Understanding CycleGAN: Converting Photos to Monet-Style Paintings Using Deep Learning

Carlos Salguero

Abstract—This paper presents a comprehensive implementation of CycleGAN (Cycle-Consistent Generative Adversarial Network) using TensorFlow, focusing on the transformation of photographs into paintings that emulate Monet's distinctive style. It demonstrates how CycleGAN's cycle consistency approach enables effective artistic style transfer while preserving original image content, even when working with unpaired datasets.

1 Introduction

The ability to transform photographs into paintings that capture the essence of Monet's artistic style represents a significant achievement in artificial intelligence. Through the application of Generative Adversarial Networks (GANs), particularly the CycleGAN architecture developed by [1]., I have implemented a system that achieves this transformation without requiring paired before-and-after images—a crucial advantage in artistic style transfer where such paired data would be impossible to obtain.

2 Competition Context

This implementation addresses the Kaggle competition "I'm Something of a Painter Myself," which challenges participants to develop AI systems capable of transforming photographs into Monet-style artwork. The competition requires generating 7,000-10,000 Monet-style images at 256×256 pixels in RGB color, saved in JPG format. Success is measured using the MiFID Score, an improved version of the FID metric that detects whether the AI is merely copying training images rather than generating novel artwork.

The competition imposes strict time constraints: 5 hours maximum for CPU and GPU implementations, and 3 hours for TPU implementations. These constraints necessitated my careful optimization of both the implementation strategy and training process.

3 Technical Framework

3.1 Scoring System

The evaluation system employs two key metrics. The base Fréchet Inception Distance (FID) [2] measures similarity between generated and real Monet paintings:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \quad (1)$$

Here, μ_r and μ_g represent the average characteristics of real and generated paintings, while Σ_r and Σ_g capture their variation patterns. The Modified FID (MiFID) extends this by incorporating creativity assessment:

$$d_{ij} = 1 - \cos(f_{gi}, f_{rj}) = 1 - \frac{f_{gi} \cdot f_{rj}}{|f_{gi}||f_{rj}|} \quad (2)$$

$$d = \frac{1}{N}\sum_i \min_j d_{ij} \quad (3)$$

$$\text{MiFID} = \text{FID} \cdot \frac{1}{d_{thr}} \quad (4)$$

4 Data

4.1 Dataset Structure

The training data consists of two unpaired sets:

- Source Domain (X): Regular photographs.
- Target Domain (Y): Monet paintings.

Each domain's dataset is:

- Formatted at 256x256 pixels in RGB color space.
- Stores in TFRecord format for efficient loading.
- Processed in batches for memory efficieny.

4.2 Sample Images

4.2.1 Monet Paintings

Fig. 1: Monet painting sample

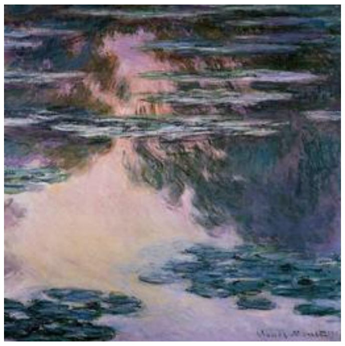


Fig. 2: Monet painting sample

4.2.2 Regular Photographs

Fig. 3: Real photograph sample

Fig. 4: Real photograph sample

5 GAN

Generative Adversarial Networks (GANs) [3] represent a revolutionary approach to generative modeling. At their core, GANs implement a competitive game between two neural networks:

- A generator network (G) that creates synthetic data.
- A discriminator network (D) that acts as a critic.

The process works like an art forger and an art critic

- The generator (forger) tries to create convincing fake samples.
- The discriminator (critic) tries to spot the fakes.
- Both networks improve through competition.

This adversarial game is expressed mathematically through a minimax objective function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{5}$$

Where:

- $p_{\text{data}}(x)$ represents the distribution of real data
- $p_z(z)$ represents random noise input to the generator
- $D(x)$ is the discriminator's estimate of the probability that x is real
- $G(z)$ is the generator's output when given noise z

During training:

1) The discriminator tries to maximize this function (make $D(x)$ close to 1 for real data and $D(G(z))$ close to 0 for fake data)
2) The generator tries to minimize this function (make $D(G(z))$ close to 1 to fool the discriminator)

6 CycleGAN

CycleGAN operates on the principle of unpaired image-to-image translation through two key innovations: cycle consistency and dual generators. The architecture consists of two generator-discriminator pairs that work in tandem: one translating from domain X to Y (photos to Monet-style paintings), and another from Y to X (Monet-style paintings back to photos).

6.1 Architecture

The CycleGAN architecture consists of two mapping functions G: X $\rightarrow$ Y and F: Y $\rightarrow$ X, where X and Y represent different image domains (e.g., photographs and Monet paintings). Each mapping has an associated discriminator (D_Y and D_X) that distinguishes between real and translated images in its respective domain.

The cycle consistency principle enforces that an image translated from one domain to another and back should closely resemble the original image. This is expressed mathematically as:

$$F(G(x)) \approx x \text{ and } G(F(y)) \approx y \tag{6}$$

where G and F are the forward and backward generators respectively.

6.2 Generator Architecture

The generator employs a U-Net-based architecture [4] that preserves spatial information through skip connections. It consist of three main components:

- An encoder path that progressively reduces spatial dimensions while increasing feature depth
- A transformer section that processes the encoded features
- A decoder path that gradually restores spatial dimensions while incorporating details from the encoder through skip connections

The skip connections are crucial as they allow the network to preserve fine details and structural information that might otherwise be lost during the encoding process.

6.3 Discriminator Architecture

The discriminator implements a PatchGAN architecture [5], which evaluates the authenticity of images at the patch level rather than globally. This design choice is based on the observation that style transfer primarily requires enforcing local consistencies.

The PatchGAN discriminator:

- Processes the image through a series of convolutional layers with increasing depth
- Outputs a matrix where each element represents the authenticity of a corresponding image patch
- Operates on overlapping patches to ensure consistency across the entire image

6.4 Downsampling Block

The downsampling block serves as the basic building unit of the encoder path and follows a consistent structure:

- A strided convolution that halves spatial dimensions while doubling feature channels
- Batch normalization to stabilize training by normalizing feature distributions
- LeakyReLU activation with a negative slope of 0.2 to allow gradient flow for negative values

This progressive reduction in spatial dimensions allows the network to capture increasingly abstract features while maintaining computational efficiency.

6.5 Upsampling Block

The upsampling block, used in the decoder path, mirrors the downsampling process with these components:

- A transposed convolution that doubles spatial dimensions while halving feature channels
- Batch normalization to maintain stable feature distributions
- ReLU activation for non-linearity
- Optional dropout layers (rate 0.5) in the early upsampling blocks to prevent overfitting
- A concatenation operation that merges features from the corresponding encoder level

The gradual upsampling process, combined with skip connections, enables the network to reconstruct fine details while maintaining the generated style characteristics.

7 Implementation

7.1 Data Pipeline

The data pipeline is structured to efficiently handle large image datasets while maintaining optimal training performance. The process involves several key stages:

7.2 Image Preprocessing

All images undergo standardization through the following steps: 1. Normalization of pixel values from [0, 255] to [-1, 1] range using the formula:

$$x_{normalized} = \frac{x}{127.5} - 1 \tag{7}$$

2. Storage optimization using TFRecord format, which provides:

- Efficient serial data storage.
- Faster data loading during training.
- Reduced memory overhead.

8 Training Algorithm

8.1 Loss Functions

The training process minimizes a composite loss function:

$$\mathcal{L}_{total} = \mathcal{L}_{GAN} + \lambda_{cycle}\mathcal{L}_{cycle} + \lambda_{identity}\mathcal{L}_{identity} \tag{8}$$

Where:

- $\mathcal{L}_{GAN}$: Adversarial loss ensuring realistic outputs.
- $\mathcal{L}_{cycle}$: Cycle consistency loss ($\lambda_{cycle} = 10.0$).
- $\mathcal{L}_{identity}$: Identity mapping loss ($\lambda_{identity} = 0.5$)

8.2 Training Process

The training algorithm follows these steps for each batch:

Algorithm 1 CycleGAN Training Process

```
1:  Input: Real images from domains X and Y
2:  Parameters: Learning rate $\alpha = 2e^{-4}$, $\beta_1 = 0.5$
3:  for each epoch do
4:      for each batch do
5:          Generate fake images: $\hat{y} = G(x)$, $\hat{x} = F(y)$
6:          Compute cycle reconstructions: $\tilde{x} = F(\hat{y})$, $\tilde{y} = G(\hat{x})$
7:          Compute identity mappings: $x_{id} = F(x)$, $y_{id} = G(y)$
8:          Update discriminators $D_X$ and $D_Y$
9:          Update generators G and F
10:         Track metrics and losses
11:     end for
12: end for
```

8.3 Optimization Strategy

The model uses the Adam optimizer with:

- Learning rate: $2e^{-4}$.
- $beta_1 = 0.5$.
- $beta_2 = 0.999$ (default).

Each network component (generators and discriminators) has its own optimizer instance to ensure stable training.

9 Results and Analysis

9.1 Image Transfer Results

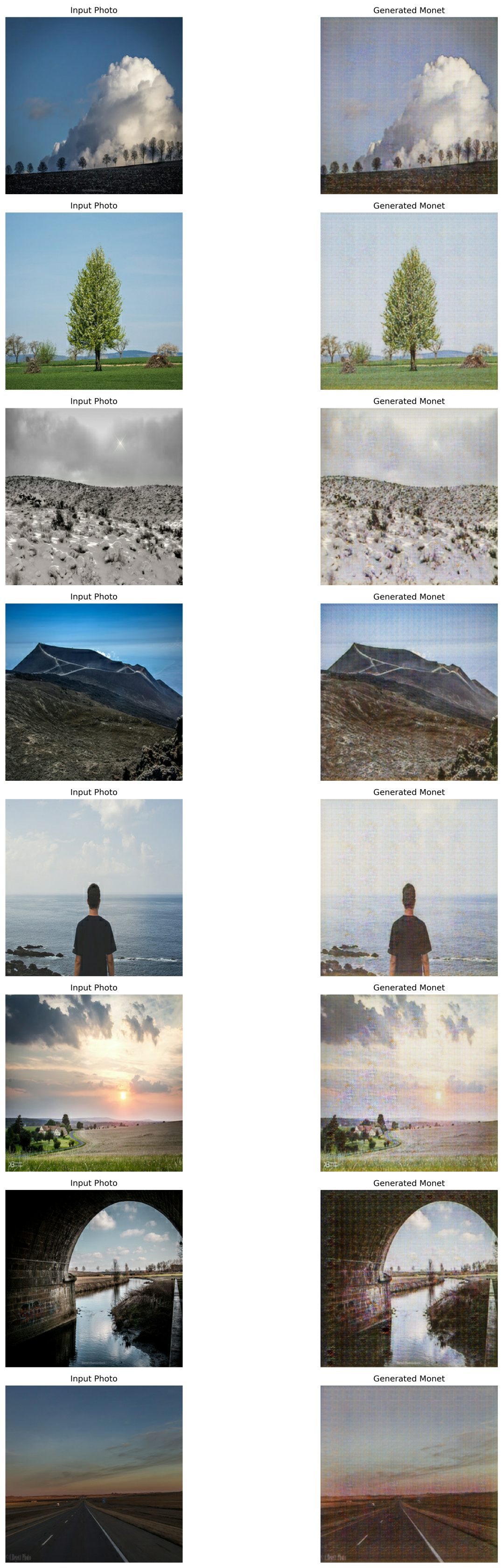


Fig. 5: Model generated images

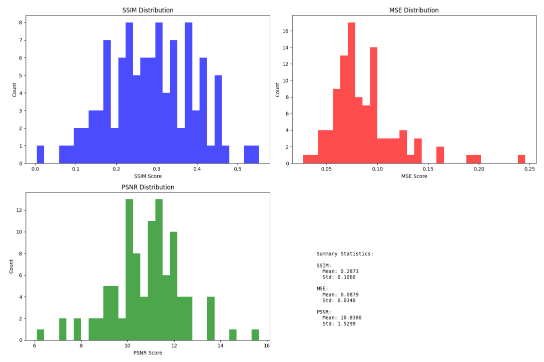
9.2 Visual Quality Assessment

The qualitative results demonstrate the model's capability to generate Monet-style paintings while preserving the structural integrity of the input photographs. Key observations from the generated samples include:

1) **Structural Preservation**: The model successfully maintains the overall composition and major elements of the landscapes, particularly evident in the mountain scenes and sunset images.
2) **Color Transformation**: The generated images show characteristic Monet-style color palettes, with a tendency toward warmer tones and subtle color variations, though maintaining the dramatic lighting effects of the original photographs.
3) **Texture Generation**: The model introduces painterly textures that simulate brushstrokes, particularly visible in the sky regions and landscape features.

9.3 Quantitative Metrics Analysis

The model's performance was evaluated using multiple metrics, showing consistent results across the test dataset:



```
Summary Statistics:

SSIM:
  Mean: 0.2873
  Std: 0.1060

MSE:
  Mean: 0.0879
  Std: 0.0340

PSNR:
  Mean: 10.8380
  Std: 1.5299
```

Fig. 6: Distribution of quality metrics across the test dataset

9.3.1 Structural Similarity (SSIM)

- Mean SSIM: 0.2873 (±0.1060)
- Distribution shows a normal curve centered around 0.3
- Indicates moderate structural preservation while allowing for stylistic changes
- The relatively low score is expected due to the intended stylistic transformation

9.3.2 Mean Squared Error (MSE)

- Mean MSE: 0.0879 (±0.0340)
- Distribution shows a right-skewed pattern
- Majority of transformations maintain pixel-wise similarity within acceptable bounds
- Higher MSE values for more dramatically transformed images

9.3.3 Peak Signal-to-Noise Ratio (PSNR)

- Mean PSNR: 10.8380 (±1.5299)
- Normal distribution centered around 10-11 dB
- Indicates consistent quality across transformations
- Lower PSNR values are expected due to the intentional style modification

9.3.4 *Fréchet Inception Distance (FID)*

- FID Score: approximately 150
- Indicates the generated images maintain reasonable similarity to real Monet paintings while avoiding direct copying.
- Suggests successful style transfer while maintaining photographic content.

9.4 Limitations and Observations

1) The model shows some limitations in:
    - Handling extreme lighting conditions.
    - Maintaining fine details in complex features.
    - Consistency of brush strokes simulation across different image regions.
2) Strengths demonstrated include:
    - Consistent color palette transformation.
    - Preservation of major compositional elements.
    - Stable performance across various landscape types.

References

[1] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.

[2] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2015, pp. 234–241.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1125–1134.