CycleGANs Progress Report

Samuel Park, Henry Le, Aidan Flaherty

April 2023

1 Overview

We will be covering the topic of Cycle-Consistent Generative Adversarial Networks (CycleGANs). We will present the necessary background on Generative Adversrial Networks (GANs) and the mathematical backing behind the global optimal solution, algorithm, and implementation.

2 GANs and Optimality

To understand CycleGANs we must first understand the GANs model. A GAN is two multi-layer perceptrons that compete and train in a minimax adversarial zero-sum game with each other.

In this network, a discriminator learns to distinguish the difference between a sample from a model distribution (from generator) or from the data distribution. The Discriminator works to distinguish increasingly indistinguishable distributions, whereas, the generator works to produce samples indistinguishable from the data distribution.

To learn the generators distribution p_g over x we define the following:

A mapping G represents a multi-layer perception with parameters θ_g

$$G(z; \theta_g)$$

A prior on input noise

$$p_z(z)$$

A second MLP D which represents the probability that a sample comes from p_g rather than from the data distribution p_data

$$D(x; \theta_d)$$

D is trained to maximize the probability of correct classification for samples from p_g and p_data. G is simultaneously trained to maximize $log(1 - D(G(z))$. In other words:

$$min_G max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)} log D_Y(x) + \mathbb{E}_{z \sim p_z(z)} log(1 - D_Y(G(z)))$$

Claim 1: when G is fixed the optimal discriminator $D^*_G = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$

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

$$V(G,D) = \int_x p_{data}(x) log(D(x)) dx + \int_z p_z(z) log(1 - D(g(z)) dz$$

$$= \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

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

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

$$p_{data}(x)(1 - D(x)) - pg(x)D(x) = 0$$

$$p_{data}(x) - p_{data}(x)D(x) - pg(x)D(x) = 0$$

$$\boxed{D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + pg(x)}}$$

Claim 2: The minimum value of the training criterion $C(G) = max_D V(G,D)$ is achieved when $p_g = p_{data}$ with the value of $-log4$

Proof : Note that the training criterion can be represented as maximizing the log-likelihood for estimating the conditional probability that given a sample from either p_g or p_data, the discriminator classifies the sample as from p_g or p_data:

$$C(G) = max_D V(G,D)$$

$$= \mathbb{E}_{x \sim p_{data}(x)} \left[log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g(x)} \left[log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \right]$$

Additionally, note that the above notation can be derived from the binary cross-entropy loss function:

Binary Cross-Entropy Loss:

$$L = -\sum y log(\hat{y}) + (1 - y) log(1 - \hat{y})$$

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:

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

PLANNING TO GIVE GENERAL OVERVIEW OF JENSEN SHANNON DIVERGENCE Using the formula for the Jensen-Shannon Divergence, we can simplify $C(G)$:

$$min_G C(G) = min_G JSD(p_g || p_{data}) - log4$$

The JS-divergence is at its minimum value of 0 when the two specified distributions are identical. Thus we have shown the global optimality of:

$$p_g = p_{data}$$

3 Algorithm

Algorithm 1 Training Algorithm for GANs

for number of training iterations **do**
  for k steps **do**
    Sample minibatch of m noise samples $z^{(1)}, ..., z^{(m)}$ from noise prior $p_g(z)$.
    Sample minibatch of m examples $x^{(1)}, ..., x^{(m)}$ from data generating distribution $p_{data}(x)$.
    Update the discriminator by ascending its stochastic gradient:

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

  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:

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

end for

4 CycleGANs

CycleGAN is a type of Generative Adversarial Network (GAN) that is used for image-to-image translation. Unlike traditional GANs, CycleGAN does not require paired training data, which means that it can learn to translate images between two domains without the need for matching examples. This makes it particularly useful for tasks like style transfer, where the goal is to change the style of an image without altering its content.

The principal idea behind CycleGAN is to use two GANs (one for each domain) that are trained simultaneously in a cycle-consistent manner. The first GAN (G) learns to map images from domain A to domain B, while the second GAN (F) learns to map images from domain B to domain A. By training both GANs together in a cycle-consistent way, the CycleGAN can learn to translate images between the two domains even when there are no matching pairs of images in the training data.

The architecture of a CycleGAN is similar to that of a traditional GAN, but with some key differences. Instead of using a single discriminator to evaluate the realism of generated images, CycleGAN uses two discriminators (one for each domain) to provide feedback to the generators. The generators themselves are typically composed of an encoder-decoder network with skip connections to help preserve image details during translation.

Overall, CycleGAN represents a powerful approach to image-to-image translation that is particularly useful in situations where paired training data is not available. By using a cycle-consistent training procedure, CycleGAN can learn to translate images between two domains in a way that preserves important image features and details. While still a relatively new technique, CycleGAN has already been successfully applied to a wide range of image translation tasks, including style transfer, day-to-night image conversion, and more.

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

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

Where G and F are the two GANs being trained in the CycleGAN

Algorithm 2 Training Algorithm for CycleGANs

for number of training iterations **do**
  for k steps **do**
    Sample minibatch of m samples $x^{(1)}, ..., x^{(m)}$ from domain, X.
    Sample minibatch of m examples $y^{(1)}, ..., y^{(m)}$ from domain Y.
    Update both discriminator D_X and D_Y by ascending its stochastic gradient:
  end for
  Sample minibatch of m samples $x^{(1)}, ..., x^{(m)}$ from domain X.
  Update the generators G and F by descending its stochastic gradient:
end for

Note that this algorithm is an extension of algorithm 2, taking into account the training of two sets of generator and discriminator pairs, instead of 1.

Additionally, note that in training cycle GANs the use of the $l2$-norm in replacement of the log-likelihood estimate has shown to produce better convergence results.

5 Problem Set

*** Probably just going to ask to derive update step for simple GAN / CycleGAN and describe how it could be applied to particular tasks ***

Suppose we have a discriminator network consisting of five 3×3 convolutional filters followed by global average pooling (take the average pixel value of all feature maps) and an output neuron with sigmoid activation. Then, suppose we also have a generator network consisting of a single 3×3 transposed convolution with a stride $s = 2$ and padding $p = 1$.

1. We will use data $x \in \mathbb{R}^{3\times3}$, so the global average pooling in the discriminator isn't necessary and generate from noise $z \in \mathbb{R}^{2\times2}$. This gives

$$A_i(x) = \text{conv}(x, K_{D,i}, p = 0, s = 1) \qquad \text{(where } i = 1, \ldots, 5\text{)}$$
$$D(x) = \sigma(W_D A(x) + b_D)$$

   We then have the generator as

$$G(z) = \text{transposeConv}(z, K_G, p = 1, s = 2)$$

   Note that with transposed convolution, the stride essentially corresponds to dilation on the inputs to the operation.

   (a) Derive the backpropagation step for the discriminator.

   (b) Derive the backpropagation step for the generator.

2. Suppose we have another generator F which takes 3×3 images as input and produces 2×2 outputs. That is,

$$F(G(z)) = \text{conv}(G(z), K_F, p = 1, s = 1)$$

   (a) Find the subgradient of the cycle-consistency loss between $F(G(z))$ and z with respect to K_F.

   (b) Find the subgradient of the cycle-consistency loss between $F(G(z))$ and z with respect to K_G.

3. Describe how CycleGAN may be applied to a semi-supervised image segmentation task. That is, we want to perform image segmentation with very few labeled images but a lot of unlabeled images.

6 Preliminary Experimental Results

Experimental design: for generative adversarial networks, observing the loss of the networks has been demonstrated to yield very little information about the quality or convergence of the network. As such, consistently observing the generated results and comparing to earlier generated samples is necessary to determine whether the network is improving, converging, or suffering from abnormalities.

The aim of our current and future are experiments are to determine the limitations of the CycleGAN architecture. This entails both attempting to train CycleGAN networks on datasets that require small stylistic modifications to the input image and therefore are anticipated to perform well, as well as datasets that require major modifications and therefore might not result in convergence.

For baselines, we use the network introduced in the original neural style transfer paper by Gatys et. al (https://arxiv.org/abs/1508.06576), as well as the real-time arbitrary style network as introduced in the paper by Ghiasi et. al (https://arxiv.org/abs/1705.06830). As these approaches have been demonstrated to perform reasonably well on arbitrary inputs, they are appropriate for comparison with the CycleGAN results (as we are specifically testing within the domain of style transfer). They likely will perform better than CycleGAN as they are specifically designed to perform well for style transfer, while CycleGAN is multipurpose in nature where style transfer is only a subset of its full capabilities.

In order to test the limitations of the CycleGAN architecture, we have constructed custom datasets by either scraping the web or combining data from several separate pre-existing datasets. Due to memory limitations we use a batch size of 4, and the learning rate is automatically adjusted by the model.

From a network that took only several hours to train, we were able to produce reasonable results in the practical application of image colorization:

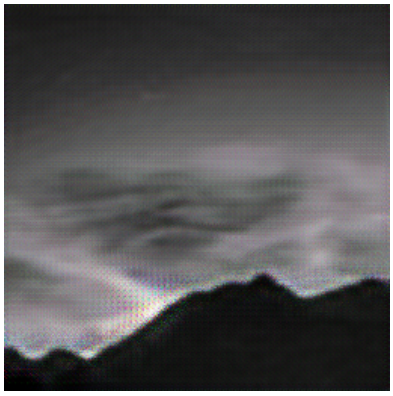


Figure 1: Original colorized image

Figure 2: Converted grayscale image

Figure 3: CycleGAN colorized image

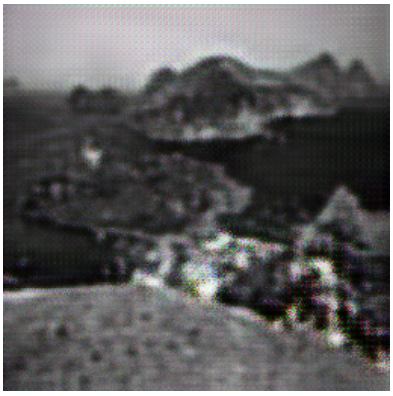


Figure 4: Converted grayscale image

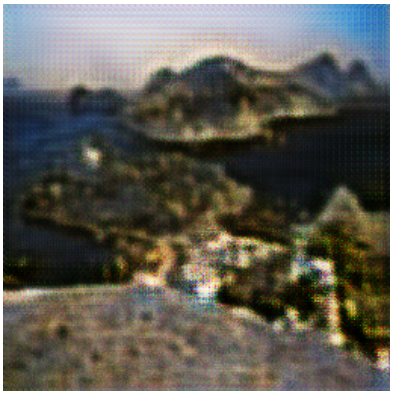


Figure 5: CycleCAN colorized image

As demonstrated by these examples, the network is able to produce a reasonable coloration of an image (even if the actual image itself was not originally colorful, as in the first case). The case from colored to grayscale is trivial so the network does well, but there are still some of the "checkerboard" artifacts that show up and add some unwanted color to the image:

Figure 6: Original colorized image

Figure 7: CycleGAN grayscale

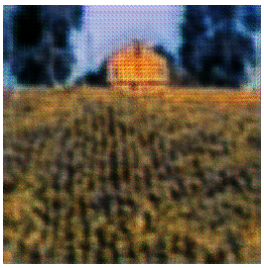


Figure 8: Original colorized image

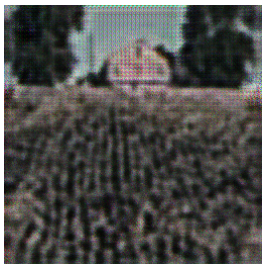


Figure 9: CycleGAN grayscale

Additionally, the network is able to produce reasonable constructions of the original images as per the cyclic architecture:

Figure 10: Original colorized image

Figure 11: Reconstructed image

Figure 12: Original colorized image

Figure 13: Reconstructed image

From figures 4 and 5, one can see that the network maps a heavily tinted photo back to a reasonable coloration based on what it observed in its training data. It's able to assign the sun a more typical sun color after correctly identifying it as such even though the original image was much more red due to the visual effect of dusk. There is some loss in image quality in reconstruction.

A more complex style transfer task is the conversion of taking drawings as inputs and returning realistic images, as the drawing-to-image conversion requires the generator to invent more nonexistent data. Our realistic inputs are all horses, and our drawings are all generally poorly drawn quadrupeds (due to data limitations). Despite spending several days training, the CycleGAN model does not quite perform as well as our benchmarks (though this may be remedied by training longer, adding more data, etc.).

The CycleGAN generator makes a fairly decent effort:

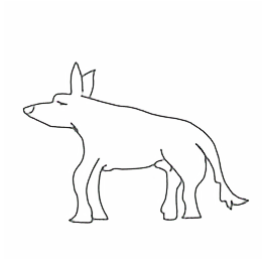


Figure 14: Original drawing

Figure 15: CycleGAN output

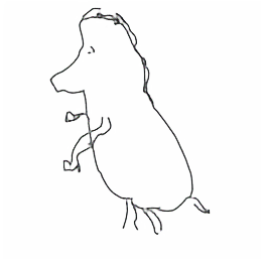


Figure 16: Original drawing

Figure 17: CycleGAN output

Figure 18: Original image

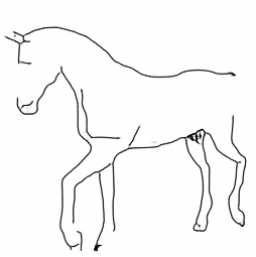


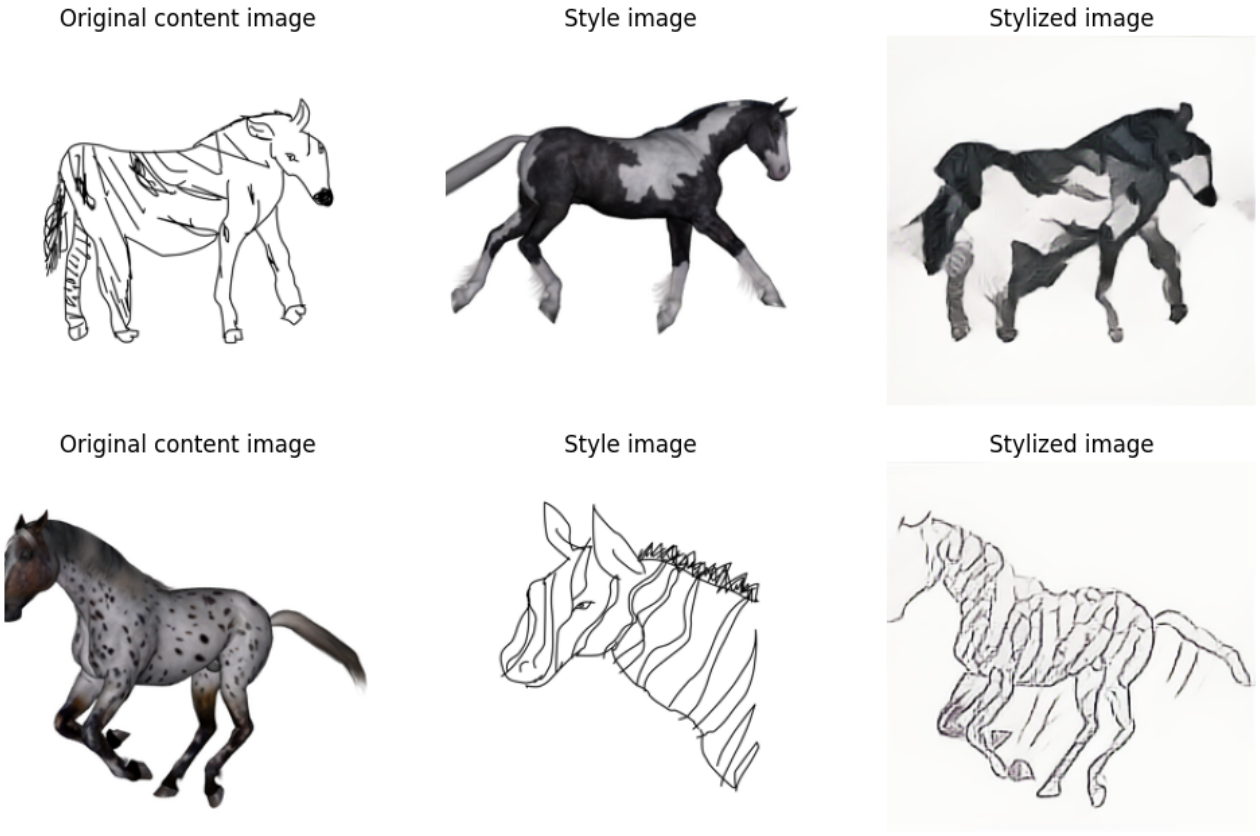
Figure 19: CycleGAN drawing

Figure 20: Original image

Figure 21: CycleGAN drawing

It's clear that the model is able to understand the general coloring of a horse, namely the coloring of the hooves and legs. Since the drawings do not have realistic proportions, it may be coloring outside of the intended region in order to make the drawing have the correct leg lengths. It is better at reducing horse images to simplistic drawings, though it doesn't really stray from the realistic image to emulate the poorly drawn versions present in the dataset. If it were true style transfer, then it would roughly approximate the horses by a poor drawing representation instead of a close to pixel perfect outline.

Our benchmarks were able to produce the following conversions sampled below, which were of slightly better quality on coloring in the drawings but still susceptible to many of the downfalls:



As is apparent from the sampled outputs, our baselines similarly fail to achieve an appropriate level of realism and instead achieve something similar to a painting (which is perfectly acceptable for their intended use case).

A third avenue of approach that we took was the transformation of real-life photos into a voxel-rendered representation (using images taken from scenery in the game Minecraft, which samples from many different real-world biomes). This is a more difficult task for the generator than it initially appears; while the render may be of lower quality than the realistic images, the 3d representation of the image is very different and depends on the viewing angle.

Our network spent several days training on our custom dataset, though it did not quite capture the desired representation:

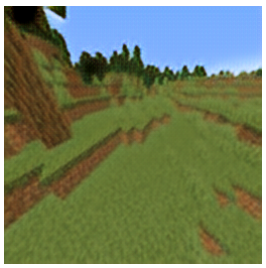


Figure 22: Original image

Figure 23: CycleGAN realism

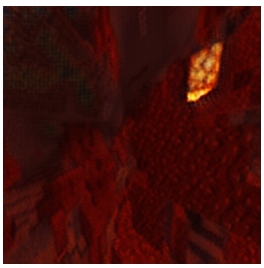


Figure 24: Original image

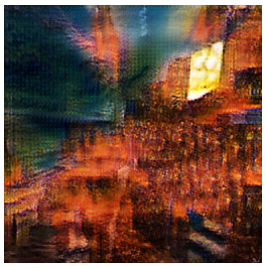


Figure 25: CycleGAN realism

Figure 26: Original image

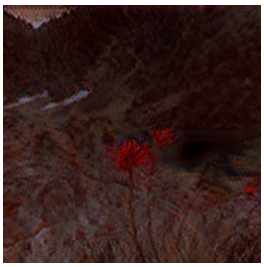
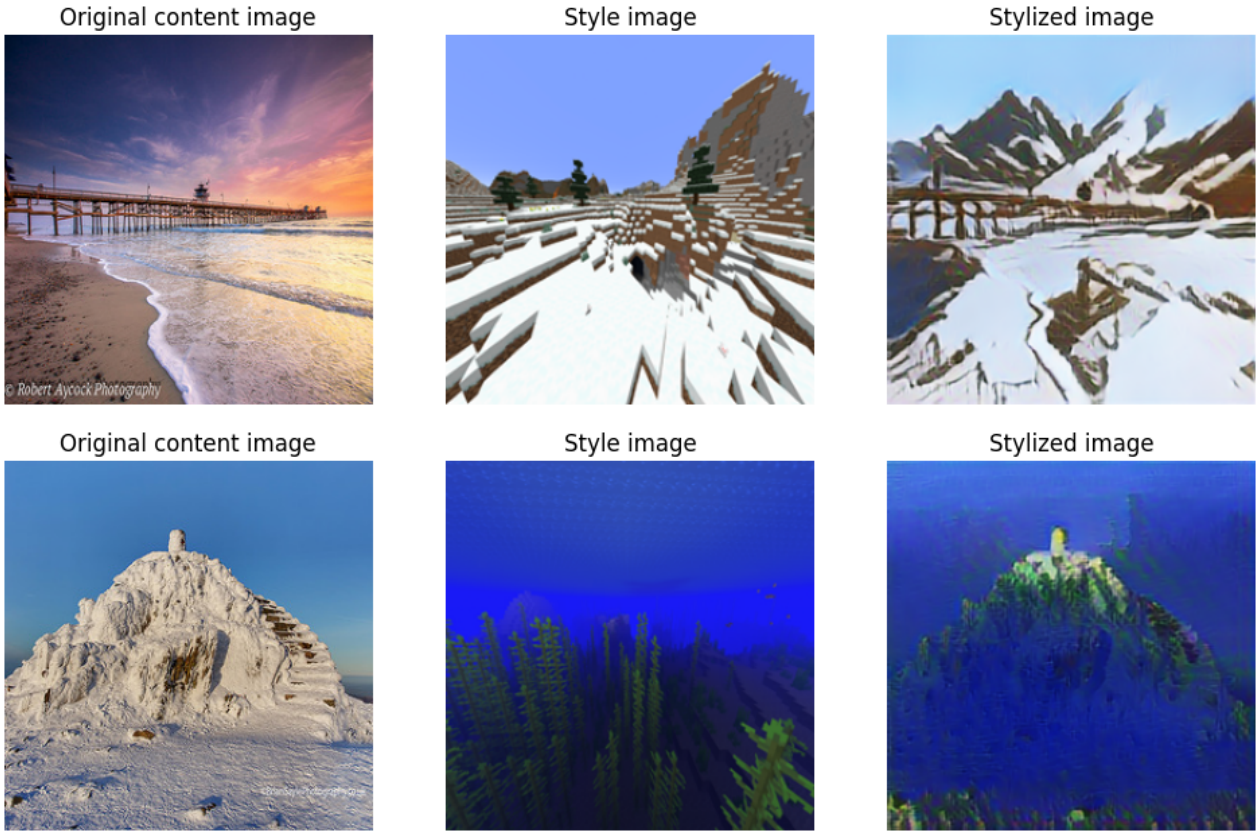


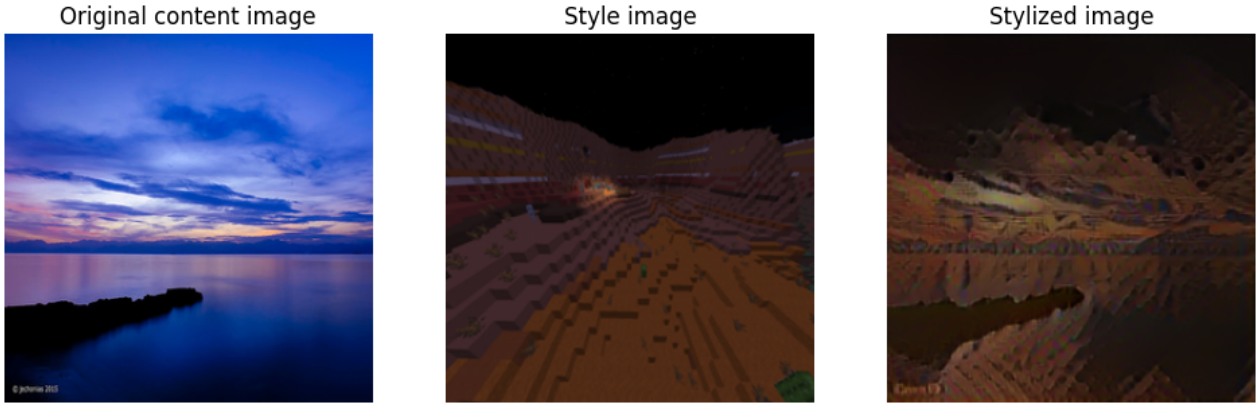
Figure 27: CycleGAN stylized

Figure 28: Original image

Figure 29: CycleGAN stylized

While the model does convert the images to the general color scheme of the game, it fails to capture the cube-like voxel rendering method. Conversely, our benchmarks were able to produce images of higher quality:





The outputs of the benchmarks are actually very impressive. The bridge to the sea was combined with the snowy mountainous landscape, but the mountains generated by the network were not directly copied from the style image. The snowy hill was transformed into a pixelated underwater mound will kelp surrounding it, which is close to the intended voxel style. The last image even took the mesa landscape from the style image and superimposed it on top of the water line, complete with rudimentary reflection.

There were also some datasets that CycleGAN failed to learn any meaningful representation for. Some earlier experiments attempted to do content transfer on a horse-to-bird dataset and a horse-to-human dataset. The network did not attempt to render one in the style of the other, as the cycle-consistency constraint ensured that the network was incapable of straying too far from the base image. It appears that CycleGANs cannot do something as drastic as rendering a horse in the style of a human, at least from over a day's worth of training time.

Future work will likely entail improving the results through increased training time (though circumventing rate limits is tricky), as well as designing new datasets to expand the scope of our tests.