CycleGAN

CS 547 Final Project Report
University of Illinois at Urbana-Champaign

github.com/xyang70/ReimplementCycleGAN

Yao Xiao
yaox9@illinois.edu

Stanley Yang
xyang70@illinois.edu

Sheng Yang
shengy3@illinois.edu

Yen-Wen Lu
ywlu2@illinois.edu

Hsiu-Yao Chang
hsiuyao2@illinois.edu

December 2019

1 Discussion of the paper and related literature

In this project, we follow the paper [2] to implement **Cycle-consistent Generative Adversarial Network** (**CycleGAN**) and test on three different datasets including *monet2photo*, *orange2apple* and *horse2zebra*. The model can translate a input image to another image such as translating a horse to a zebra.
The goal of this model is to learn the mapping $G : X \to Y$, which is highly under-constrained. Therefore, the paper proposed cycle consistency loss $F(G(X)) \approx X$ to enforce the correctness of inverse mapping. Since adding the cycle consistency loss, the objective function should be slightly modified. We discuss the detail of the objective function in the next section.
For the evaluation, in the original paper, the author used "cityscape" dataset and a pre-trained fully convolutional network (FCN8) [1] to quantitatively evaluate how realistic the generated images were. However, the pre-trained model was in Caffe, which is not supported by bluewater. We have tried to evaluate the images on Colab supptorted by Google but the GPU ran out of memory with the default Caffe setting. Therefore, the evaluation was limited by the hardware we have access to. Since using different datasets are without ground truth, we are not able to show the same evaluation table mentioned in the original paper.

2 Model architecture and objective function

Model architecture is as follow:

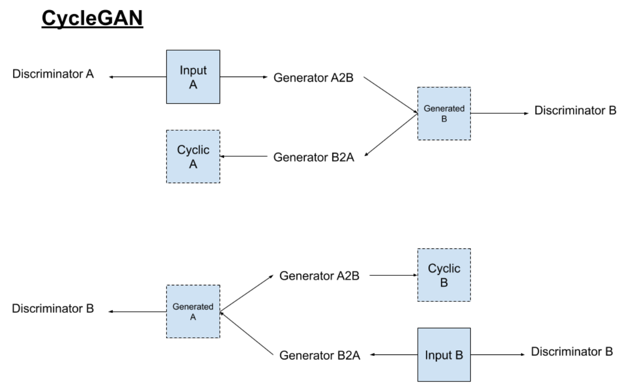
Full Objective is:

$$\begin{aligned}
\mathcal{L}(G, F, D_X, D_Y) &= \mathcal{L}_{\mathcal{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\mathcal{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\mathcal{CYC}}(G, F) && (1)\\
&= \mathbb{E}_{y\sim p_{data}(y)}[(D_Y(y)-1)^2] + \mathbb{E}_{x\sim p_{data}(x)}[D_Y(G(x))^2] && (2)\\
&+ \mathbb{E}_{x\sim p_{data}(x)}[(D_X(x)-1)^2] + \mathbb{E}_{y\sim p_{data}(y)}[D_X(G(y))^2] && (3)\\
&+ \lambda(\mathbb{E}_{x\sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim p_{data}(y)}[\|G(F(y)) - y\|_1]), && (4)
\end{aligned}$$

and from the equations, we could tell that the adversarial loss is essentially MSE loss and cycle consistency loss is L1 loss.

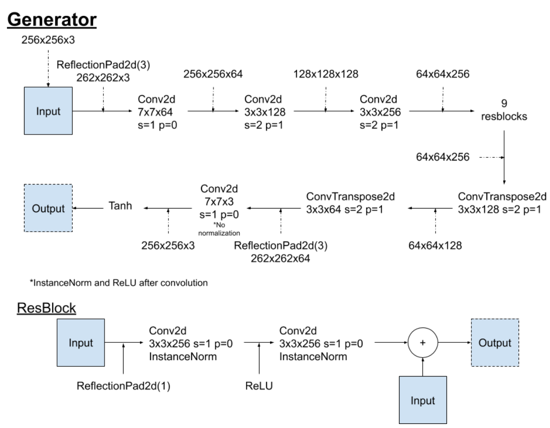
Moreover, as indicated in the paper, we add an extra identity loss term. This serves two purposes. First, it helps preserving the color composition. Second, it ensures input of set A stays the same if it is passed into a generator that takes images from set B as input.

$$\mathcal{L}(G, F) = \mathbb{E}_{x\sim p_{data}(x)}[\|F(x) - x\|_1] + \mathbb{E}_{y\sim p_{data}(y)}[\|G(y)) - y\|_1] \quad (5)$$

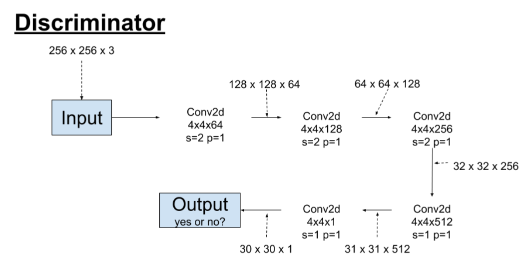


(a) CycleGAN

Figure 1: CycleGAN Architecture



(a) Generator



(b) Discriminator

Figure 2: Generator and Discriminator

3 Training methods

3.1 Optimizations

- **Residual Block**

  Since CycleGAN is a deep neural network in both the discriminator and generator, the paper [2] suggests to use residual block to prevent the vanishing gradient problem. As we know residual block is useful in complex CNN architecture.

- **Reflection Padding**

  Reflection padding pads the image with its outer edge pixel [5]. In this case, it will help the network to train because the pixels are related.

- **Instance Normalization**

  Instance normalization [4] is the same for batch normalization with batch size equals to 1. Instance normalization normalizes one image at once, while batch normalization can normalize many.

- **ReLU**

  ReLU is used in the generator.

- **Leaky ReLU**

  Leaky ReLU is used in the discriminator.

3.2 Data Augmentations

1. **Image Resize With Bicubic Interpolation**

   Images are resized to be 1.12 times larger and are smoothed by bicubic interpolation

2. **Random Crop**

   Images are randomly cropped back to the original input dimensions

3. **Random Flip**

   Images are randomly flipped horizontally

4. **Dataset Normalization**

   Images are normalized to have values in [-1,1] with $\mu = 0$, $\sigma^2 = 1$

4 Hyper-parameters

4.1 Loss weighting

1. **Cycle consistency loss**

   An additional loss to measure the difference between the generated output of the second generator and the original image, and the reverse.

2. **Identity mapping loss**

   The identity mapping loss (identity loss) can regularize the generator to be near an identity mapping when real samples of the target domain are provided. If something already looks like from the target domain, you should not map it into a different image. If identity loss is 0, the images will be inverted in color.

The identity loss was test for 1, 5, 10. During the training, we observed color inversion effect on the output images if we don't apply any identity loss. However, with a large identity loss, the output images might result in almost the same as the input images.

4.2 Epoch number

With a insufficient epoch number, the effect of style transfer was not significant. In addition, the checkered effect on output images was very strong because the discriminator was not able to discriminate the fake and real images. Since GAN doesn't have over-fitting, the number of epoch was chosen to maximize the number of epoch with a reasonable training time.

4.3 Hyper-parameters on datasets

Parameters				
Datasets	Epoches	Lambda Identity	Cycle Consistency	Decay Epoch
Apple2orange	150	5	10	100
Horse2zebra	100	5	10	50
Monet2photo	75	10	10	74
Summer2winter	250	10	10	125

5 Description of the datasets used for training and testing

All four datasets are download from UC Berkely website [6]. The dataset was split into training and test set.

- **Apple2orange**

  This Dataset is download from the official Cycle GAN project site. In training data, there are 995 apple photos and 1019 orange photos. In test data, there are 226 apple photos and 248 orange photos.

- **Horse2zebra**

  939 horse images and 1177 zebra images downloaded from ImageNet using keywords wild horse and zebra [7].

- **Monet2photo**

  1074 monet drawing and 6853 real photos from Flickr. Dataset is downloaded from the official Cycle GAN project site. There are 751 test images for this dataset.

- **Summer2winter**

  1540 summer Yosemite images and 1200 winter Yosemite images were downloaded from the official Cycle GAN project site. The training contains 1231 summer and 962 winter images. The test set contains 309 summer and 238 winter images.

6 Computational cost of training

Since training GAN is time consuming, each of the group members trains different datasets on different harware platforms.
monet2photo The dataset monet2photo is trained on a RTX2060 6GB platform, the computation time with 75 epochs is around 65.6 hours with 4.1 GB of GPU memory.
horse2zebra The dataset is trained on a RTX2060 6GB platform, the computation time with 100 epochs is around 18.3 hours with 4 GB of GPU memory.
orange2apple The dataset is trained on a Nvidia Tesla K80 GPU platform, the computation with 150 epochs is around 25.4 hours with 4GB of GPU memory.
summer2winter The dataset monet2photo is trained on a NVIDIA V100 GPU 16GB platform, the computation time with 250 epochs is around 20 hours with 4 GB of GPU memory.

7 Results

(a) Input A (b) Fake B (c) Input B (d) Fake A

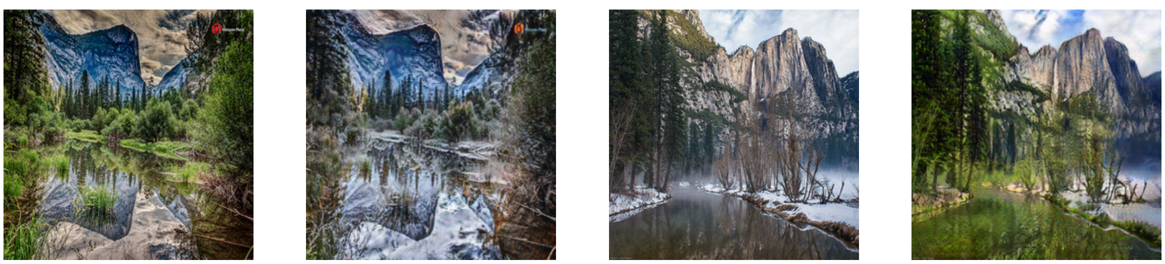
Figure 3: apple2orange

(a) Input A (b) Fake B (c) Input B (d) Fake A

Figure 4: Horse2zebra

(a) Input A (b) Fake B (c) Input B (d) Fake A

Figure 5: Monet2photo



(a) Input A (b) Fake B (c) Input B (d) Fake A

Figure 6: summer2winter

From the result, we may observe that, similar to the paper's result, cyclegan may alter the color, texture and pattern of the image, but not the shape. In apple2orange dataset, the ordering of the fruits, the light and shade, shape and size remain the same. We may also notice that the model manages to transform all apples to oranges. The object at left buttom corner of the 3(a) is not identified as apple, and thus not being transformed to orange. But the tint of the object does shift a little bit. For Horse2zebra dataset, we pick one transform that contains a tiny artifact (4(d)), as one leg of the zebra is not transformed to horse. This may due to the fact that the leg is hard to identify. In Monet2photo dataset, Monet's signature flurry of small strokes of broken color reflects in our examples successfully. In summer2winter dataset, the result is also convincing, as the color palettes mimic the earthy tone of the winter and the green tone in summer.

8 Description of your code

- **Discriminator.py**: define the network of discriminator and its forward pass
- **Generator.py**: define the network of generator and its forward pass

- **Cyclegan.py**: initialize the components of cyclegan, including two generators, two discriminators, optimizers, lr_schedulers and replay buffers; define the forward pass with load() and backward pass with optimize_parameters()
- **ImageDataset.py**: Data utility functions to split the input dateset into train set and test set. In each set of data, data will be splitted into category A and category B.
- **Train_notebook.py**: This is the training framework for CycleGAN. It loads the train set, define the model, train the model, and produce intermediate result for verification. All epoch statistics will be logged.
- **test.py**: The testing framework for CycleGAN. This script will load the test set, load the trained model, and produce test images.

9 Publicly available sources

Our source code is on github: https://github.com/xyang70/ReimplementCycleGAN

There are two existing repositories on github.

1. **Official site**: https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
2. https://github.com/aitorzip/PyTorch-CycleGAN

We followed the 2nd repo's structure but in re-implanted the CycleGAN in a objective-oriented fashion. We implanted the whole CycleGAN model in a class. The optimization for generators and discriminator was separated into functions for better reuse. In addition, we implanted the de-normalization to reconstruct the fake images.

References

[1] Long, Jonathan and Shelhamer, Evan and Darrell, Trevor *Fully convolutional networks for semantic segmentation*. Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440. 2015.

[2] Zhu, Jun-Yan, 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.

[3] Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*. CVPR 2017

[4] Anon. 2018. Instance Normalisation vs Batch normalisation. (January 2018). Retrieved December 7, 2019 from https://stackoverflow.com/questions/45463778/instance-normalisation-vs-batch-normalisation

[5] Anon. r/learnmachinelearning - Can someone explain to me what reflection padding does? Retrieved December 7, 2019 Retrieved from https://www.reddit.com/r/learnmachinelearning/comments/8hetaz/can_someone_ex plain_to_me_what_reflection_padding/

[6] Park. T. (n.d). CycleGAN dataset. Retrieved from https://people.eecs.berkeley.edu/ taesung_park/CycleGAN/datasets/.

[7] Junyanz. junyanz/pytorch-CycleGAN-and-pix2pix. Retrieved December 11, 2019 from https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/blob/master/docs/datasets.md