Monet Style Images

Generating Paintings in the Style of Monet through a GAN

Team Name:
Shadow_Hunters

Team Members:
Diya
Yashvi Sharma
Nandini Choudhary

Abstract

Image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs. However, for many tasks, paired training data will not be available. We present an approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples.
Our goal is to learn a mapping $G : X \rightarrow Y$ such that the distribution of images from G(X) is indistinguishable from the distribution Y using an adversarial loss. Because this mapping is highly under-constrained, we couple it with an inverse mapping $F : Y \rightarrow X$ and introduce a cycle consistency loss to enforce $F(G(X)) \approx X$ (and vice versa).

Qualitative results are presented on several tasks where paired training data does not exist, including collection style transfer, object transfiguration, season transfer, photo enhancement, etc. Quantitative comparisons against several prior methods demonstrate the superiority of our approach.

Introduction

Computer Vision (CV) has grown to immense heights in the past few years, and the Generative Adversarial Network (GAN) is one of the most captivating examples of modern deep learning.

The idea is quite intuitive: if we want to generate images that match our dataset, then we can have a generator network make an image, and a discriminator attempts to distinguish between the generated image and the real image. The magic of backpropagation ensures that, in theory, the generator produces better images to the point that the generated and real images are indistinguishable.

We recognize the works of artists through their unique style, such as color choices or brush strokes. The "je ne sais quoi" of artists like Claude Monet can now be imitated with algorithms like Generative Adversarial Networks (GANs). So can (data) science, in the form of GANs, trick classifiers into believing you've created a true Monet?

GANs help us create more realistic images which can further be used to generate new datasets for machine learning models where we have less data, as well as creating high resolution images from blurry low resolution images.

Problem Statement

The problem at hand is to generate 7,000 to 10,000 high quality Monet-style images using a Generative Adversarial Network (GAN) consisting of a generator and discriminator neural network.

Notations:

GAN - Generative Adversarial Network

DCGAN - Deep Convolutional Generative Adversarial Network

Project Objectives

1. To develop a GAN architecture that captures special characteristics of one image collection, in our case Monet paintings, and translates these characteristics into the other image collection, all in the absence of any paired training examples.

2. To generate 7,000 to 10,000 Monet-style images which would trick the classifiers into believing that we have created a true monet using CycleGAN.

3. To build a generator neural network responsible for creating the images, and a discriminator neural network which is trained to distinguish between real and generated images.

Literature Review

We use CycleGAN in this project in order to generate Monet-style images for the input photos. While researching on the GAN architectures we have come across other GANs which can be used for image translation from one domain to another.

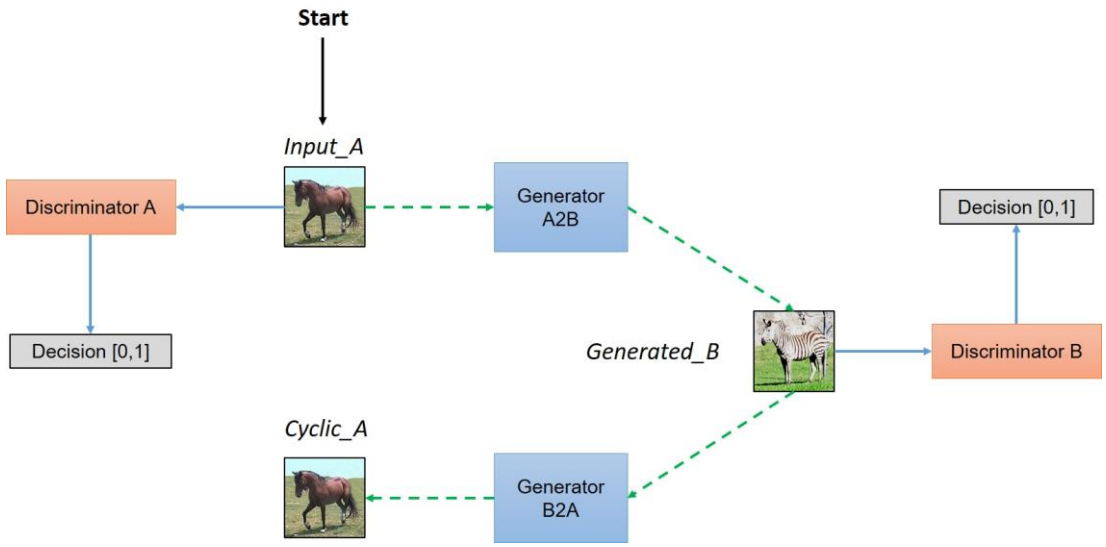
1. Pix2Pix - The Pix2Pix GAN is a general approach for image-to-image translation. It is based on the conditional generative adversarial network, where a target image is generated, conditional on a given input image. In this case, the Pix2Pix GAN changes the loss function so that the generated image is both plausible in the content of the target domain, and is a plausible translation of the input image.

2. DCGAN - DCGAN is one of the popular and successful network design for GAN. It is mainly composed of convolution layers without max pooling or fully connected layers. It uses convolutional stride and transposed convolution for the downsampling and the upsampling.

Methodology and Approach

The first network, the generator, generates new data. The process is, simply put, the reverse of neural networks' classification function. Instead of taking raw data and mapping it to determined outputs in the model, the generator traces back from the output and tries to generate the input data that would map to that output.

The second network, the discriminator, is a classifier DNN. It rates the quality of the results of the generator on a scale of 0 to 1. If the score is too low, the generator corrects the data and resubmits it to the discriminator. The GAN repeats the cycle in super-rapid successions until it can create data that maps to the desired output with a high score.



Loss Function

Sigmoid cross entropy is used to calculate the adversary losses in the discriminator and generator.

1. Discriminator Loss

The calculation of the loss has two components:

- real_loss compares the real image with a matrix of 1 (All Ok).
- generate_loss compares the fake image with a matrix of 0 (All Fake).

So the total_loss is 1/2 of the sum of the real_loss and the generate_loss.

2. Generator Adversary Loss

It takes as input the output of the discriminator.

- For the loss of the generator_monet the function will take the output of the discriminator_monet executed with fake Monet.
- For the loss of the generator_photo the function will take the output of the discriminator_photo executed with fake photo.

The perfect generator will have the discriminator output only ones. Therefore, we compare the generated image with a matrix of 1 to find the loss.

3. Generator Cycle Consistency Loss

To calculate the cycle consistency loss, two terms are calculated:

- Average absolute error for photo, is calculated between Input Photo and Cycle Photo.
- Mean absolute error for monet, is calculated between Input Monet and Cycle Monet.

The cycle error will be the sum of both terms.

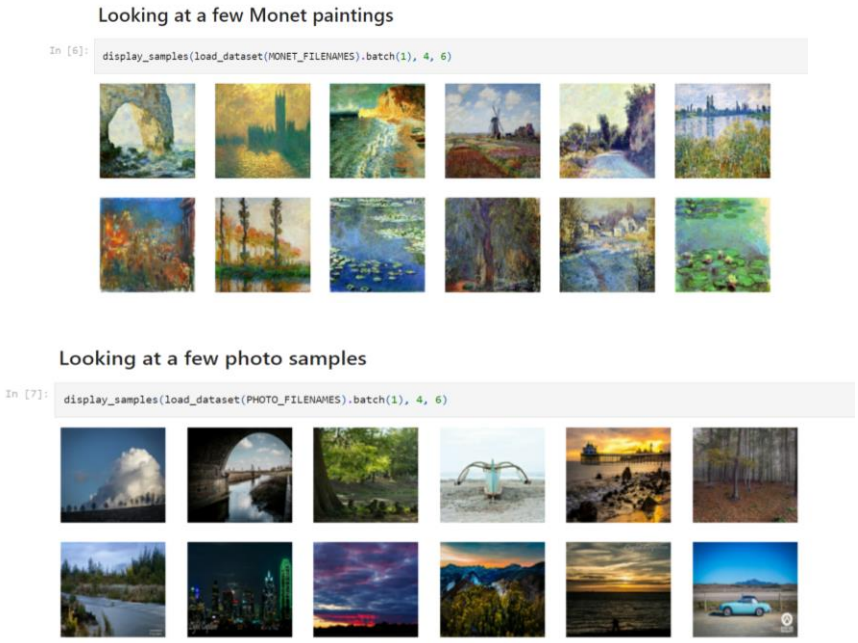
4. Identity Loss

It is desired that if the generator_monet is executed with the image of the original Monet painting, the result should be a very similar image, and also with the generator_photo and the original photo. Therefore, the loss of identity forces what the generator generates to resemble the input. It takes as inputs, 2 images, the original image and the output of running the generator with the original image.

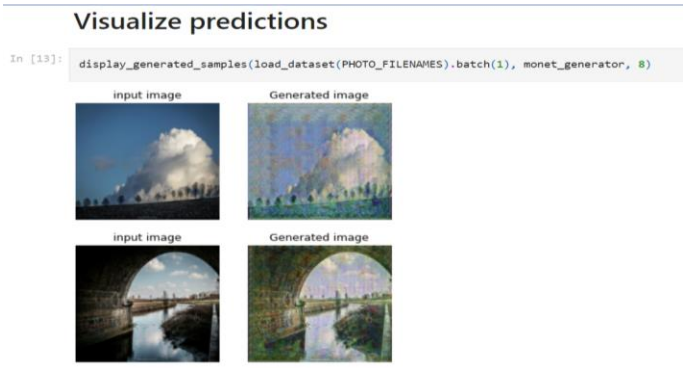
Results

Datasets

● The dataset contains four directories: monet_tfrec, photo_tfrec, monet_jpg, and photo_jpg. The monet_tfrec and monet_jpg directories contain the same painting images, and the photo_tfrec and photo_jpg directories contain the same photos. The Monet directories contain Monet paintings. We have used these images to train our model.



Sample images from the dataset.



Input images converted to Monet style images.

Evaluation Metrics

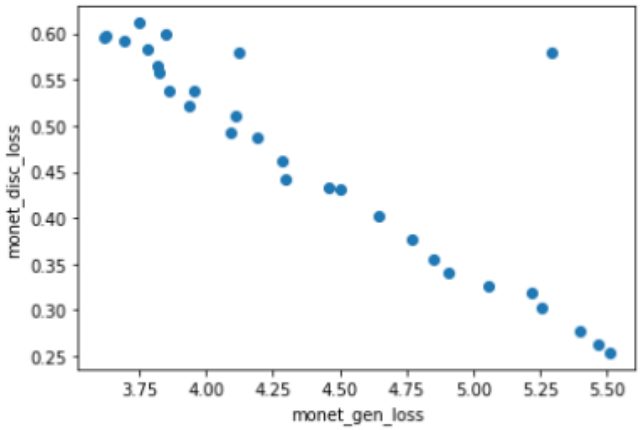
```
Epoch 1/50
300/300 - 37s - monet_disc_loss: 0.6529 - photo_gen_loss: 5.2627 - monet_gen_loss: 5.1095 - photo_disc_loss: 0.6408
Epoch 2/50
300/300 - 37s - monet_disc_loss: 0.6559 - photo_gen_loss: 3.6500 - monet_gen_loss: 3.5742 - photo_disc_loss: 0.6271
Epoch 3/50
300/300 - 37s - monet_disc_loss: 0.6506 - photo_gen_loss: 3.7302 - monet_gen_loss: 3.6005 - photo_disc_loss: 0.6190
Epoch 4/50
300/300 - 37s - monet_disc_loss: 0.6329 - photo_gen_loss: 3.6507 - monet_gen_loss: 3.5098 - photo_disc_loss: 0.6002
Epoch 5/50
300/300 - 37s - monet_disc_loss: 0.6227 - photo_gen_loss: 3.4638 - monet_gen_loss: 3.3906 - photo_disc_loss: 0.6145
Epoch 6/50
300/300 - 37s - monet_disc_loss: 0.6033 - photo_gen_loss: 3.3960 - monet_gen_loss: 3.3719 - photo_disc_loss: 0.6155
Epoch 7/50
300/300 - 37s - monet_disc_loss: 0.6180 - photo_gen_loss: 3.3020 - monet_gen_loss: 3.2777 - photo_disc_loss: 0.6247
Epoch 8/50
300/300 - 37s - monet_disc_loss: 0.6026 - photo_gen_loss: 3.3057 - monet_gen_loss: 3.2490 - photo_disc_loss: 0.6015
Epoch 9/50
300/300 - 37s - monet_disc_loss: 0.6109 - photo_gen_loss: 3.2598 - monet_gen_loss: 3.2281 - photo_disc_loss: 0.6187
Epoch 10/50
300/300 - 37s - monet_disc_loss: 0.6208 - photo_gen_loss: 3.1193 - monet_gen_loss: 3.0658 - photo_disc_loss: 0.6118
Epoch 11/50
300/300 - 37s - monet_disc_loss: 0.6148 - photo_gen_loss: 3.1307 - monet_gen_loss: 3.0677 - photo_disc_loss: 0.6060
Epoch 12/50
300/300 - 38s - monet_disc_loss: 0.6147 - photo_gen_loss: 3.0903 - monet_gen_loss: 3.0537 - photo_disc_loss: 0.6163
```

Model is trained and various parameters have been evaluated.

- Discriminator Loss {0: fake, 1: real} (The discriminator loss outputs the average of the real and generated loss)
- Generator Loss
- Cycle Consistency Loss (measures if original photo and the twice transformed photo to be similar to one another)
- Identity Loss (compares the image with its generator (i.e. photo with photo generator))

Cross Validation

Since CycleGANs are unsupervised, we are using the generator loss for photos and Monets as well as the discriminator loss for photos and monet for cross validation. The epoch value for the model is set to 50 and each epoch generates monet_gen_loss, photo_gen_loss, monet_disc_loss, photo_dis_loss. These metrics determine how the model should learn in order to generate better Monet images. As the model keeps learning the generator loss as well as the discriminator loss decreases. Below is a visualization:



Conclusion

GANs are exciting and a rapidly changing field delivering promise on generative models using deep learning methods.

Overall, the results produced by CycleGAN are very good, especially the image quality approaches that of paired image-to-image translation on many tasks. This is impressive, because paired translation tasks are a form of fully supervised learning, and this is not.

Future Work

1. Enhancing the model by integrating Reinforcement Learning along with the use of Generative Adversarial Networks (GANs) and modifying the Q-learning function to generate a more efficient model.
2. Using Autoencoders and Variational Autoencoders instead of GANs to convert images to Monet style.
3. Incorporating different styles like Van Gogh along with Monet style.
4. Along with the above suggestions, we would like to try and build a User Interface for our project where the user will be able to upload any photo and its Monet style would be generated and vice versa.

References

https://www.youtube.com/playlist?list=PLhhyoLH6IjfwIp8bZnzX8QR30TRcHO8Va

https://www.youtube.com/watch?v=_9g5WU3Yhls

https://www.tensorflow.org/tutorials/generative/cyclegan

https://www.techtarget.com/searchenterpriseai/definition/generative-adversarial-network-GAN

https://hardikbansal.github.io/CycleGANBlog/

Image-to-Image Translation using CycleGAN Model | by Rahil Vijay | Towards Data Science