Akash Govindarajula

Cycle-GAN: Cycle Consistent Style-Transfer

MSAI 449 Project - Advanced topics on Deep Learning

Overview

Beyond the scope of the documentation provided details of the work done, this report is to highlight the approach and changes applied or developed during the training and implementation of the pre-trained models from the Pytorch-based code.

Problems targeted:

i. Reimplementation – initial task and replicating results.
ii. Changes and improvements: Specifically reducing noise and improvement of clarity, and dehazing of generated images.
iii. Model used is mostly the same, with changes done to parameters. Preliminary approach has been to resize the convolutions and achieve this quality. [tests with varying architecture, and a new version has been submitted].

Proposal / Changes

1. Introducing a weight consistency metric in the objective of training section to improve crispness and optimize the training on images with lesser quality. This was demonstrated in presentations and the final render developed in code has a simple numeric value which can be tweaked and altered as needed.

   i. Initial function:

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

   ii. Proposed new function introduced weights, based on this paper. However, in training and final submit, numeric coefficients are used, and hence tuning is easier for studying changes in image generation.

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

   iii. Changes done include varying for following terms in training:
   - --netD (discriminator architecture), --norm, epoch, --gan_mode, lr_policy, etc.

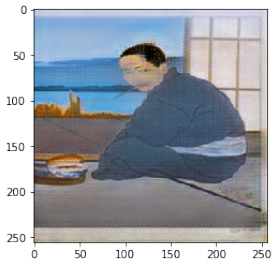
- As these changes are to be accommodated across *util, options* and *train.py* files, a newer and trimmed down approach has been designed and submitted.

2. Newer model submitted by narrowing down options, helpers and util/scripts into train.py, thereby reducing the use of compute power drastically. The files and results have been saved in repo for implementation.

Issues with pre-trained and lesser epochs:

In regard to style-transfer data, the models usually have shown that there is an aspect of hue/tint-based issue in generated images from photographs.

- While image generation at initial stages does generate interesting and valid results, we observe issue of more training needed in regard to color and achieving the properties of the unpaired target image data. This is more prominent in style-transfer of ukiyo-e and vangogh palettes.
- This is handled by using a color related loss-term but modifying it further will end up as a supervised learning model, moving away from the unpaired and unsupervised approach of the model.



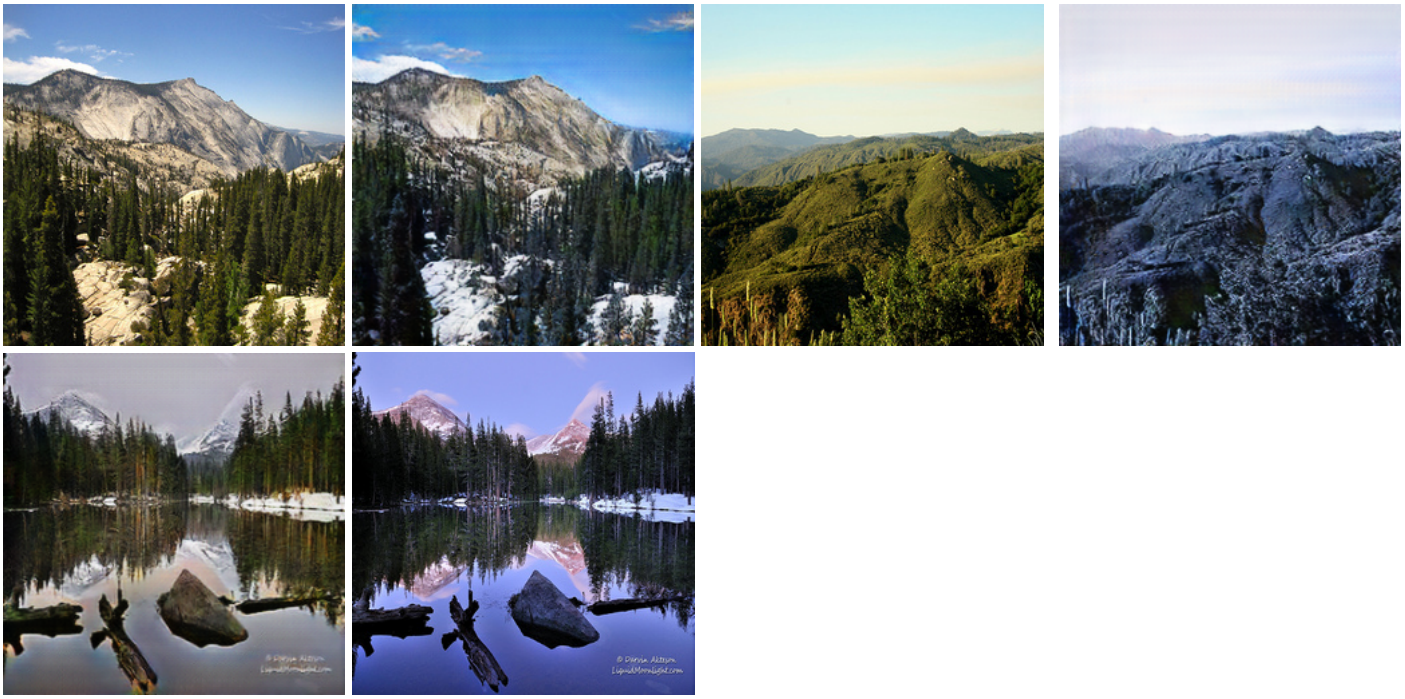
- Another aspect is seen in apple2orange image generation (object transfiguration) where we observe the noise. The newer version has been explored with lr-policy changes and tuning --netG, norm, etc. and improvements are observed.
- Modifying the smaller version of code has been done based on target image generation and recursively running Cycle-GAN again on the generated image as a second iteration. This has refined the images and based on performance (tested for various tweaks/changes done), we fine-tune our baseline model accordingly.

```
Epoch: ( 49) (   21/   30) | Generator Loss:6.25e+00 | Discriminator Loss:5.31e-01
Epoch: ( 49) (   22/   30) | Generator Loss:6.91e+00 | Discriminator Loss:6.18e-01
Epoch: ( 49) (   23/   30) | Generator Loss:6.31e+00 | Discriminator Loss:5.11e-01
Epoch: ( 49) (   24/   30) | Generator Loss:5.88e+00 | Discriminator Loss:4.32e-01
Epoch: ( 49) (   25/   30) | Generator Loss:5.56e+00 | Discriminator Loss:3.21e-01
Epoch: ( 49) (   26/   30) | Generator Loss:5.43e+00 | Discriminator Loss:3.29e-01
Epoch: ( 49) (   27/   30) | Generator Loss:8.67e+00 | Discriminator Loss:2.70e-01
Epoch: ( 49) (   28/   30) | Generator Loss:9.99e+00 | Discriminator Loss:4.74e-01
Epoch: ( 49) (   29/   30) | Generator Loss:5.23e+00 | Discriminator Loss:4.88e-01
Epoch: ( 49) (   30/   30) | Generator Loss:7.73e+00 | Discriminator Loss:3.01e-01
learning rate = 0.0005000
Epoch: ( 50) (    1/   30) | Generator Loss:7.01e+00 | Discriminator Loss:4.59e-01
Epoch: ( 50) (    2/   30) | Generator Loss:6.48e+00 | Discriminator Loss:5.37e-01
Epoch: ( 50) (    3/   30) | Generator Loss:6.30e+00 | Discriminator Loss:4.42e-01
Epoch: ( 50) (    4/   30) | Generator Loss:6.05e+00 | Discriminator Loss:3.03e-01
Epoch: ( 50) (    5/   30) | Generator Loss:5.92e+00 | Discriminator Loss:1.36e-01
Epoch: ( 50) (    6/   30) | Generator Loss:5.48e+00 | Discriminator Loss:2.33e-01
Epoch: ( 50) (    7/   30) | Generator Loss:6.35e+00 | Discriminator Loss:2.95e-01
Epoch: ( 50) (    8/   30) | Generator Loss:6.93e+00 | Discriminator Loss:4.42e-01
Epoch: ( 50) (    9/   30) | Generator Loss:5.64e+00 | Discriminator Loss:4.63e-01
```

Results and Observations

1. Results: These are illustrated in the repo in a separate folder. A few of the best images generated are shown below.

(summer2winter_yosemite)



(horse2zebra)

(monet2photo)

2. New model (simple - cg): The images have been explored for reconstruction and the cyclic training done, which has later improved the quality of desired target image.
   - The best results are seen for instance normalization, with better clarity in images for subsequent model runs.

Submission:

1. GitHub Repo: *github.com/govindarajula/cyc-gan*
2. AMI (Amazon Machine Image) : ‘‘

```
Warning: Permanently added 'ec2-54-162-15-233.compute-1.amazonaws.com,54.162.15.233' (ECDSA) to the list of known hosts.
================================================================================
       __|  __|_  )
       _|  (     /   Deep Learning AMI (Ubuntu 18.04) Version 27.0
      ___|\___|___|
================================================================================

Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-1060-aws x86_64v)

Please use one of the following commands to start the required environment with the framework of your choice:
for MXNet(+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) ____________________________________ source activate mxnet_p36
for MXNet(+Keras2) with Python2 (CUDA 10.1 and Intel MKL-DNN) ____________________________________ source activate mxnet_p27
for MXNet(+AWS Neuron) with Python3 ___________________________________________________ source activate aws_neuron_mxnet_p36
for TensorFlow(+Keras2) with Python3 (CUDA 10.0 and Intel MKL-DNN) ______________________________ source activate tensorflow_p36
for TensorFlow(+Keras2) with Python2 (CUDA 10.0 and Intel MKL-DNN) ______________________________ source activate tensorflow_p27
for Tensorflow(+AWS Neuron) with Python3 ____________________________________________ source activate aws_neuron_tensorflow_p36
for TensorFlow 2(+Keras2) with Python3 (CUDA 10.1 and Intel MKL-DNN) ____________________________ source activate tensorflow2_p36
for TensorFlow 2(+Keras2) with Python2 (CUDA 10.1 and Intel MKL-DNN) ____________________________ source activate tensorflow2_p27
for PyTorch with Python3 (CUDA 10.1 and Intel MKL) ____________________________________________ source activate pytorch_p36
for PyTorch with Python2 (CUDA 10.1 and Intel MKL) ____________________________________________ source activate pytorch_p27
```

3. Documentation and the reports
4. The Amazon Virtual (Machine) image is also available upon an EC2 instance, and the key is in AMI folder of the repo.

   Please mention and it can be shared, but all the files utilized / built have been pushed to GitHub repo.

   - ⇨ Using the AMI file and working on Amazon console is ideal to evaluate in a CPU-based system. If in a GPU-supported computer, local training can be done.
   - ⇨ Public ID (AMI): **ami-0c0752e75eb4d3922**
   - ⇨ Or, connecting to EC2 instance {*shell/cmd*}
     [please inform by email if my AWS instance is needed for access]

   - chmod 400 adl-gan.pem
   - ssh -L localhost:8888:localhost:8888 -i adl-gan.pem ubuntu@<Public DNS>
   - Run the python files for preprocessing, if needed and check dependencies before using Jupyter.