Cycle GANs and Style Transfer

Samuel Park, Henry Ye, Aidan Flaherty

April 25, 2023

Table of Contents

Introduction/Background

Cycle GANs Analysis

Experimental Results

- Generative Adversarial Networks
  - Published: June 2014
  - Authors: Ian J. Goodfellow, Jean Pouget-Abadie , Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair , Aaron Courville, Yoshua Bengio
- Cycle-Consistent GANs
  - Published: August 2020
  - Authors: Jun-Yan, Zhu Taesung Park Phillip, Isola Alexei A. Efros
- Style Transfer
  - Published: September 2015
  - Authors: Leon A. Gatys, Alexander S. Ecker, Matthias Bethge

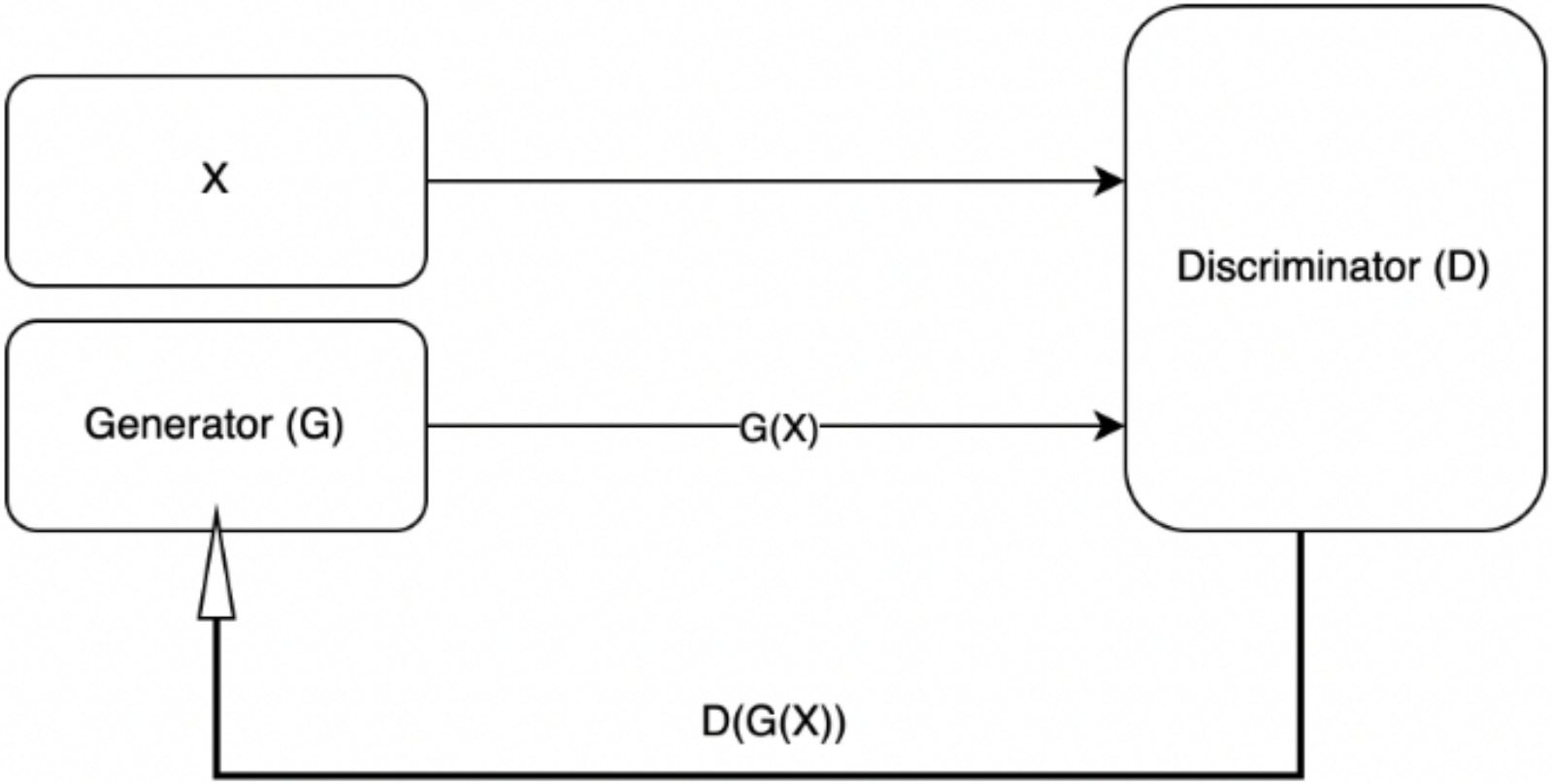
Image to Image Translation and Style Transfer

- Image-to-image translation is a computer vision task that involves mapping an input image from one domain to a corresponding output image in another domain.
- The goal is to learn a mapping between the two domains that is able to preserve important semantic and visual features of the input image while generating a corresponding output image that is consistent with the target domain.
- In the process of mapping images using neural networks, the processing of these images are separable into two different aspects: the images' content and style.
- The task of style transfer aims to change the style of an image while preserving its content. For example, transferring the style of a painting onto a photograph.

Preliminary Knowledge: Generative Adversarial Networks

- GANs (Generative Adversarial Networks) are a type of nueral network that is used to generate new data, such as images or sounds, that is similar to a given training dataset.
- Consists of two main components: a generator and a discriminator. The generator produces new data samples, while the discriminator tries to distinguish the generated and real data.
- The generator and discriminator are trained in a feedback loop: the generator tries to produce convincing samples that the discriminator cannot distinguish from the real data, while the discriminator learns to improve at distinguishing real from fake data.

Preliminary Knowledge: Generative Adversarial Networks



Limitations of Traditional Methods

- GANs can suffer from mode collapse, where the generator learns to produce a limited range of samples, resulting in limited variation in the generated data.
- GANs require large amounts of training data to generate high-quality samples, which can be a limitation in some applications where the available data is limited.

How Cycle GANs address these limitations

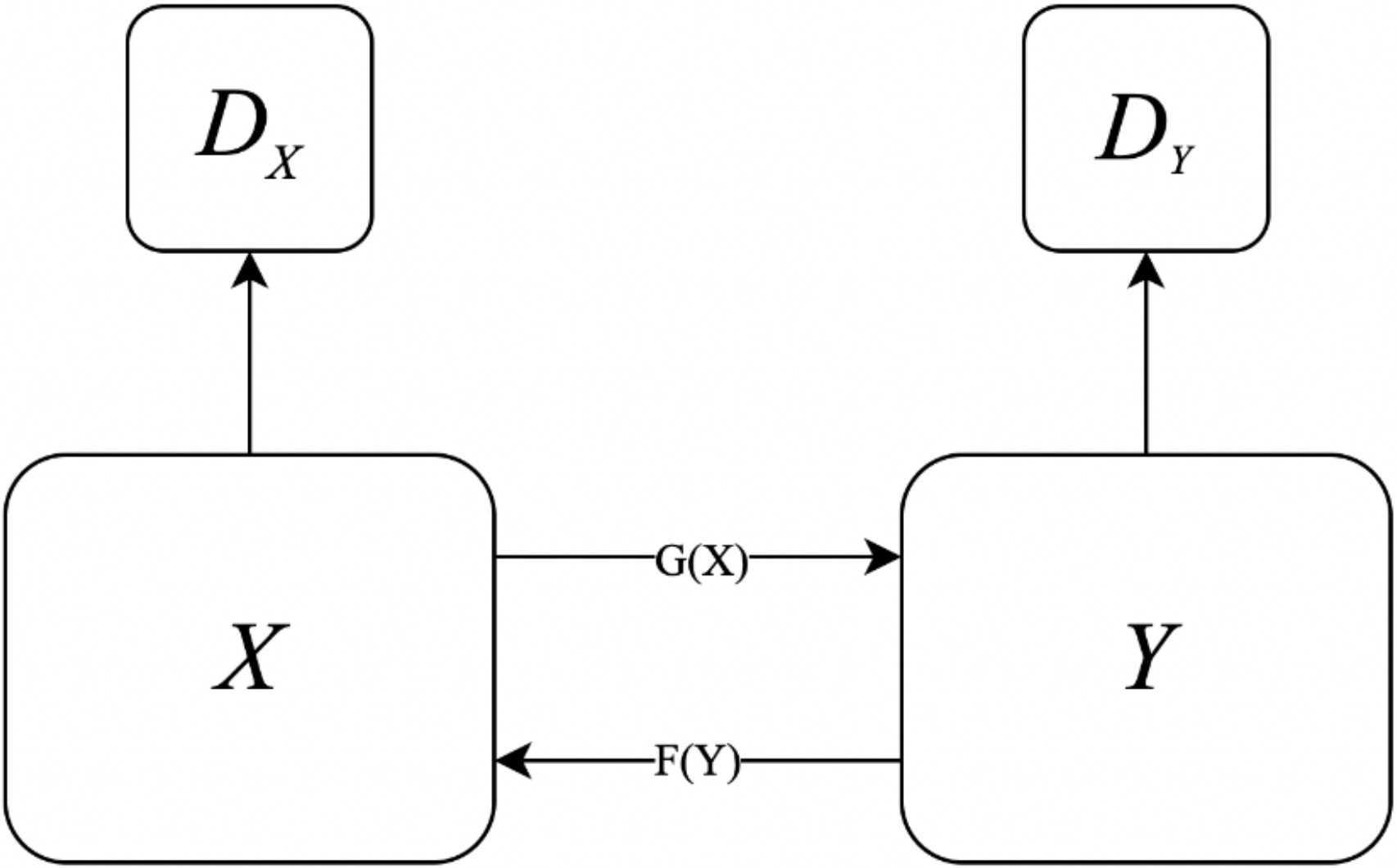
Cycle GANs can:

- learn to perform image-to-image translation without the need for paired training data, which can be difficult and expensive to obtain in some applications.
- incorporates cycle consistency loss, which ensures that the original image can be reconstructed from the translated image, leading to more visually pleasing and realistic results.

CycleGANs Architecture

- Cycle GANs consist of two generators and two discriminators (2 conventional GANs). The generators translate images from one domain to another, while the discriminators distinguish between real and translated images.
- To ensure that the generated images are consistent with the input, CycleGANs use a cycle consistency loss in addition to the adversarial loss.

CycleGANs Architecture



CycleGANs Analysis

We start by analyzing conventional GANs:

- To find the generator's distribution over data x, $p_g(x)$ we define $G(z;\theta_g)$ which represents a NN with parameters θ_g where $p_z(z)$ is the prior on input noise.
- A second NN, $D(x;\theta_d)$, which outputs a probability that the x is from the true data distribution rather than p_g.
- We then maximize D's probability of classifying the samples correctly. Additionally, training G to minimize D's probability of making a correct classification (Minimax).

Adversarial Loss Analysis

- We can then model the Loss function as the following for generator $G : X \to Y$ and its discriminator D_Y:

$$L_{adv}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))]$$

- Then modeled as a minimax game:

$$\min_G \max_{D_Y} L(G, D_Y, X, Y)$$

Loss derived from Binary Cross Entropy

- Additionally, note that the adversarial loss function can be derived from the binary cross-entropy loss function:
- Binary Cross-Entropy Loss: $L = -\sum y \log(\hat{y}) + (1-y)\log(1-\hat{y})$
- Where $y = 0$ represents when an image sample is not from the real data set, and $y = 1$ represents that the sample is from the real data set:
- When $y = 0$ and $\hat{y} = D(G(z))$:

$$L = -\log(1 - D(G(z))$$

- When $y = 1$ and $\hat{y} = D(x)$:

$$L = -\log(D(x))$$

- Adding and taking into account we are maximizing for D:

$$L = \log(D(x)) + \log(1 - D(G(z))$$

- Then by taking the expectation of the above equation:

$$E(L) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{x \sim p_g(x)} [\log(1 - D(G(z))]$$

Global Optimality of $p_g = p_{\text{data}}$

- Claim 1: when G is fixed the optimal discriminator:

$$D_G^* = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$$

- $Proof$: For any G, we want to maximize the following for D:

$$\begin{aligned} V(G, D) &= \int_x p_{\text{data}}(x) \log(D(x)) dx + \int_z p_z(z) \log(1 - D(G(z)) dz \\ &= \int_x p_{\text{data}}(x) \log(D(x)) + p_g(x) \log(1 - D(x)) dx \end{aligned}$$

Global Optimality of $p_g = p_{\text{data}}$ Cont.

- We find the maximum of $V(G, D)$ w.r.t D by solving for the value of D that sets the derivative of $V(G, D)$ to 0 w.r.t D.

$$\frac{d}{dD}V(G, D) = \int_x \frac{p_{\text{data}}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} dx = 0$$

- Setting the value inside the integral to zero gives:

$$\frac{p_{\text{data}}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} = 0$$

- Multiplying by $D(x)(1 - D(x))$ we get:

$$p_{\text{data}}(x)(1 - D(x)) - p_g(x)D(x) = 0$$

- Solving for $D(x)$, we derive the following: $D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}$

- Thus $V(G, D)$ is maximized w.r.t D when $\boxed{D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}}$ ■

Jensen-Shannon Divergence

- Jensen-Shannon Divergence, is a smoothed version of KL Divergence, defined as

$$D_{\mathsf{JS}}(P \parallel Q) = \frac{1}{2} D_{\mathsf{KL}}(P \parallel M) + \frac{1}{2} D_{\mathsf{KL}}(Q \parallel M)$$

  where $M = \frac{1}{2}(P + Q)$.

- Similar to KL Divergence, $D_{\mathsf{JS}}(P \parallel Q)$ reaches the minimum value of 0, when the two distributions of P and Q are equivalent to one another.

Global Optimality of $p_g = p_{\text{data}}$ Cont.

- **Claim 2:** The minimum value of the training criterion $C(G)$ is achieved with the value of $-\log 4$ when $p_g = p_{\text{data}}$.
- $Proof$: For any D we want to the minimize the following for generator, G:

$$
\begin{aligned}
C(G) &= \max_D V(G, D) \\
&= \mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \right] + \\
&\quad \mathbb{E}_{x \sim p_g(x)} \left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right]
\end{aligned}
$$

Global Optimality Cont.

- Assuming $p_g = p_{\text{data}}$ we will find
  $C(G) = \log(1/2) + \log(1/2) = \log(1/4) = -\log(4)$
- Then, we can add and subtract $-\log(4)$ from the original $C(G)$ to get

$$C(G) = -\log(4) + D_{\text{KL}}\left(p_{\text{data}} \parallel \frac{p_{\text{data}} + p_g}{2}\right) + D_{\text{KL}}\left(p_g \parallel \frac{p_{\text{data}} + p_g}{2}\right)$$

- Using the formula for the Jensen-Shannon Divergence, we can simplify $C(G)$. Additionally noting that JS Divergence is always non-negative, we find the minimum is achieved when $p_g = p_{\text{data}}$.

$$\min_G C(G) = 2D_{\text{JS}}(p_g \parallel p_{\text{data}}) - \log(4) = -\log(4)$$

- Thus we have shown the global optimality of $\boxed{p_g = p_{\text{data}}}$ ■

GAN Training Algorithm

Takes input of:

- The noise prior distribution: $p_z(z)$
- The true data distribution: $p_{\text{data}}(x)$
- The number of epochs E
- Batch size m
- Discriminator steps per generator step k
- Discriminator, D, parametrized by θ_d
- Generator, G, parametrized by θ_g
- Learning rate α

Produces an output of a trained GAN model.

GAN Training Algorithm Cont.

Algorithm 1 General GAN Training Procedure

for E epochs **do**

for k steps **do**

Sample minibatch of m noise samples $z^{(1)}, ..., z^{(m)}$ from noise prior $p_z(z)$.

Sample minibatch of m examples $x^{(1)}, ..., x^{(m)}$ from data generating distribution $p_{\text{data}}(x)$.

Update the discriminator by ascending its stochastic gradient with learning rate α:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right]$$

end for

Sample minibatch of m noise samples $z^{(1)}, ..., z^{(m)}$ from noise prior $p_g(z)$.

Update the generator by descending its stochastic gradient with learning rate α:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$

end for

Cycle-consistency Loss with Multiple Domains

- The key component in the implementation of CycleGANs is the use of a **Cycle Consistent Loss**.
- This loss is used as a regularization term and can be defined as the following:

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)} \|F(G(x)) - x\|_1 + \mathbb{E}_{y \sim p_{\text{data}}(y)} \|G(F(y)) - y\|_1$$

- Where $G: X \to Y$ and $F: Y \to X$.

Full Objective

- Full Objective Function:

$$L(G, F, D_X, D_Y) = L_{adv}(G, D_Y, X, Y) + L_{adv}(F, D_X, Y, X) + \lambda L_{cyc}(G, F)$$

- We thus solve the following problem:

$$G^*, F^* = \underset{G,F}{\operatorname{argmin}} \max_{D_x, D_Y} L(G, F, D_X, D_Y)$$

CycleGAN Training Algorithm

The training algorithm for the CycleGANs model takes an input of the following:

- The true data: X
- The true data: Y
- The number of epochs E
- Batch size m
- Discriminator steps per generator step k
- Discriminators, D_X and D_Y, parametrized by θ_{D_X} and θ_{D_Y}
- Generators, G and F, parametrized by θ_G and θ_F
- Learning rate α

Produces an output of a trained GAN model.

CycleGAN Training Algorithm Cont.

- Note that this Algorithm is an extension of Algorithm 1, taking into account the training of two sets of generator and discriminator pairs, instead of 1.
- Additionally, note that in practice, the training algorithm for CycleGANs uses the 2-norm instead of the log-likelihood estimate since it has shown to produce better results.
- In particular, a generator G is trained to minimize $\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[(D(G(x))-1)^2\right]$ and a discriminator D to minimize $\mathbb{E}_{y\sim p_{\text{data}}}(y)\left[(D(y)-1)^2\right]+\mathbb{E}_{x\sim p_{\text{data}}(x)}\left[D(G(x))^2\right]$

CycleGAN Training Algorithm Cont.

Algorithm 2 CycleGAN Training Procedure

for E epochs **do**
  for k steps **do**
    Sample minibatch of m samples $x^{(1)}, \ldots, x^{(m)}$ from domain X and $y^{(1)}, \ldots, y^{(m)}$ from domain Y.
    Update both discriminator D_X and D_Y by ascending their stochastic gradient with learning rate α:

$$\nabla_{\theta_{D_Y}} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(y^{(i)}\right) + \log\left(1 - D\left(G\left(x^{(i)}\right)\right)\right) \right]$$

$$\nabla_{\theta_{D_X}} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(F\left(y^{(i)}\right)\right)\right) \right]$$

  end for
  Sample minibatch of m samples $x^{(1)}, \ldots, x^{(m)}$ from domain X and $y^{(1)}, \ldots, y^{(m)}$ from domain Y.
  Update the generators G and F by descending their stochastic gradient with learning rate α:

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(x^{(i)}))) + \lambda \| F(G(x^{(i)})) - x^{(i)} \|_1$$

$$\nabla_{\theta_F} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(F(y^{(i)}))) + \lambda \| G(F(y^{(i)})) - y^{(i)} \|_1$$

end for

Experimental Design

- In order to analyze the performance of the CycleGAN model, we cannot look at loss functions as the information does not provide any insight about the quality of the model
- Necessary to intermittently check results in order to monitor improvement, stagnation, or mode collapse
- Our empirical analysis of the method surveyed a wide range of tasks not present in the paper in order to test the limitations of the model
- For the generator, we use the standard architecture of three convolutions, 9 residual blocks, two fractionally-strided convolutions with stride 12, and one convolution that maps features to RGB, all with instance normalization
- For the discriminator networks we use 70×70 PatchGANs
- Learning rate linearly decays from 0.0002 to 0

Baselines

- Our aim was to primarily test the style transfer capabilities of the network, so we selected baselines known to perform well in that area
- Our first baseline was the original neural style transfer algorithm introduced by Gatys et. al, which uses the feature space provided by the 16 convolutional and 5 pooling layers of the 19 layer pretrained VGG-Network
- Our second baseline was the real-time arbitrary style transfer algorithm introduced by Ghiasi et. al, which also uses the same VGG-Network and follows the Inception-v3 architecture closely
- Since CycleGAN is intended for arbitrary image-to-image translation, the model does not need to exceed the quality of the baselines in order to be deemed as satisfactory

Initial Failures

- The model did not successfully generate any meaningful results for some of the datasets we trained it on
- Our custom datasets horse2bird and horse2human were particularly unsuccessful, and the model did not make attempts to modify the image content even after 24 hours of training
- As such, it is assumed that CycleGAN generally performs poorly in style transfer tasks that require significant changes to the content of the image rather than just the coloration
- This is admitted by the authors of the CycleGAN paper as well, who claim that the network only minimally changed the input for a dog-to-cat dataset and works best for problems that only require textural changes

Image Colorization

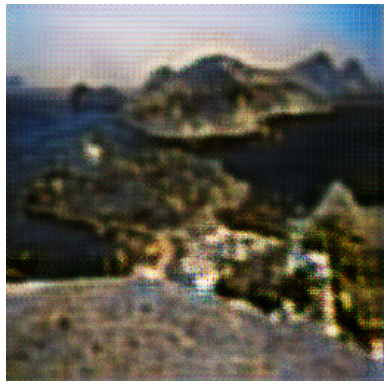
- Our first successful task was fairly simple: transforming images between colorized and grayscale versions
- Custom dataset was created with simple image modification software
- 1,000 256x256x3 images each for Class A and Class B training data, 200 256x256x3 images each for Class A and Class B test data
- CycleGAN performed surprisingly well, and even added realistic color to images where the non-grayscale version was already fairly monotone to begin with
- Image reconstruction was accurate, and even produced more reasonable colors when the original image was heavily tinted

Image Colorization Examples

Converted grayscale image

CycleGAN colorized image

Grayscale converted image



CycleGAN colorized image

Converted grayscale image

CycleGAN colorized image

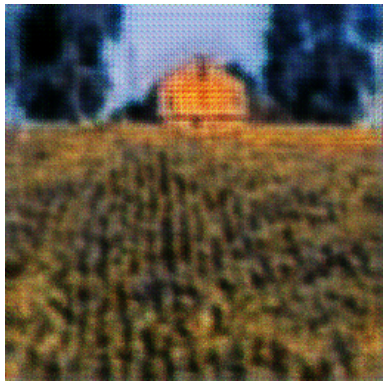
Grayscale converted image

CycleGAN colorized image

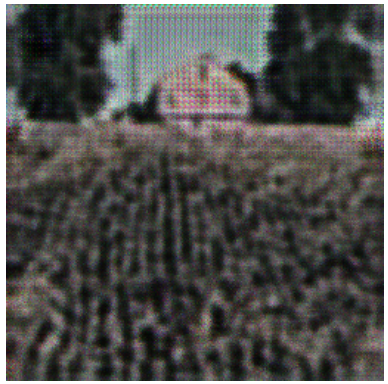
Image Colorization Examples

Original colorized image

CycleGAN grayscale



Original colorized image



CycleGAN grayscale

Original colorized image

Reconstructed image

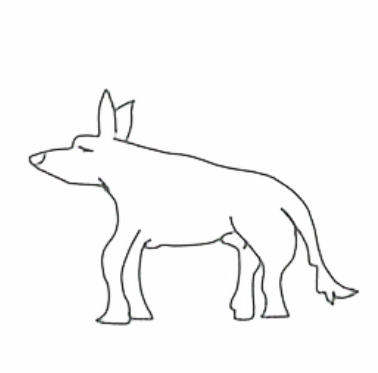
Original colorized image

Reconstructed image

Art Completion

- We aimed for a more ambitious application of CycleGANs in our next custom dataset, constructed by aggregating existing datasets
- One image space consists of poorly drawn quadrupeds, and the other consists of realistic 3D renders of horses from various angles
- 500 256x256x3 images each for Class A and Class B training data, 60 256x256x3 images each for Class A and Class B test data
- Simple task is converting a high-detail image to a drawing, more difficult task is converting a drawing to a high-quality render
- Similar performance was achieved relative to the baseline models

Art Completion Examples



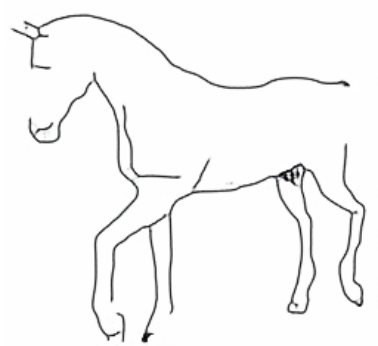
Original drawing

CycleGAN output

Original drawing

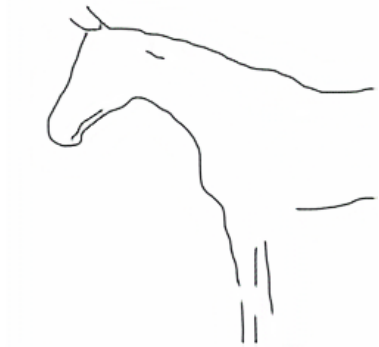
CycleGAN output

Original image



CycleGAN drawing

Original image



CycleGAN drawing

Art Completion Baseline Performance

Original content image

Style image

Stylized image

Original content image

Style image

Stylized image

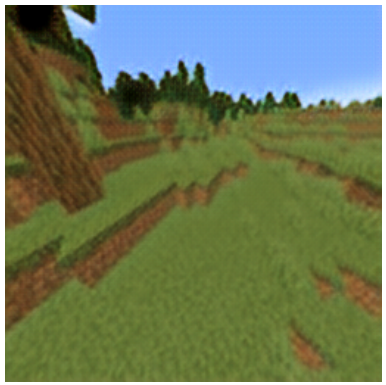
Art Completion Results

- From the observed differences between the CycleGAN model and the baselines, it would appear that CycleGAN performs approximately as well
- Both CycleGAN and out baselines do not quite accomplish photorealism in transferring from drawing to horse, and stay quite faithful to the original drawing boundary lines
- CycleGAN extends the legs of the drawings when they are unrealistically short, demonstrating that it may have greater flexibility in image modification

Voxel Rendering

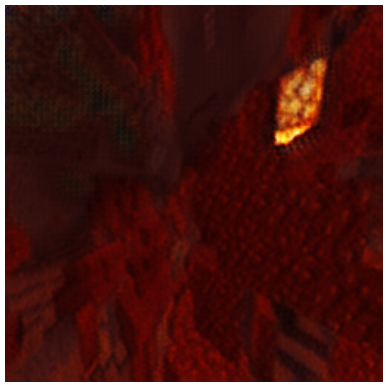
- Changing the 3D representation of an image is also a difficult style transfer task, so we constructed a custom dataset using images of scenery from the real world and the cube-based game Minecraft
- 1,000 256x256x3 images for Class A training data, 400 256x256x3 images for Class B training data, 200 256x256x3 images for Class A testing data, 30 256x256x3 images for Class B testing data
- Even though the voxel representation has lower information, transformation from real-world images into a cube-like format requires the network to understand the 3D layouts of objects
- In this task, our baselines actually performed much better than the CycleGAN model

Voxel Rendering Examples

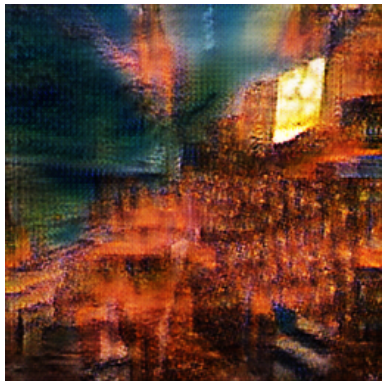


Original image

CycleGAN realism

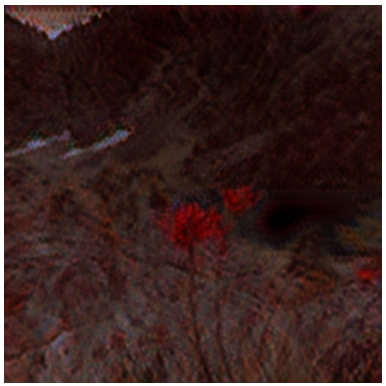


Original image



CycleGAN realism

Original image



CycleGAN stylized

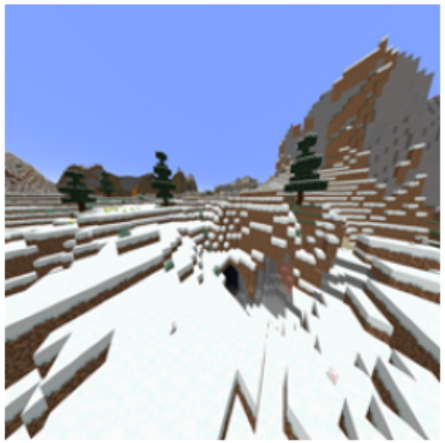
Original image

CycleGAN stylized

Voxel Rendering Baselines

Original content image

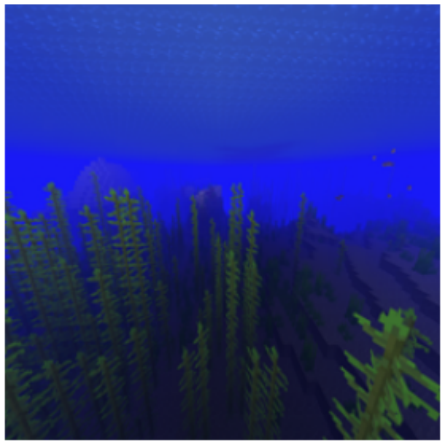
Style image



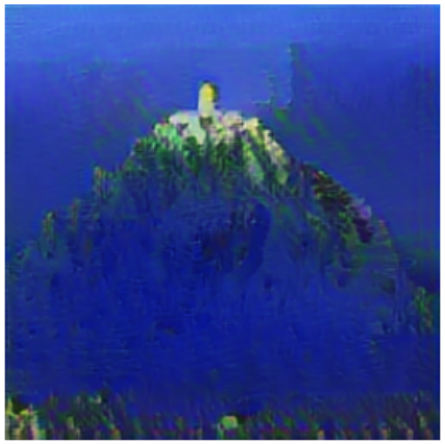
Stylized image

Original content image

Style image



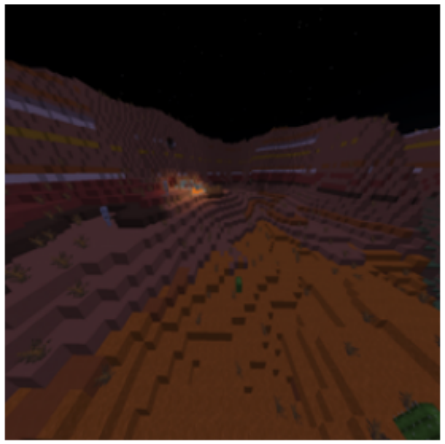
Stylized image



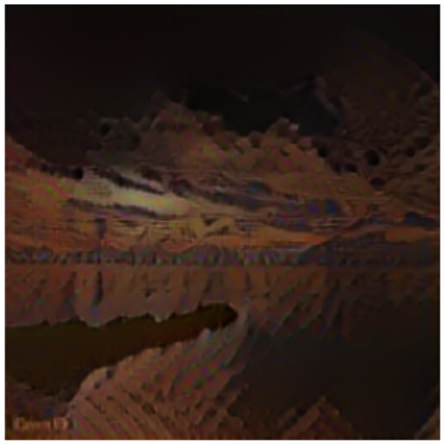
Voxel Rendering Baselines

Original content image

Style image

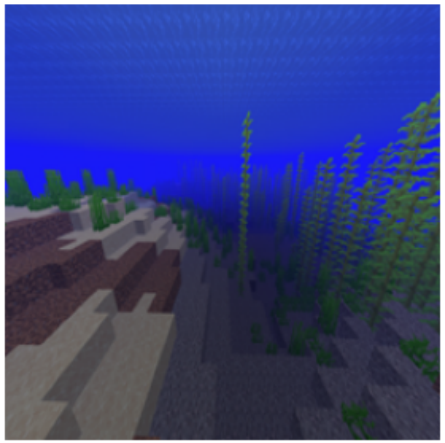


Stylized image



Original content image

Style image



Stylized image

Voxel Rendering Results

- It is clear that while the CycleGAN model is able to produce some reasonable results, it utterly fails at capturing any sort of cube-like representation
- CycleGAN is able to map voxel images to a semi-realistic environment, but the best it can do is map realistic images to the non-varied color scheme present in its training data
- The baseline models were able to preserve the original content while also creating the pixelized style from the game
- As such, it seems as though CycleGAN does not work as well in pixelization tasks

Face Completion

- Since images of people wearing masks have become easier to collect ever since the epidemic, we constructed a dataset of uncovered and covered faces
- 400 256x256x3 images each for Class A and Class B training data, 50 256x256x3 images each for Class A and Class B test data
- Our aim was to generate what the unobscured version of a face should look like only using information about their eyes, upper nose, and general facial structure
- While neither the CycleGAN model nor the baselines performed well, the CycleGAN model surprisingly outperformed the baselines

Face Completion Examples

CycleGAN faces from masks

CycleGAN masks from faces

Face Completion Baselines

Original content image

Style image

Stylized image

Original content image

Style image

Stylized image

Face Completion Baselines

Original content image

Style image

Stylized image

Original content image

Style image

Stylized image

Face Completion Results

- While the CycleGAN model was able to understand the image translation task at hand, the benchmarks completely failed at recognizing the problem
- The CycleGAN model demonstrates the core functionality of not straying too far from the original to minimize cyclic loss, meaning that it refuses to put masks on faces and lose data
- The CycleGAN model succeeded more in the mask to face image translation even though it is a more difficult problem, as it requires creating extra information rather than erasing it
- Still looks fairly creepy

Empirical Analysis Conclusion

- The CycleGAN architecture, while not completely surpassing its predecessors in the field of style transfer, is able to produce quality results in some specific applications
- CycleGAN struggles with stylistic modifications that require many changes in image content, whereas our baseline models do not seem to have much difficulty
- For best results, CycleGAN should only be employed on image transfer tasks that can be accomplished solely by changing the color of the image

Sources

- Gatys, L. A., Ecker, A. S., & Bethge, M. (2015, September 2). A neural algorithm of artistic style. arXiv.org. Retrieved April 19, 2023, from https://arxiv.org/abs/1508.06576
- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014, June 10). Generative Adversarial Networks. arXiv.org. Retrieved April 19, 2023, from https://arxiv.org/abs/1406.2661
- Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2020, August 24). Unpaired image-to-image translation using cycle-consistent adversarial networks. arXiv.org. Retrieved April 19, 2023, from https://arxiv.org/abs/1703.10593
- Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., & Shlens, J. (2017). Exploring the structure of a real-time, arbitrary neural artistic stylization network. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1705.06830

Thank You