Recycling call by CycleGAN

Siming, Su (siming@ucsb.edu)

University of California, Santa Barbara

ABSTRACT

In this article, I will present a project on generating images of cities filled with trashes and garbage by using Cycle-Consistent Adversarial Networks (CycleGANs). The aim of this paper is to call for recycling of garbage and avoiding plastic usage. This paper takes two datasets, one is the garbage dataset from Kaggle, the other is European city scene. By transferring the style of garbage field into a city scene or the other way around, CycleGAN has been implemented to generate pictures of city filled with garbage. Such images can be used in environmental posters to emphasize the importance of protecting the environment especially on plastic usage and production. I will also show some examples of posters that I have created based on the result of this CycleGAN.

1. Introduction

In rural areas of developing country, garbage disposal is always a huge problem. Due to the remote instruction, I have stayed in my hometown in China, I have noticed that the garbage sorting system is extremely undeveloped in rural region. Not only this, but local residents also are so lack of recycling knowledge and willingness. Thus, often time, piles of trashes and garbage appear on mostly every corner of the city. I would like to make some progress by raising people’s vigilance of the consequences of casual trash dumping. By utilizing CycleGAN, this tool may help us to visualize the result of how plastics and other garbage can be accumulated and end up transforming the whole city into a bulky pile of garbage. In the next few sections, we will dive into how CycleGAN works, and how it can help us to visualize the future of the garbage-dumping city.

My home-city’s trash scene

These pictures are from www.qzwb.com

2. Motivation

As shown in the pictures above, most of garbage piles made mostly by plastics. Regarding plastic and garbage problem, garbage soring seems to be a workable solution to it. However, this is far from enough. First of all, the production of plastics is out of humans’ control. According to National Geographic, until 2016, the production of plastics has been increased exponentially from 2.3 million to 448 million since it was invented [8]. Second of all, the lifespan of plastic is extremely long. According to the BIO-TECH environmental organization, the following is the list of lifespans of different materials:

- 450 years for plastic waster bottle
- 500 years for disposable diapers
- 450 years for 6-pack collar

- 5000 more years for extruded polystyrene foam

These four materials are extremely common in daily life, and yet they take a long time for degradation [10]. Thirdly, in addition to the two facts above, we, as humans, do not do recycling as efficiently as we think. A global analysis of mass-produced plastics has demonstrated that until 2015, for 8300 million tons (Mt) of plastics have been produced, 6300 Mt of plastic wastes have been generated. Among these wastes, only 600 Mt (9%) of them are recycled; 800 Mt(12%) of them are incinerated; the rest of 4900 Mt are being discarded whether into the ocean or became landfills on the ground [9].

3. Overview of CycleGAN

The content of this section is based on the article, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks" by Zhu, Park, Isola, and Efros [6]. In CycleGAN, we have two mapping functions $G: X \rightarrow Y$ and $F: Y \rightarrow X$ to train. If we have picture x from X domain, then, we should have $G(x) \in Y$. Similarly, if we have a picture y from Y domain, then we should have $F(y) \in X$. After we got these two functions, it should serve the function of transforming pictures' styles between two different domains.

3.1 Loss

The current way we train neural networks is through loss functions. Thus, the result will be heavily dependent on which loss functions we apply, and how we apply these loss terms.

3.11 Adversarial Loss

Adversarial loss is one of the three loss functions that CycleGAN has used, and it has been used commonly in GAN particularly [7]. For the discriminator D_Y of $G: X \rightarrow Y$, we have the function formulation as following:

$$L_{GAN}(G, D_Y, X, Y) = \mathrm{E}_{\mathrm{y} \sim \mathrm{P}_{\mathrm{data(y)}}}[\log \mathrm{D_Y}(\mathrm{y})] + \mathrm{E}_{\mathrm{x} \sim \mathrm{P}_{\mathrm{data(x)}}}[1 - \log \mathrm{D_Y}(\mathrm{G(x)})]$$

Here, in this function term, discriminator D_Y is trying to maximize this function by discriminate between G(x) and true y. However, generator G is trying to fool discriminator D_Y by maximizing $log\mathrm{D_Y}(\mathrm{G(x)})$, which minimizes the whole loss function. Thus, we are doing a min-max game, which is the formulation here $min_G max_D L_{GAN}(G, D_Y, X, Y)$.

3.12 Cycle Consistency Loss

This is what makes the CycleGAN being called CycleGAN because this loss function looks like a cycle. Basically, this loss term means that if we apply G map to take x from X to Y, then when we use F to bring it back to X, it should look the same. Thus, the loss function looks like the following:

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

This function punishes every $F(\mathrm{G(x)})$ that does not look like x and $G(\mathrm{F(y)})$ that does not look like y.

3.13 Identity Loss

Originally, the above losses are all losses that the author has, however, they add an additional term that is called identity loss to ensure that the colors between the original image and the result image remain the same. The idea is that when we take $F: Y \rightarrow X$ mapping on image x, it should generate the same image as x. Ie. $F(x) \approx x$ and $G(y) \approx y$. The loss formulation looks like the following:

$$L_{identity}(G,F) = E_{x\sim p_{data}(x)}\left[\left|\left|F(x) - x\right|\right|_1\right] + E_{y\sim p_{data}(y)}\left[\left|\left|G(y) - y\right|\right|_1\right]$$

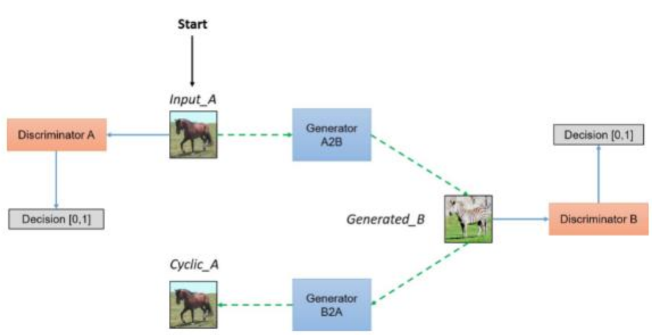
3.14 full loss function for generator

In summary, the full loss function consists of the following three loss functions:

$$L_{total} = L_{GAN}(G, D_Y, X, Y) + \lambda L_{cyc}(G,F) + \beta L_{identity}(G,F)$$

In this paper, I have used all three loss terms to create the outputs I have. Due to the device limitation I have, I only trained for about 31 epochs. However, some decent outputs still can be picked from the outputs.

Network outline



This image is from https://hardikbansal.github.io/CycleGANBlog/

3.15 Discriminator Loss

For discriminator, I use PatchGAN to output classification matrix instead of a single value, so does the original CycleGAN paper [2, 3]. For discriminator, we expect it to output 1 if it encounters a real image, otherwise zero. Thus, we punish the discriminator by taking the MSE loss between the classification matrix and 1 or zero. The following is the formulation that I implanted.

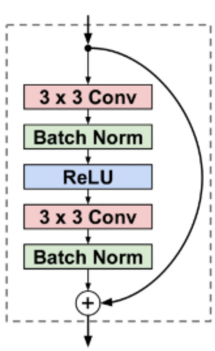
$$L_{disc} = MSE\left(D_y\left(G(x)\right), 0\right) + MSE\left(D_y(y), 1\right)$$

The formula is the same when we train for D_x.

3.2 Architecture

As you may notice that the difference between CycleGAN and other GAN's is that the CycleGAN has two discriminators and two generators. For its generator, the architecture is the following: It uses 7 by 7 up feature-mapping convolution first, then down sample them using two convolutional layers. After the down sampling, they use 9 consecutive residual blocks to help with training in deeper layers [1]. However, just to correct what appears in the picture, I apply instance normalization instead of batch normalization in Residual blocks [4]. Then, they use two up sampling convolutional layers following with one 7 by 7 down feature-mapping convolution layer to bring everything back to the original RGB image. I think an architecture picture will help a lot with the understanding:

Generator's Architecture



Layer	Activation Size
Input	3x256x256
64x7x7 conv, stride 1, pad 3	64x256x256
128x3x3 conv, stride 2, pad 1	128x128x128
256x3x3 conv, stride 2, pad 1	256x64x64
9 consecutive Residual Blocks	256x64x64
128x3x3 convTranspose, stride 2, pad 1, out_pad 1	128x128x128
64x3x3 convTranspose, stride 2, pad 1, out_pad 1	64x256x256
3x7x7 conv, stride 1, pad 3	3x256x256

This image is from https://towardsdatascience.com/overview-of-cyclegan-architecture-and-training-afee31612a2f

For discriminators, it is different from what we have seen from other GAN's discriminator. The discriminator of CycleGAN is not only outputs a number determining how likely the image to be true. However, it outputs a classification matrix. For each entry in this matrix, it classifies overlapping 70 by 70 images. In total then, we have 30 by 30 true or false matrix to represent the classification possibilities of 256 by 256 images [2, 3]. This technique is also called PatchGAN. As usual, image may be salutary for understanding. The following is what the generator architecture looks like:

Discriminator's Architecture

Layer	Activation Size
Input	3x256x256
64x4x4 conv, stride 2, pad 1	64x128x128
128x4x4 conv, stride 2, pad 1	128x64x64
256x4x4 conv, stride 2, pad 1	256x32x32
512x4x4 conv, stride 1, pad 1	512x31x31
1x4x4 conv, stride 1, pad 1	1x30x30

This image is from https://towardsdatascience.com/overview-of-cyclegan-architecture-and-training-afee31612a2f

By using tools above, now we have a full understanding of what CycleGAN is and how it works. It is time to train. I choose 0.0002 as my learning rate with batch size 1 by Adam Optimizer to train the whole networks [5].

4 Related Work

As mentioned before, since my results are rudimentary due to the device limitation. Thus, I would like to share a related work regarding climate whose results are more satisfying. This paper applies CycleGAN to help visualize the climate crisis, flooding. The authors gathered unpaired datasets that consist of flooding image dataset and non-flooding image dataset. By applying and training networks above, this network is enabled to transfer non-flooding images to flooding images and vice versa. Worth noticing here, they have gathered datasets manually such that the datasets were constraints to certain circumstances. By doing this, the CycleGAN networks could quickly learn the relationship between two different styles from different datasets, because image noises have been reduced by a significant amount [6]. The following is the selected results they produced.

Flooding images by CycleGAN

5 Results

After training the network, I have found my favorite pair of photos to build my example poster. I have attached the poster below. Regarding some limitations and problems I found in the datasets, since I have not manually selected the datasets, so the noise of images may negatively influence the performance of this network. For example, since I did not specify the location of the garbage piles in domain X, then for the generated picture $G(x)$, it would change everything to garbage colors including buildings (Image Pair 1), people (Image Pair 2), and clouds (Image Pair 2). In addition to this, because the training itself takes only 30 epochs, so far, my networks only work for pictures of city distant view. If the scene is a close shot, then this network will generate a colorful scene instead of garbage piles. I would improve this results by more careful datasets selection and increasing the number of epochs.

Image Pair 1

Image Pair 2

Example Poster

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, 2016.

[2] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. ECCV, 2016.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Imageto-image translation with conditional adversarial networks. In CVPR, 2017.

[4] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022, 2016.

[5] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[6] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, pp. 2223–2232, 2017.

[7] I. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160, 2016.

[8] Parker, Laura. The world's plastic pollution crisis explained. National Geographic Magazine. www.nationalgeographic.com/environment/article/plastic-pollution.

[9] Geyer, Roland, Jambeck R. Jenna, and Law Lavender Kara. Production, use, and fate of all plastics ever made. DOI: 10.1126/sciadv.1700782. Vol3.Issue7. 2017.

[10] BIO-TECH Environmental organization. Lifespan of Plastic. www.goecopure.com/lifespan-of-plastic.aspx.

Image url:

[10] www.qzwb.com

[11] https://hardikbansal.github.io/CycleGANBlog/

[12] https://towardsdatascience.com/overview-of-cyclegan-architecture-and-training-afee31612a2f