CS230

# Paint like Vincent Van Gogh: Artistic Style Generator Using CycleGAN and VGG19

**Junkun Pan**
Department of Chemistry

Stanford University
jpan48@stanford.edu

**Chih-Hsuan Kao**
Institute for Computational & Mathematical Engineering
Stanford University
chkao831@stanford.edu

**Xiaoye Yuan**
Institute for Computational & Mathematical Engineering
Stanford University
chamus@stanford.edu

## Abstract

People have always been dreaming about being great artists and producing famous artworks like Claude Monet, Pablo Picasso, Vincent Van Gogh etc. Nowadays, fortunately, those art styles can be creatively generated by utilizing multiple computer vision techniques, especially by Generative Adversarial Network (GAN). In this project, we will mainly focus on generating landscape images following Vincent Van Gogh's painting style.

## 1 Introduction

In generating artificial output images with certain artistic style, we explored two different artistic style generating algorithms: CycleGAN and Neural Style Transfer by VGG19. CycleGAN [11] is capable of finding the relationship amongst given unpaired image datasets, further facilitating unpaired image-to-image translation. With its transitivity within the network, we mapped from dataset Vincent Van Gogh to dataset real photo in the respect of artistic style and content. VGG19 [9] is a convolutional neural network that is 19 layers deep. It is widely applied in general use of image classification tasks, as well as the facial recognition tasks in terms of transfer learning. Given any content input image, our first algorithm outputs an image with a general Vincent-style filter, while the second one needs another style input image in specific to generate the corresponding output. Eventually, our goal is to investigate both methods in terms of the model architecture, the quality of the generated artworks and the efficiency of the whole generating process to figure out which method can better help us become a "modern Vincent Van Gogh".

## 2 Related Work

**The Convolutional Neural Network (CNN)** [4] first made the concept of image style transfer possible. The author was able to successfully transfer the style between images, but the synthesized images are limited by resolution and suffer from low-level noise.

**The Generative Adversarial Network (GAN)** [10] further proved improvement with its generative methods. Since its development, the resolution and quality of the output images have seen rapid enhancement, becoming standard practices for style transfer applications nowadays.

**Pix2Pix** [8] is a Generative Adversarial Network designed for general purpose image-to-image translation. The pix2pix model is a type of conditional GAN where the generation of the output image is conditional on input image. The discriminator is provided both with a source image and the

CS230: Deep Learning, Fall 2020, Stanford University, CA.

target image and must determine whether the target is a plausible transformation of the source image. This serves as the basis of our CycleGAN architecture.

# 3 Dataset and Preprocessing

We collected Vincent Van Gogh's 877 paintings from Kaggle's *Best Artworks of All Time* [1] (a collection of 8446 artworks of 50 influential artists). For the content images, we collected 7038 real pictures from Kaggle's *I'm Something of a Painter Myself* [7] (a dataset for Monet art style generator). We have showcased both datasets respectively in Figure 7 and Figure 8 at the Appendix section.

For data preprocessing, we have checked the quality for both datasets, and there's no duplicated images or broken ones, which is good. For both models, to resolve the problem of having varied resolution across images, we converted our input images into size of 256 x 256 pixels, in JPEG format. In addition, our fast neural style transfer model is built on VGG19, which is pre-trained on ImageNet.

# 4 Methods

## 4.1 CycleGAN - Pytorch Lightning

### 4.1.1 Algorithms & Architectures

In our first task, we took advantage of an existing Kaggle notebook, which implements CycleGAN using Pytorch Lightning [2]. Our generator model is built on the typical encoder-decoder structure, which is conceptually represented in Figure 1.

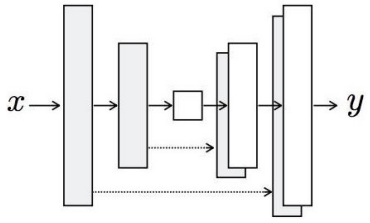


Figure 1: Encoder-decoder Structure

On the one hand, our CycleGAN generator takes in a series of images, downsamples and encodes each input image down to a bottleneck layer, skips connections (represented as dotted arrows in Figure 3) within the downsampling layers after the bottleneck layer, and then generates an output image. On the other hand, our CycleGAN discriminator is a Convolutional Neural Network that performs image classification. It takes in an image as input and predicts the likelihood of whether that image is a real or fake image.

CycleGAN distinguishes itself from typical generative networks particularly in its composition, with 2 generators and 2 discriminators, instead of 1 for each. During the training phase, one generator gets an additional feedback from the other generator. Such a feedback ensures that an image generated by a generator is cycle consistent, meaning that applying consecutively both generators on an image should yield a similar image. Then, similar to general GAN's, the discriminators are there to check if images computed by corresponding generators seem real or fake. Through this process, generators can become better with the feedback of their respective discriminators.

### 4.1.2 Training

We used learning rate of 0.0002, mini-batch size of 30, and the training ran for more than 200 epochs. The generator is optimized using L1 loss to measure the mean absolute error between input components and output components. The resulting Generator Loss is then a weighted loss of the validation loss, reconstruction loss and identity loss. Such values capture how well our generator creates stylized images. For our discriminators, we utilized MSE Loss to measure the mean squared error (squared L2 Norm) between the input components and target. We calculated our discriminator

loss by weighting the validation loss and the generator loss. This further captures how well our discriminators identify the true and generated images.

### 4.2 Fast Style Transfer with VGG19 - TensorFlow

#### 4.2.1 Algorithms & Architectures

In our second model, we stylized our content images using the concept of Neural Style Transfer. Our work is primarily adopted from a very recent Kaggle notebook [5]. By building a feedforward network, which is a residual autoencoder network that takes content image as input and spits out stylized image, we apply artistic style to content images using pre-defined loss function from [4]. For the encoder part, input images are passed in, and it propogates to the decoder part of same size as input and further predict the generated image. Then, in training time, the generated image is passed to the loss model, VGG19. The general network structure is conceptually visualized by Figure 2 below.

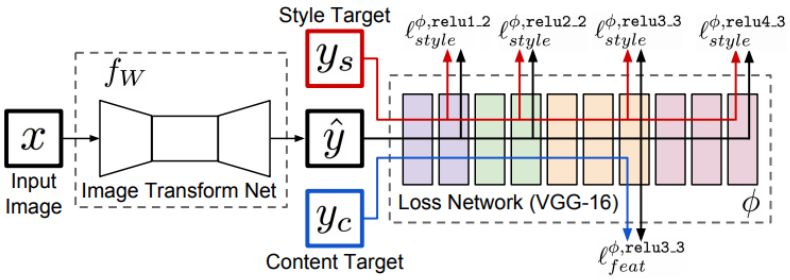


Figure 2: Fast Style Transfer Network, from [5].

#### 4.2.2 Training

Training is done by using perceptual loss defined in paper [4]. The output from decoder is passed to VGG from which we extract features and calculate style loss and content loss whose weighted sum provide perceptual loss. Throughout the training, we calculated the loss by passing style and content images both into the VGG network and using some layers to extract features. For content loss, since we'd like to learn complex features which preserves some content of image, higher layers is used. For style loss, on the other hand, lower layers of networks are used because we'd just like to learn small features. In defining our loss functions according to our content and style image shapes, we utilized the Gram Matrix by which we could obtain the MSE loss between content and style images. Then, the weighted sum of the calculated style loss and content loss would give us the perceptual loss.

With default number of epochs of 10, content image weight of 2*1e1 and style image weight of 1e2, we iteratively captured the perceptual loss at each pre-defined checkpoint throughout each epoch.

## 5 Results and Analyses

### 5.1 CycleGAN - Pytorch Lightning

We've trained the CycleGAN architectures for more than 200 epochs, which takes about 18 hours to finish. In general, we could tell from the second row of Figure 3 that images generated by CycleGAN are good combinations of arts and reality. As training progresses, this model keeps optimizing the color and the texture of the generated image as well as learning Van Gogh's art style. The visual results from every 10 epoch together with the plot of the training loss in the left part of Figure 5 suggest that after 100 epochs, this CycleGAN architecture can help us get consistent results with refined picture texture and high-quality artistic filter. Surprisingly, the generated images still preserve a sense of realism.

One of the possible weakness of this artistic style generator may come from the distribution of the real photo. As shown in Figure 4, although we can generate realistic photos for those inputs with vibrant color and high contrast("strong"), for images with light background and soft lightning("weak"), it's more likely to output oil-painting style images, which might lack a sense of reality.

Figure 3: Originals(1st row) vs. Generated: CycleGAN(2nd row) vs. VGG19(3rd row)

Figure 4: "strong" input/generated image vs. "weak" input/generated image

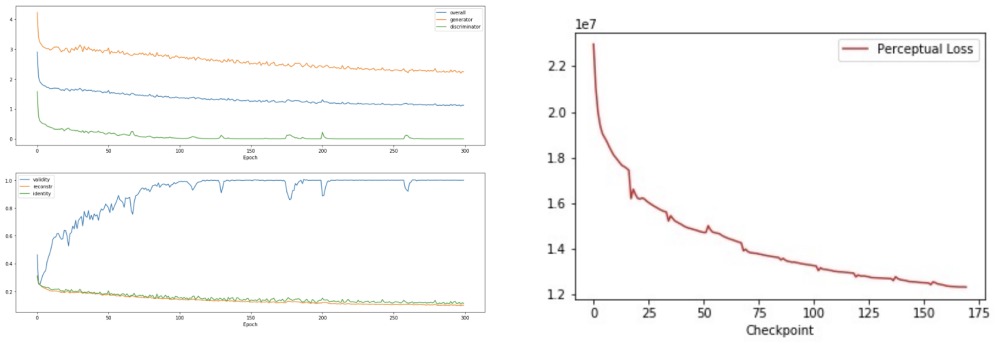


Figure 5: CycleGAN weighted loss(left) V.S. VGG19 loss(right)

### 5.2 Fast Style Transfer with VGG19 - TensorFlow

For this pretrained VGG19 model, the total time spent on training is around 6 hours, much faster compared to its CycleGAN counterpart. And ideally, we have exponentially decreasing training loss as shown in the right part of Figure 5. Since neural style transfer requires both the style images and the content images, we're targeting superimposing the art style of Van Gogh's famous painting *Starry Night* to all landscape inputs. The generated images presented in the third row of Figure 3 are the showcase results from training epoch 10 (we have 18 checkpoints setup for each epoch). We observed that they are good captures of Van Gogh's *Starry Night*, both in color distributions and textures.

Although Van Gogh's art style is easy to be recognized, the resolution and the general feelings from those generated images are much like oil-paintings rather than real photos, which is quite different from pictures generated by CycleGAN. In addition, since the art style of *Starry Night* is fairly distinctive, the individual distribution of any input image is less determinant in this case. Consequently, all generated images have consistent Van Gogh style at the expense of their realities.

## 6 Evaluation and Model Comparison

To compare and assess 2 models, we designed our evaluation metrics based on Frechet Inception Distance (FID) and qualitative human evaluation. FID is a evaluation metric that captures the similarity between generated images and real inputs, where a lower FID score corresponds to a higher similarity to the target photo [3] [6]. The formula for calculating FID score is given below:

$$d^2 = ||\mu_1 - \mu_2||^2 + Tr(C_1 + C_2 - 2\sqrt{C_1 C_2})$$

where $\mu$ refers to feature-wise mean and $C$ is the covariance matrix.

The FID values for both models are presented in Table 1.

| FID for CycleGAN | FID for VGG |
|---|---|
| 13.9289 | 203.6878 |

Table 1: The Frechet Inception Distance for last epoch

Based on the values above, the CycleGAN-generated images are much closer to the real landscape photos. In addition to the quantitative evaluation using FID, we also launched an online survey for our models. The survey asks participants to choose the real Van Gogh painting(s) from a mixture of CycleGAN-generated images, VGG19-generated images and real Van Gogh paintings. The goal of such human evaluation is to determine which model could produce the most Van Gogh-like images. It is showcased in Figure 6. We collected data from 80 Stanford students. The probability of VGG19-generated images being selected as Van Gogh's artwork is roughly 45% higher than the CycleGAN-generated ones.

## 7 Conclusion / Future Work

We implemented two artistic style generating architectures including CycleGAN and pre-trained VGG19. While the first algorithm learned to mimic the style of Van Gogh's entire artwork collections, the latter one targeted on the style of a single piece of art and performed neural style transfer. Both models can generate fantastic Vincent style images but with different focus. On the one hand, pictures generated by CycleGAN look more realistic and are regarded as real photos by most people. On the other hand, pictures generated by VGG19 look more artistic and the Vincent style can be more easily recognized correspondingly. In a word, CycleGAN and VGG19 can both serve as the better choice of artistic style generators based on various style/content requirements.

In our future work, we'd like to explore more different computer vision architectures, probably some other Generative Adversarial Network. Also, the input artworks could be extended to other famous artists like Monet, Picasso or even some oriental painters. We may also consider transfer learning techniques if the data set is relatively small. Finally, our evaluation metrics might be further improved by introducing more mathematical and statistical components.

## 8 Contributions

Carolyn Kao, Xiaoye Yuan and Junkun Pan brainstormed the ideas and proposed the architectures together. Carolyn is responsible for implementing baseline models, model training and optimization. Xiaoye Yuan is responsible for data preprocessing, implementing baseline models, and analyzing result. Junkun Pan is responsible for model training, model comparison and designing human evaluation. Finally, we completed the report together.

## References


[1] *Best Artworks of All Time*. https://www.kaggle.com/ikarus777/best-artworks-of-all-time. Accessed: 2020-09-29.

[2] *CycleGAN - Pytorch Lightning*. https://www.kaggle.com/bootiu/cyclegan-pytorch-lightning?select=pytorch_lightning-0.7.6-py3-none-any.whl.

[3] *FID score for PyTorch*. https://github.com/mseitzer/pytorch-fid. Accessed: 2020-11-10.

[4] Leon Gatys, Alexander S. Ecker, and Matthias Bethge. *A Neural Algorithm of Artistic Style*. 2015. arXiv: 1508.06576 [cs.CV].

[5] *Generate Art using Fast Style Transfer in a second*. https://www.kaggle.com/tarunbisht11/generate-art-using-fast-style-transfer-in-a-second.

[6] Martin Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. 2018. arXiv: 1706.08500v6 [cs.LG].

[7] *I'm Something of a Painter Myself*. https://www.kaggle.com/c/gan-getting-started/data. Accessed: 2020-10-30.

[8] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV].

[9] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[10] Yaxing Wang et al. *Transferring GANs: Generating Images from Limited Data*. Vol. 6. ECCV, 2018, pp. 220–236. arXiv: 1805.01677 [cs.CV].

[11] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2020. arXiv: 1703.10593 [cs.CV].

# 9 Appendix

## 9.1 Qualitative Survey Snapshot

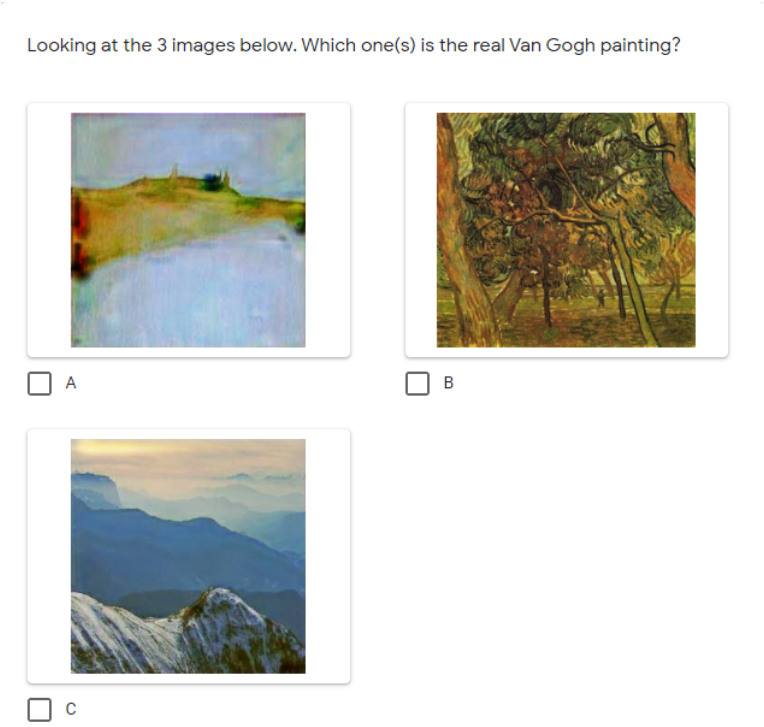


Figure 6: Google Survey we created at `https://forms.gle/uwfqEkFbu12HxTGo7`

### 9.2 Dataset Showcase

Figure 7: Style input [Vincent Van Gogh's Paintings]

Figure 8: Content input [Landscape Images]