CycleGAN PyTorch Implementation

Author: Haotian Tang, Shanghai Jiaotong University

This implementation is based on **Jun-Yan Zhu*, Taesung Park*, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", in IEEE International Conference on Computer Vision (ICCV), 2017.**

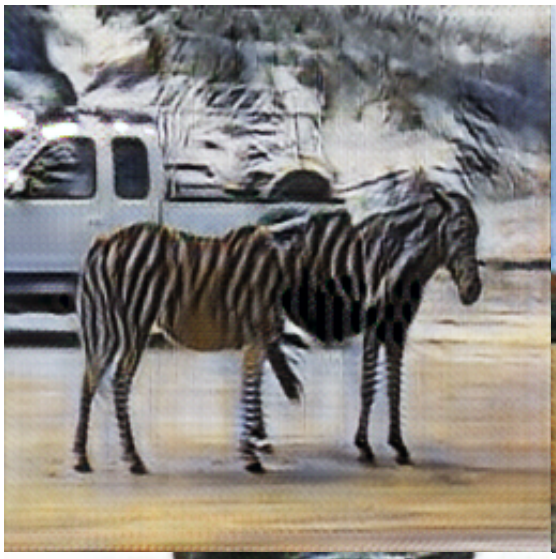
The environment

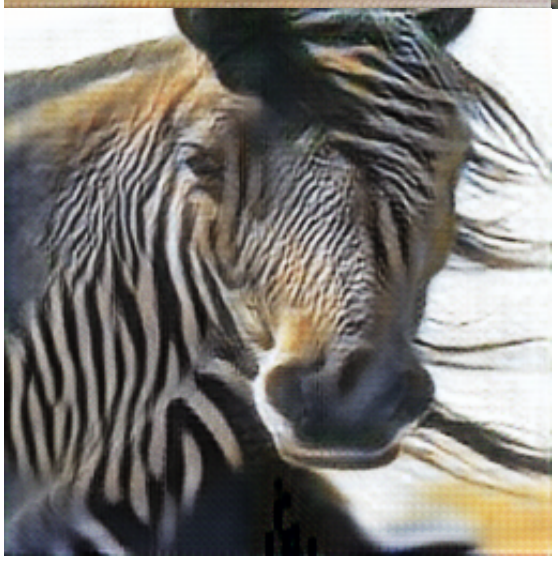
This CycleGAN implementation is written under PyTorch 0.2.0(GPU version), with torchvision toolbox.

We train the model on one GTX 1060 GPU for 200 epoches, the training process takes about 18 hours.

Satisfactory Results(horse2zebra)

The results are achieved after training the model mentioned in Zhu et al's paper for 180 epoches under the same learning policy as is purposed in the paper. **NOTE: Most results are actually NOT satisfying.**





Project Structure

The main directory of this project contains several sub-categories:

- config: The config directory, where training/test settings are stored.
- dataset: Where datasets are stored. The training data and testing data subfolder should be named as 'trainA', 'trainB', 'testA' and 'testB' respectively, which is the same as the naming convention used in Taesung Park's Berkeley FTP.
- log: The log dir which contains two subfolders. The txt folder stores text log, a typical line looks like

  > Epoch 200/200, Cyclic loss 2.6728, loss_G 4.6285, loss_D_A 0.0060, loss_D_B 0.0050.

  The snapshot folder stores the snapshots of our model, which can be easily loaded in PyTorch with torch.load.
- models: The place where key models are stored. Several python files are stored under this folder, including Discriminator.py, Generator.py, loss.py, CycleGAN.py and ResidualBlock.py. The functions of these files are as clear as their names.
- pic: The place where results file are stored. Empty at first.
- utils: Some utility functions. Mainly ImagePool and Dataset.

- Also, there are train.py and test.py under the root.

My implementation uses Zhu et al's PyTorch implementation as reference. But the structure is quite different.

Startup

To use this model, you should install PyTorch 0.2.0 and torchvision (both **GPU** versions) and CUDA8.0, CUDNN 6.1+ first.

Then, clone this github repository, and run the following command to start training:

```
python train.py --data_dir=dataset/horse2zebra --name=horse2zebra20180126
```

Note: data_dir and name are two arguments that are required. The parameter name is used in naming the log files and snapshots.

To test the model, run the following command:

```
python test.py --data_dir=dataset/horse2zebra --name=horse2zebra20180126 --
epoch=200
```

Note: data_dir and name are required. The parameter name should be the same one used in training. Epoch is not required, it indicates which snapshots to use. But if left default, we will use the snapshot taken after 10 epoches.

To Be Done

To be frank, the results of my implementation is not quite satisfactory (comparing to Zhu and Park et al's project page), and I don't have enough time (Restricted by the computational capability of GTX 1060s, which might be replaced by better GPUs in the future) to train other models before handing in this project(e.g.: apple2orange, or style transfer). Also, I didn't explore the possibility of using UNet-Generators and identity mapping loss which are used in Zhu's PyTorch implementation. These are all left for future work.