essential for solving the proposed

problem. StarGAN requires an unpaired data set, which is important because a paired data set for multiple domains would be extremely difficult to gather.

7 Comparison

One of the advantages of using an adversarial network like CycleGAN is that it is more flexible in its applications. The trade-off is the time it takes to get such a model tuned and trained. Unlike an adversarial network, an image analogy would only require two reference images. Since these two reference images must be a paired mapping between the two domains, this makes it difficult or often impossible to obtain in many scenarios. If we want to map a photograph to a Van Gogh self-portrait, we would have to resurrect Van Gogh! The applications of both adversarial networks and image analogies are similar, but the data requirements can be different. The most important difference for our application is the data set of unpaired images. As described in Hertzman et al. [4] a human with experience would be needed to select and prepare the pair of images, A and A′. The same is true for a conditional adversarial network, except instead of one image pairing, a large set of pairs would be needed for training. With our goal in mind, neither of these are an ideal solution to our problem. Since we require that images of different styles be generated without the need for a professional to create an image of a certain style, it is necessary to find a system that does not require paired data sets.

StarGAN also could have advantages because a one to many mapping would allow output of several styles easily. However, unlike CycleGAN and DualGAN there are no checks on the reconstruction loss, so a model might fail to generate results that strongly correlate to the input. Future work could be done to see how to combine cycle-consistency loss with StarGAN. A blending of these systems might prove useful for training a domain mapping, such as portraits, which would be common for an art studio, to many domains.

For these reasons, an unpaired adversarial network such as DualGAN or CycleGAN is the best option for our art studio. This can be used to generate images of different styles as needed, and given a set of images with the desired style, we can transfer that style to any other image.

One of the biggest advantages of a cycle-consistent adversarial network is its ability to discriminate what looks like it was derived from the input image and what does not. Unlike most GANs, CycleGAN or DualGAN insists that the output corresponds to the input by measuring the cycle-consistency loss. If a model were to output a translation, the model should also be able to take that translation and output a result that represents the original input. The difference between the reverse-translation and the original input is what is described as the cycle-consistency loss. By aiming to minimize the cycle-consistency with each epoch, we keep the subject of our image, while changing the style.

Because of the similarity between DualGAN and CycleGAN, both will be trained on the same unpaired images. We define the relationship A, B where A and B each describe a specific domain. In the implementation section, we will

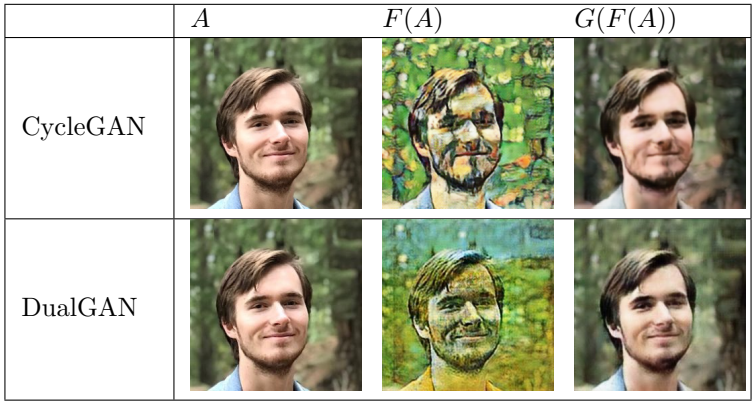

Figure 2: Example of domain transfer for CycleGAN and DualGAN. Both were trained on same data sets and the results compared on the same test set. This is a subset of the images tested upon that represent the overall quantitative results.

explore models where domain A is a collection of portraits and domain B is a collection of Van Gogh paintings. We define the mapping between the images as so:

$$(A) = B, g(B) = A \tag{4}$$

To understand the reconstruction loss, or cycle-consistency, it then follows from (3) that

$$g(f(A)) \approx A, f(g(B)) \approx B \tag{5}$$

8 Implementation

For training, both the CycleGAN and DualGAN models were trained on the same data sets. For the training data set, a subset of 400 images from the CelebA data set [7] were mapped to a collection of 750 Van Gogh paintings collected off Flicker by Zhu et al. [11]

I believe that from a qualitative measure, CycleGAN is a much stronger performer based on the images tested. Figure 2 is representative of the two model's performance.

8.1 Setup

Both CycleGAN (https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix) and DualGAN (https://github.com/duxingren14/DualGAN) implementations written by the paper's author's can be found on Github. In this section, I will explain how to use Google Cloud Platform to train both systems and test images on the results.

8.1.1 Google Cloud Platform

Google Cloud Platform is a versatile computing platform that allows us to easily access powerful computing resources. To begin with Google Cloud Platform (GCP), we will need to create two Deep Learning Virtual Machines. One requires the Tensorflow 1.13 framework, and another Pytorch 1.1. There are several implementations available for CycleGAN, and you should feel free to find the one that you feel is best. All are derivatives of the one we will be using.

Once your two virtual machines are set up, and you are able to SSH onto them from your local machine, it is recommended that you increase your GPU quota to have at least 1 GPU for training. Although not required, this will greatly reduce calculation time. I set up my vm's to share a single Tesla K80 GPU, and I trained one model at a time.

Now, we have allocated all the resources necessary, we need to set up our data sets. Both CycleGAN and DualGAN have a set of provided sample data sets, but for our tests, we want to use the same data set on both systems so we can compare the results. I used images provided by the authors of CycleGAN, as well as some images from the CelebA data set [7]. Figure 3 outlines the tree structure required for each system as well as folder names. It is important to consider the images you include in your data set. You want to find images that represent the entirety of your domain as best as possible. So if you want one domain to be facial portraits, you should not exclusively use pictures of one person when trying to represent the portrait of anyone. For our tests, 400-700 images were used in both domains.

With data collected, it is time to train the models. This will take significant time, so it is best to plan accordingly. Even on a powerful compute platform, it will take several days for each to complete. Refer to each system's README file for more information on the necessary command for training and testing. Take note of the options available upon training. For the tests outlined in this paper, we set an epoch count of 200 on a single GPU. It is expected that more epochs will provide diminishing returns on results. Given more time, ideally several tests would be run at different epoch counts to determine which is best considering time, cost and benefit of increased epochs.

Compared to training, testing is very quick. Both systems will save checkpoints of models so you can test for different iterations. For our problem, we will most likely be producing output for one image at a time, so the test directory can be populated with one image in the A directory and any number of images in the B directory. If we were analyzing the performance of the models, more images should be used. When it comes time to test, you may want to adjust options so an reconstruction is not made, because this will not always be desired, especially for our purposes.

8.2 CycleGAN Results and training time

Training a model over 200 epochs with a batch size of 100 took approximately 74 hours to complete. Some results from several data sets are shown in figure

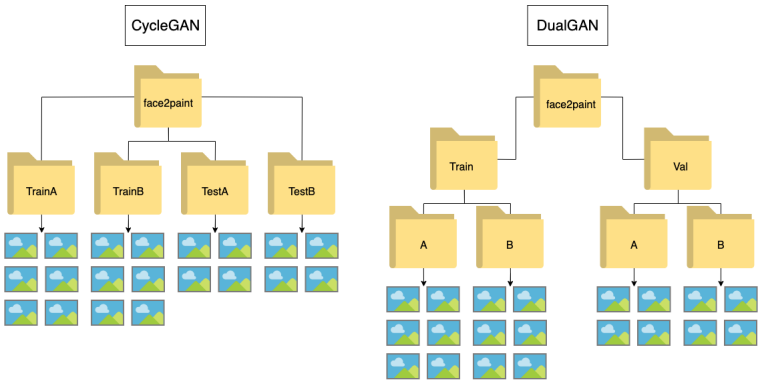

Figure 3: The structure of a data set required by each system

4.

8.3 DualGAN results and training time

Training a model over 200 epochs with a batch size of 100 took approximately 63 hours to complete. Some results from several data sets are shown in figure 4.

8.4 Analysis

Qualitatively, the images produced by CycleGAN are generally more representative of a painting than the ones produced by DualGAN. This statement is made under the premise that the CycleGAN images have better reduced the entropy of a photograph into something that could be done by a painter. Colors are more even, there is less noise, and you can clearly see simulated "brush strokes" in the outputted images.

To measure the images qualitatively, we will use the mean squared error equation and the structured similarity index to calculate the difference between the model's reconstructed image and their original. For images, structured similarity is often considered superior in calculating the perceived difference of images, while mean squared error is a more straightforward averaged pixel difference. The equations are defined as:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[x(i,j) - y(i,j)]^2] \tag{6}$$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{7}$$

Where μ_x is the average of the image x^*, μ_x is the average of y^*, σ_x^2 is the variance of x^*, σ_y^2 is the variance of y^*, and σ_{xy} is the covariance of x and y.

Figure 4: Example of domain transfer from *landscapes* $\rightarrow$ *Monetpaintings*. Both were trained on same data sets and the results compared on the same test set. This is a subset of the images tested upon that represent the overall quantitative results.

Using a simple python script, we can compare a large set of test images and return their averaged differences. The results for varying data sets are shown in figure 5. MSE is calculated as a constant while $SSIM$ is a percent calculation, where 100% identifies the images as identical. Though DualGAN performs better with ultimate reconstruction error of the domain transfer $F(G(A))$, for MSE, and they perform nearly identically in $SSIM$, CycleGAN is superior in producing believable fakes. It has also been mentioned in section (5) that such a measure is not an accurate way of determining the performance of the model. Although an essential characteristic of the algorithms that encourages proper domain transfer, the reconstruction/cycle loss may not properly represent an algorithm's performance because in an attempt to reduce this loss, such systems can create inefficient techniques for "hiding" information in images to create precisely accurate reconstructions. Thus, for these two model, mean squared error may not be a great performance measure, but this should make you wonder

	Portrait $\rightarrow VanGogh$	Landscape $\rightarrow Monet$
CycleGAN (MSE)	852.177	450.103
DualGAN (MSE)	522.102	370.400
CycleGAN (SSIM)	0.883679	0.789487
DualGAN (SSIM)	0.874689	0.8123349

Figure 5: Averaged mean square error and Structural similarity index for domain reconstruction $F(G(A))$

what counteraction can be taken to prevent such steganography.

9 Conclusion and future work

There are a lot of techniques for image-to-image translation, many of them relying on adversarial networks. I propose an implementation of CycleGAN over DualGAN and other GANs as a solution to generating artistically styled images from a reference image because it uses cycle-consistency loss to regulate context switching when translating over domains, and requires relatively few, unpaired images to train. It should be noted that although DualGAN seems to under-perform quantitatively compared to CycleGAN, it was able to train the same number of epochs in a quarter of the time. This could prove valuable in some use cases.

9.1 Future work

There are some disadvantages to CycleGAN, including its one-to-one domain mapping, which is why further research should be done on what results could be achieved by inter-playing StarGAN with CycleGAN. To improve our output image, research should be done into techniques to reduce static generated by the model that serves no other purpose than to make the reconstruction decipherable from the output. To further improve results, a shotgun technique, where several models are trained for a single purpose, each with differing parameters and data sets can be used to help understand what makes models perform better for the outlined purpose. To better understand the differences in performance between DualGAN and CycleGAN, more tests should be performed with different domain mappings. Several tests with differing epoch counts will also help understand the optimal epochs for training a specific data set. This will take a significant amount of time (weeks to months), but could pay off in the long run. Lastly, it will eventually be necessary to increase the resolution of the outputted photos for them to be used for their artistic value. A problem with this is increasing the resolution exponentially increases the run time, because step will require significantly more calculations. A solution proposed by [11] is to use a tiling method to split large images into smaller segments, individually producing results for each segment. Understanding the output this would create, and a clean way to stitch tiles together is an important next step in this

process.

References

[1] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[2] C. Chu, A. Zhmoginov, and M. Sandler. Cyclegan, a master of steganography. *arXiv preprint arXiv:1712.02950*, 2017.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340. ACM, 2001.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[6] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[7] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 12 2015.

[8] P. Welander, S. Karlsson, and A. Eklund. Generative adversarial networks for image-to-image translation on multi-contrast mr images-a comparison of cyclegan and unit. *arXiv preprint arXiv:1806.07777*, 2018.

[9] Z. Yi, H. Zhang, P. Tan, and M. Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857, 2017.

[10] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5907–5915, 2017.

[11] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.