Painting with Generative Adversarial Networks: Generating Monet-Style Images Using Novel Techniques

Garrett Devereux
gdev@uw.edu

Deekshita S Doli
deekdoli@uw.edu

Begoña García Malaxechebarría
begogar9@uw.edu

Abstract

Generative Adversarial Networks (GANs) have emerged as a powerful and versatile tool for generative modeling. In recent years, they have gained significant popularity in various research domains, particularly in image generation, enabling researchers to address challenging problems by generating realistic samples from complex data distributions. In the art industry, where tasks often require significant investments of time, labor, and creativity, there is a growing need for more efficient approaches that can streamline subsequent project phases. To this end, we present MonetGAN, a pioneering framework based on the Least Squares Deep Convolutional CycleGAN, specifically tailored to generate images in the distinctive style of 19th-century renowned painter Claude Monet. After achieving non-trivial results with our baseline model, we explore the integration of advanced techniques and architectures such as ResNet Generators, a Progressive Growth Mechanism, Differential Augmentation, and Dual-Objective Discriminators. Through our research, we find that we can achieve superior results with small changes to gradually build up a successful model, rather than adding too many varying complexities all at once. These outcomes underscore the intricacies involved in the training of GANs, while also opening up promising avenues for further advancements at the intersection of art and artificial intelligence.

1. Introduction

Imagine being able to bring the artist Claude Monet back to life and request him to paint anything you desire or instantaneously reproduce the exact scene he gazed upon while creating any of his iconic masterpieces. While time travel may not be possible, we can provide a similar outcome through MonetGAN - a resurrection of Monet in the shape of a GAN, or, in other words, a Least Squares Deep Convolutional CycleGAN trained on a collection of real-world images and Monet's distinctive style paintings.

In this paper, we propose the development of MonetGAN with the aim of creating realistic and high-quality images that closely resemble Monet's paintings. The ability to generate images that capture the essence of a particular artist opens up new opportunities for the art industry, allowing for the creation of original and personalized pieces, the reproduction of lost or damaged artwork, and providing an immersive experience for museum or gallery visitors. Additionally, capturing realistic images through paintings enables the recovery of significant historical heritage, offering a clearer visualization of the past. Our goal is to create a robust and reliable generative model that leverages the latest advances in the field to achieve competitive results.

2. Related Work

The field of image style transfer has rapidly evolved in recent years, with numerous papers contributing to its development. One of the earliest works was "A Neural Algorithm of Artistic Style" by Gatys et al., which introduced a novel method for generating stylized images using neural networks [5]. However, the General Adversarial Network (GAN) model, introduced by Goodfellow et al., had a significant impact in the field and enabled the generation of high-quality images through adversarial training of generators and discriminators [6].

Moving forward, Radford et al. further advanced this work by proposing a new architecture, the Deep Convolutional GAN (DCGAN), that improved stability and convergence by replacing pooling layers with strided convolutions and using batch normalization [11]. Additionally, the introduction of Least Squares GANs (LSGANs) by Mao et al. brought about improvements in the stability and convergence of GAN training [10]. More recently, Zhu et al. developed a new framework called CycleGAN for unpaired image-to-image translation using cycle-consistent adversarial networks with ResNet generators. [15].

Significant contributions to the field also include advancements in the architecture of Generators and Discriminators. Ronneberger et al. introduced the UNet Generator, which employed a combination of down-sampling and up-sampling with long skip connections to better preserve image details [12]. Similarly, Isola et al. proposed the PatchGAN discriminator, which operates on local image patches

to provide more detailed feedback on image quality compared to traditional discriminators [7].

Additionally, several studies have explored the limitations of GANs when dealing with a limited amount of training data. To address this issue, techniques like Zhao et al.'s Differential Augmentation [13] and Nguyen et al.'s Dual-Objective Discriminators [4] were proposed. These methods aimed to mitigate the performance deterioration of GANs under such conditions.

As the field continues to expand, larger companies such as NVIDIA play a significant role in developing architectures such as StyleGAN with truncation [9], which achieves hyper-realistic results in Human eYe Perceptual Evaluation [14]. Inspired by these advancements, we implemented a Progressive Growth Mechanism on top of our baseline model, showcasing a promising starting point for new developments.

Overall, the advancements in image style transfer have been driven by a multitude of papers over the years, each making valuable contributions that we attempt to leverage in our implementation of the MonetGAN.

3. Methods

3.1. Design Principles

When first beginning our project, we outlined several core design principles to help guide our progress and ensure we had reasonable but novel deliverables. These principles included well-defined goals, re-usability, re-producibility, and a focus on the learning objectives over each stage of development. Keeping these in mind, we first decided to base our project around the Kaggle competition "I'm Something of a Painter Myself" [8]. This handles the collecting, cleaning, and labeling of images as well as the evaluation of our project while providing valuable tutorials and resources. With a very limited project timeline and little previous experience in the space, this was a key decision that contributed to the velocity of our baseline understanding and implementation.

That being said, we spent the first stage of the project implementing our own baseline notebook with the purpose of becoming comfortable coding, training, and tuning models. This would also provide initial results we could work to improve in the coming weeks. This notebook is thoroughly documented and diagrammed, giving us a reusable pipeline for future models while outlining clear learning objectives and implementation details that allow reproducible results. The descriptions in this paper are based upon this documentation, and it is highly recommended to check out the actual notebook and Kaggle submission for more thorough details and visual results [3].

Additionally, we worked to consistently meet as a team to communicate our progress and ensure we stay aligned with these goals. Entering the second stage of the project, we each branched off to work on a unique model. These meetings severed as a way to share our results with each other throughout our parallel development and plan out ways to connect everything together into a poster and final report.

3.2. Dataset

The Kaggle Provided dataset consists of two sets of images: 300 Monet Paintings, and 7028 Photos, all sized 256x256 in JPEG format [8]. The objective of the competition is to create a model that learns the style of the paintings in the dataset and then is able to turn the 7028 Photos into Monet Paintings for final evaluation. Several examples of input Photos and Paintings can be seen in the documented notebooks [3] [1] [2] as well as figures throughout the paper.

3.3. MonetGANv1 - LS DC CycleGAN

The General Adversarial Network (GAN) architecture is an approach to training a model for image generation that is comprised of two models: a generator and a discriminator [6]. The generator takes in input from one domain, such as Photos, and outputs an image of the other domain, in our case, Monet Paintings. In contrast, the discriminator takes an image as input and predicts whether it is from the dataset or generated. Both models are trained against each other in a game, as the generator is updated to better fool the discriminator and the discriminator is updated to better detect generated images.

The CycleGAN is simply an extension of the GAN architecture that involves the simultaneous training of two generators and two discriminators [15]. Essentially, we have one generator that maps Photos to Monet Paintings, another that maps Monet Paintings to Photos, and a discriminator for each domain. In the end, we will remove the desired generator from the architecture and can generate Monet Paintings from input Photos. MonetGANv1 utilizes this architecture, which is outlined in Figure 1.

3.3.1 Generators

MonetGANv1 has two generators: one that takes in a Photo and outputs a Monet Painting and another that takes in a Monet Painting and outputs a Photo. For these generators, we take advantage of the UNet architecture introduced by Ronneberger et al. [12]. This works by progressively downsampling the input 8 times through a series of convolutions, going from an initial size of 3x256x256 to an intermediate size of 512x1x1. The generator then upsamples the image back up to 3x256x256 through transposed convolutions while establishing long skip connections, giving it the U-shaped architecture. These connections store the intermediate results after each downsample, and concatenate it with

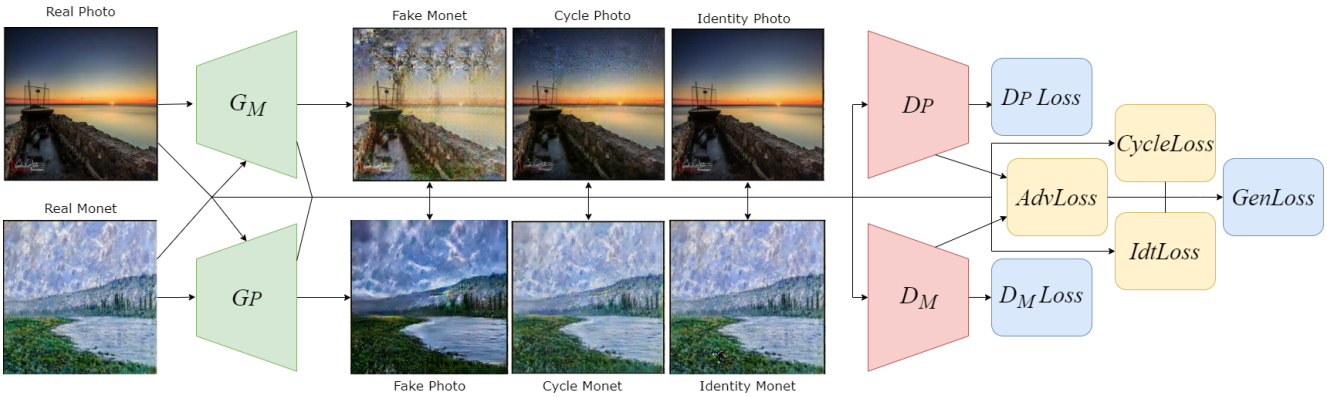


Figure 1. MonetGANv1 Architecture

the corresponding upsampling result. By adding skip connections, we are adding a highway for gradient flow, helping bypass the vanishing gradient problem during training. This process of upsampling and downsampling through convolutions makes MonetGANv1 a Deep Convolutional CycleGAN.

3.3.2 Discriminators

Similarly, MonetGANv1 has two discriminators: one for identifying images from the Photos dataset, and another for identifying images from the Monet Paintings dataset. Rather than simply outputting a single prediction score of how likely an input is real or generated, our baseline models use the PatchGAN architecture to instead predict if certain patches of the image are real or fake [7]. Specifically, our discriminators take in an image of size 3x256x256 and then run convolutionally across the image with a series of down-samples to produce a 1x30x30 matrix of scores. This provides a more fine-grained prediction that will allow us to define more specific loss functions and improve learning.

3.3.3 Loss Functions

During the training process, we employ multiple loss functions to assess the performance of the generators and discriminators. The Generator Loss is comprised of three components: Adversarial Loss, Cycle Consistency Loss, and Identity Loss. Adversarial Loss quantifies how effectively the generator fools the discriminator by computing the mean squared error between the fake image's discriminator score and an ideal tensor of all 1s. Through backpropagation, we optimize the generator by minimizing this squared error, leading to improved fake image generation. Moreover, the other two components of the Generator Loss measure properties we want the generators to retain and ensure that they continue to produce reasonable results while preventing issues such as mode collapse. Cycle Consistency Loss calculates the L1 difference between the original image and the reconstructed image (obtained by passing the original image through both generators). This loss component considers both generators and facilitates their collaboration in the generation process. Identity Loss verifies that the input image remains unchanged by passing it through its corresponding generator. The total Generator Loss is the sum of these three losses, and a detailed computation graph of the process can be found in the baseline notebook's loss section [3].

In contrast, the Discriminator Loss evaluates the loss for a single discriminator at a time. It assesses the discriminator's performance on real and generated images by minimizing the mean squared error between the real and fake discriminator scores and tensors of 1s and 0s respectively, computing its ability to discern images. It is worth noting that the choice of mean squared error is based on the LSGAN architecture [10], categorizing MonetGAN as a Least Squares DC CycleGAN. Finally, the Discriminator Loss is computed by averaging these scores. Through backpropagation, we iteratively minimize the mean squared loss of the discriminators, enhancing their image recognition capabilities.

3.3.4 Training

With our defined loss functions, we can proceed to the training phase. To initialize the CycleGAN module, we start by setting up Monet and Photo generators and discriminators. During each training step, we utilize the generators and discriminators to generate various inputs required for the loss functions, such as generated images, identity images, reconstructed images, and relevant discriminator scores. Subsequently, we compute the total Generator Loss and update both generators through the backward and step functions. Similarly, we calculate both Discriminator Losses and perform backward and step operations for each, concluding a

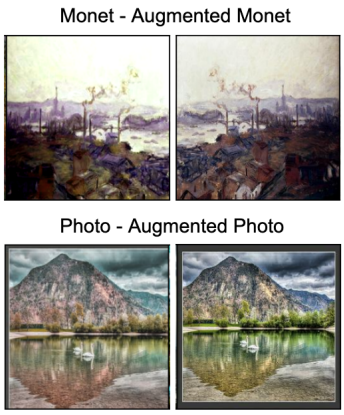


Figure 2. Data Augmentation

single training step. This training process forms the foundation for our future models, which will follow a similar approach with the inclusion of new techniques.

3.4. MonetGANv2: Introducing Complexities

After successfully completing our baseline model, we began to add complexities aiming to improve performance. To begin the process, we explored the techniques of adding data preprocessing and learning rate schedulers as well as looking at new generator architectures.

3.4.1 Data Preprocessing

To improve the performance and generalization of our model, we implemented data augmentation. By applying various transformations to the training images, we can effectively increase the size and diversity of the dataset, providing the models with more varied examples to learn from. Examples of our augmentations can be seen in Figure 2.

The custom data augmentation pipeline we implemented serves multiple purposes. First, the resizing of images ensures that any input image can flow into the network, allowing the generator to downsample and upsample without worrying about the size. Second, random cropping introduces spatial variability, forcing the models to learn invariant features that are not tied to specific image regions. Third, the random horizontal flip adds robustness to the models by training them to handle images with different orientations. Finally, the color jittering enhances the color distribution in the dataset, making the models more adaptable to variations in lighting conditions and color tones.

Overall, the augmented dataset exposes the models to a broader range of image variations, enabling them to learn more robust representations and perform better on unseen data.

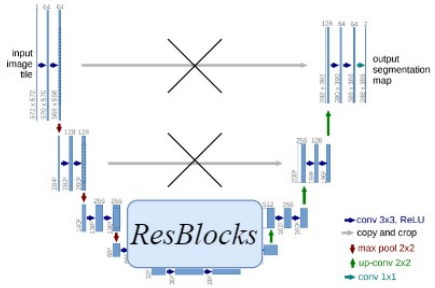


Figure 3. ResNet Generator

3.4.2 ResNet Generators

The choice of generator architecture has a significant impact on the performance and quality of the generated images. While our baseline implementation relied on UNet generators, we worked to implement ResNet generators, the generator in the initial CycleGAN paper, for our MonetGANv2 [15].

The ResNet generator is almost identical to the UNet generator consisting of the same downsampling path and upsampling path. The difference is that the ResNet generator does not have long skip connections from the concatenations of outputs. Instead, it uses residual blocks with short skip connections where the original input is added to the output. This can be seen in Figure 3.

3.5. Industry Architecture

Due to our interest in improving the evaluation of image quality produced by our model, we began exploring the concept of Human eYe Perceptual Evaluation [14] after it was mentioned in lecture. This led us to delve deeper into NVIDIA's StyleGAN with truncation architecture, known for its ability to achieve hyper-realistic results in HYPE. However, integrating the StyleGAN architecture into our already complex model proved to be a more challenging task than anticipated. Considering our limited time constraints, we made the decision to initially focus on implementing its Progressive Growth Mechanism. This approach allowed us to gradually tackle the intricacies of the StyleGAN integration while making progress within our given timeframe.

3.5.1 Progressive Growth Mechanism

The Progressive Growth Mechanism, a key feature of NVIDIA's StyleGAN with truncation architecture, addresses the challenge of training large-scale generative models by gradually increasing the complexity of both the generator and discriminator networks during the training process. In our model, we initiate the procedure by generating small-resolution images (512x4x4) and systematically introduce additional layers to both networks, progressively

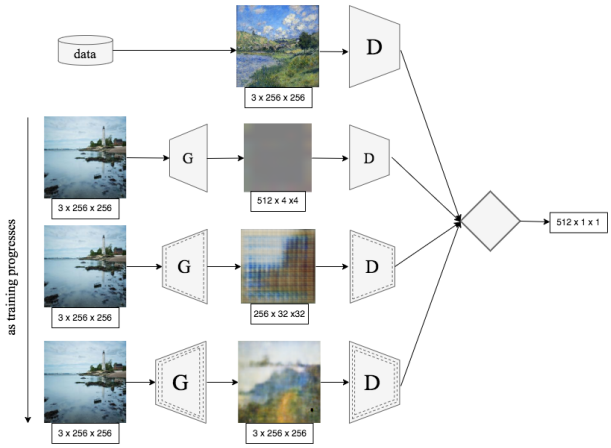


Figure 4. Progressive Growth Mechanism

increasing the image resolution at each stage until we reach the original image size (3x256x256), as depicted in Figure 4. Furthermore, the discriminators now produce a 512x1x1 matrix of scores. This incremental approach allows for stable training and enables the capturing of intricate details in the generated images. However, it is worth noting that our current implementation employs a simplified approach, involving symmetric layer increments and decrements, which led to the exclusion of UNet and PatchGAN architectures and may contribute to the observed significant loss of accuracy (see Figure 6a). To achieve remarkable outcomes in terms of hyper-realistic image generation and to explore further possibilities in advanced image style transfer techniques, we believe refining and expanding upon this process in the future would be highly beneficial.

3.6. A New Architecture - LS-DC-D2 CycleGAN with DiffAug

One of the goals of our project was to do something new in the space. Because it seemed impossible to come up with a new architecture in the span of a few weeks, we opted to combine a variety of advanced techniques and explore if their advantages could work together to produce superior performance.

To accomplish this, we began by reading a variety of papers outlining advancements in the training of GANs. While many techniques stood out, it was clear that our particular problem had an issue that we could work to solve. Recall that while the dataset contained 7028 Photos, it only had 300 Monet Paintings. With such a low number of images from one of our domains, we focused on papers aiming to solve the problem of Data Efficient learning.

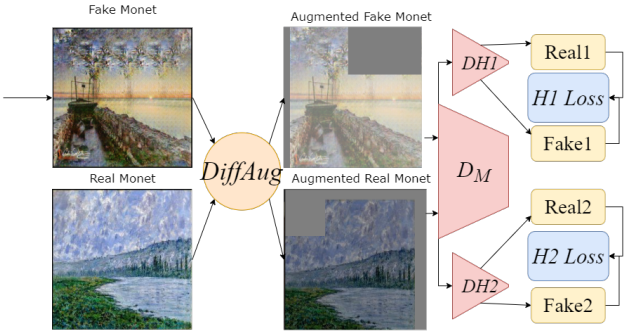


Figure 5. Dual-Objective Discriminator with Differential Augmentation

3.6.1 Differential Augmentation

The issue which such little data in one of the domains is that the discriminator is able to memorize the dataset. If this is the case, the discriminator has then only learned features about the exact images presented, and the generator will then be forced to transform images to the dataset rather than complete true style transfer. Zhao et al. tackle just this in "Differentiable Augmentation for Data-Efficient GAN Training" [13]. Applying their findings to our own paper, we introduce a Differential Augmentation block right before the Monet Discriminator. This takes in the Generated Monet and the Real Monet and performs random augmentations such as random changes in brightness, saturation, and contrast, as well as random translations and cutouts. Examples of these augmentations are shown in Figure 5. By introducing these augmentations for every image that is sent to the Monet Discriminator, memorization is prevented and the Discriminator is able to learn more valuable features that are then passed to the generator. The implementation as well as more visual examples of Differential Augmentation can be found in two of our public Kaggle submissions [1] [2].

3.6.2 Dual-Objective Discriminator

After successfully implementing Differential Augmentation, we wanted to take it further with another complex strategy. After some searching, we found the paper "Dual Discriminator Generative Adversarial Nets" (D2GANs) by Nguyen et al. that proposed splitting the discriminator into two heads with separate loss functions. This acts as regularization for the discriminator, as there are now two varying objectives that simultaneously prevent memorization of the dataset while providing more feedback for the generator to learn. Theoretically, the generator would learn to produce better results, as it needs to trick both heads of the discriminator.

While this paper explored isolated D2GANs, (without

LS, DC, Cycle...), we worked to use the implementation as motivation and build it into our baseline architecture. Figure 5 shows this architecture of the now LS-DC-D2 CycleGAN. Essentially, there is a shared discriminator body that performs the first series of downsamples to produce intermediate discriminator scores, before sending them to the two heads. The two heads will then receive the intermediate scores from both the augmented fake Monet, and the augmented real Monet, and then perform their own computations to calculate the loss. In our implementation, we kept Head 1 the same as the Photo Discriminator using mean squared error, but for Head 2 we used Binary Cross Entropy. Finally, we sum the loss of the two heads to produce the total Monet Discriminator loss before performing the same backpropagation as the initial model.

3.7. Evaluation

While we can work to evaluate our models subjectively by exploring how realistic the output generation is, or objectively with the losses, Kaggle provides its own evaluation and score for the competition. This utilizes the MiFID (Memorization-informed Fréchet Inception Distance), or minimum cosine distance of all training samples in the feature space, averaged across all user-generated image samples [8]. This evaluation provides clear performance metrics to compare our MonetGAN's performance directly with other competitors' models as well as the different versions of our own.

4. Experiments

4.1. Kaggle Submissions & Colab

The majority of our experiments ran in the form of Kaggle competition submissions. The competition had a clear set of rules with a GPU runtime limit of 5 hours, and the networks were required to be trained from scratch. The exact output submission as well as loss graphs and output results can be seen in the linked submissions [3] [1] [2]. This includes the final visualization of the Monet's generated, as well as Photos generated from Monet's and discriminator scores.

However, due to time and GPU limitations, we decided not to submit the Progressive Growth Mechanism to Kaggle. Instead, using Colab we trained it using 50% of our database for a duration of 50 epochs, which took approximately 4 hours.

4.2. Results

Figure 6a contains a compilation of example Photos from the Dataset along with the generated results from each of our models along with a corresponding MiFID evaluation score (lower is better).

4.2.1 MonetGANv1

With lots of trial and error, MonetGANv1 was our first successful competition submission and formed the baseline results for our project. This earned an MiFID evaluation score of 48.06251 and produced good-looking results. This can be seen referencing Figure 6a and as well as the submission notebook which shows the generated Monet Paintings from Photos and generated Photos from Monet's [3]. Additionally, the loss graphs in Figure 7 are reasonable. After the first epoch, both the discriminator and generator losses drop drastically, then seem to bounce up and down for the rest of the training. This is expected because the losses depend on each other so flat loss graphs are not indicative of the stopping of learning, but rather the parallel improvement of generators and discriminators. While not great for evaluation, we include them to visualize the training process and ensure that everything proceeds as expected. This same pattern is repeated in future loss graphs.

4.2.2 MonetGANv2

The evaluation of our model using the ResNet generator and data augmentation yielded very interesting outcomes. Visually inspecting the generated images (Figure 6a), we observed a remarkable resemblance to paintings, capturing some key features of Monet's artistic style. The generated images exhibited characteristic brushstrokes, and a sense of artistic expression, showcasing the model's ability to mimic the textures present in Monet's paintings.

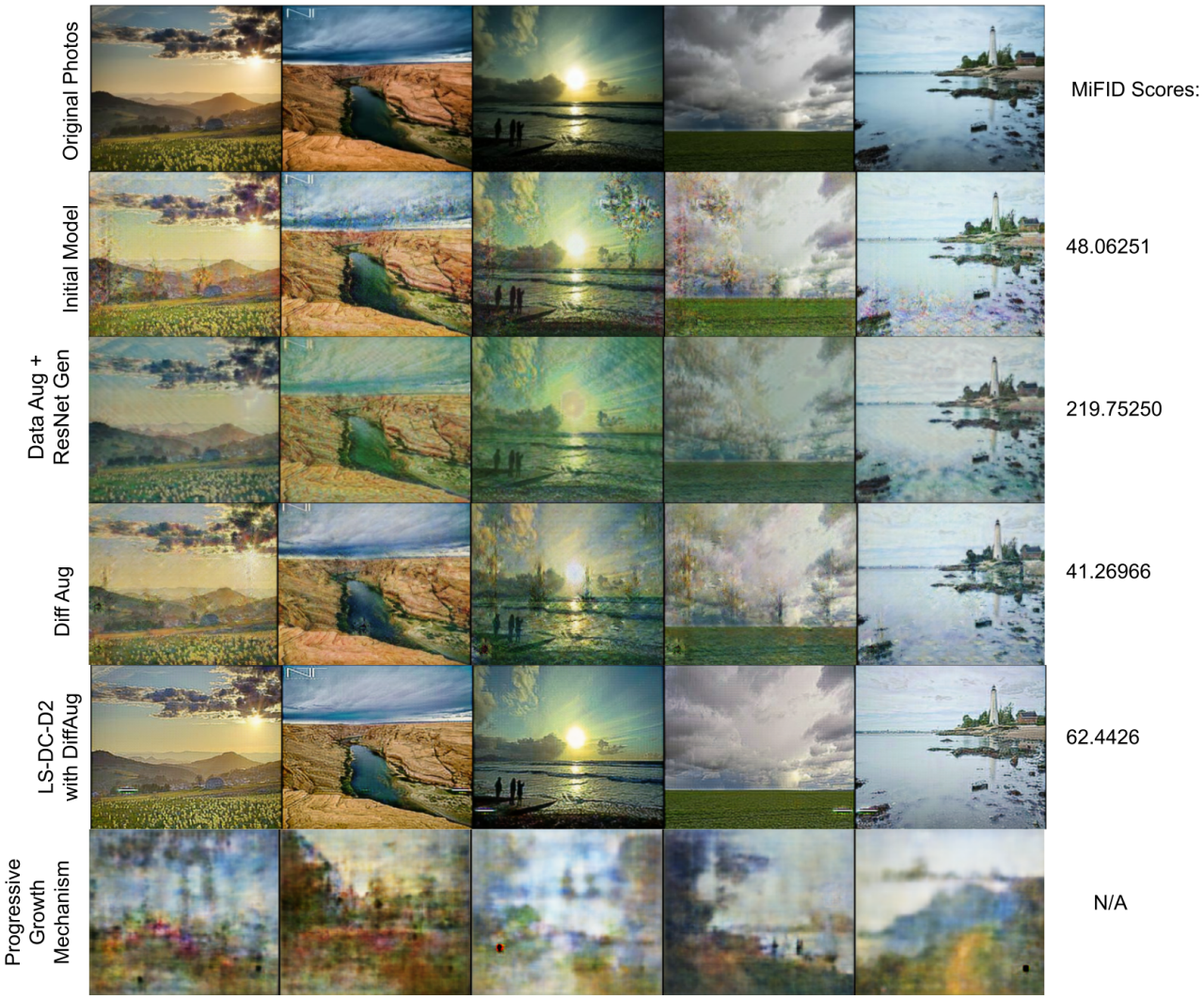
However, when our model was evaluated using the MiFID metric on Kaggle, it obtained a score of 219.75250, which is drastically higher than the baseline model's score of 48.06251, indicating a much worse performance.

4.2.3 Progressive Growth Mechanism

As mentioned earlier, the simplified approach we employed resulted in a significant loss of accuracy compared to the baseline model's performance. Analyzing the loss functions of the generator and discriminator (Figure 9), we observe that they follow the typical patterns of GAN training, but even after 50 epochs, the generator's loss remains considerably high.

4.2.4 LS-DC-D2 CycleGAN with DiffAug

Finally, the LS-DC-D2 CycleGAN with DiffAug achieved a MiFID score of 62.4426. While its evaluation was much better than that of MonetGANv2, its results looked worse, and its evaluation score was still worse than the initial model. Looking at the outputs in Figure 6a, it seems that it learned to copy over the original photos with weird yellow horizontal lines in certain positions. Looking at the loss



(a) Generation Results after training with MiFID scores.

graphs in Figure 10, it seems that the Generator was still learning new features after 25 epochs, but the Monet Discriminator flattened out instantly. This is problematic because if our discriminator can't distinguish real from fake, then the generator isn't going to be able to learn from it in the first place.

With this in mind, we submitted a final experiment retaining our Differential Augmentation but reverting the Dual-Discriminator back to the initial model. This worked quite well, producing an evaluation score of 41.26966 and placing us around number 19 on the leaderboard [1]. Additionally, the results looked objectively the best with subjectively smoother brush strokes and fewer impurities.

5. Discussion

In this study, we presented MonetGAN, an LS DC CycleGAN designed to generate high-quality Monet-style images. After achieving promising results with a MiFID score of 48.06251, we branched off to explore the integration of advanced techniques.

The first of these was adding complexities to the baseline in the form of Data Augmentation and ResNet Generators. While solid visual results, it received a poor MiFID score of 219.75250. We believe that this is due to the addition of the ResNet generator. With the removal of long skip connections, there is potential for reduced gradient flow resulting in worse generator learning. Additionally, the use of data augmentation may have introduced variations that deviate from Monet's style, teaching the generators to learn different features than intended.

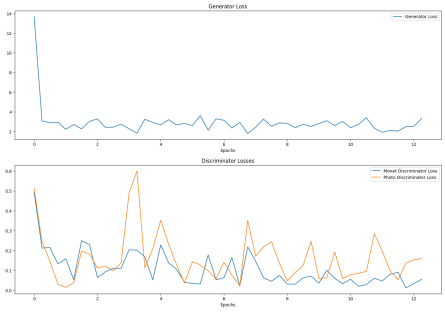


Figure 7. MonetGANv1 Loss Graphs

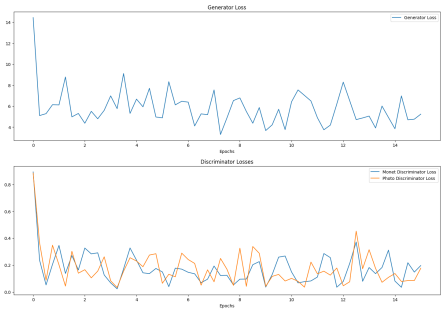


Figure 8. MonetGANv2 Loss Graphs

Next, we found promising results from the Progressive Growth Mechanism. This produced the most unique results with mesmerizing artistic outputs. However, they were all quite far from the initial input images. We believe this discrepancy may be attributed to the increased resolution of the generated images during training, requiring more time to achieve meaningful results. We hypothesize that training the model on the full dataset for 100 epochs would yield exceptional outcomes, enabling a fair comparison with the baseline model and achieving a favorable MiFID evaluation score.

Finally, we worked to create a novel architecture by combining the advanced techniques of Differential Augmentation and Dual-Objective Discriminators. Similarly to MonetGANv2, we were surprised to see worse results than the baseline model despite the added complexity. After an analysis of the architecture and the loss, we found that adding too many varying objectives was hurting the model rather than helping, and the solution was to add one complexity at a time. This led to our final result of the Differential Augmentation model which surpassed our initial performance and had the best results.

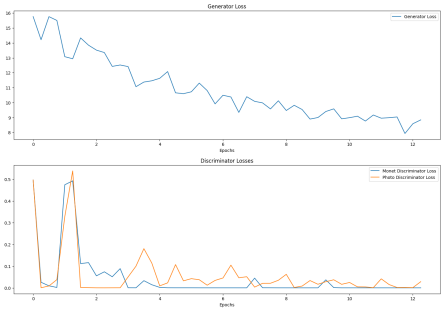


Figure 9. Progressive Growth Mechanism Loss Graphs

6. Conclusion

Overall, through our research, we found that the process of training CycleGANs is quite complicated and involves balancing a variety of varying objectives. We learned that when working to improve a model, it is important to make small changes to gradually build up a successful model, rather than adding many complexities all at once. Additionally, we learned the value of saving models during training, taking into account GPU usage, and how to successfully complete a Kaggle competition.

In the end, we are left with vast experience in training models and success in meeting our goal with the creation of MonetGAN - a robust and reliable competitive model leveraging the latest advances in the field to generate realistic style-paintings in the style of Claude Monet.

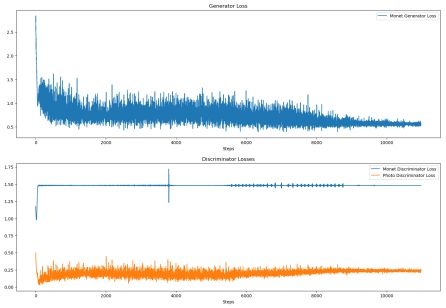


Figure 10. LS-DC-D2 CycleGAN with DiffAug Loss Graph

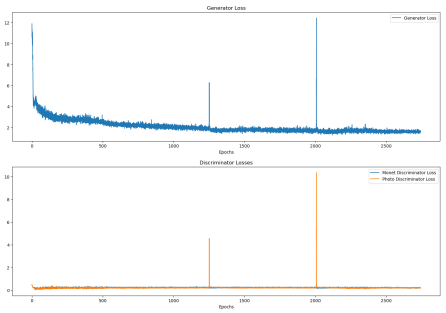


Figure 11. DiffAug on MonetGANv1 Loss Graph

References

[1] Garrett Devereux, Deekshita Doli, and Begoña García Malaxechebarría. MonetGAN - DiffAug, 2023. https://www.kaggle.com/code/garrettdevereux/uw-deep-learning-diffaug-dc-cyclegan?scriptVersionId=131758181.

[2] Garrett Devereux, Deekshita Doli, and Begoña García Malaxechebarría. MonetGAN - LS-DC-D2-CycleGAN, 2023. https://www.kaggle.com/code/garrettdevereux/uw-dc-d2cyclegan?scriptVersionId=131778413.

[3] Garrett Devereux, Deekshita Doli, and Begoña García Malaxechebarría. MonetGANv1, 2023. https://www.kaggle.com/code/garrettdevereux/uw-deep-learning-monetganv1.

[4] Tu Dinh Nguyen, Trung Le, Hung Vu, and Dinh Phung. Dual Discriminator Generative Adversarial Nets. *arXiv e-prints*, page arXiv:1709.03831, Sept. 2017.

[5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A Neural Algorithm of Artistic Style. *arXiv e-prints*, page arXiv:1508.06576, Aug. 2015.

[6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1406.2661, June 2014.

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *arXiv e-prints*, page arXiv:1611.07004, Nov. 2016.

[8] Amy Jang, Ana S. Uzsoy, and Phil Culliton. I'm Something of a Painter Myself, 2020. https://kaggle.com/competitions/gan-getting-started.

[9] Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1812.04948, Dec. 2018.

[10] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least Squares Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1611.04076, Nov. 2016.

[11] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1511.06434, Nov. 2015.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv e-prints*, page arXiv:1505.04597, May 2015.

[13] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable Augmentation for Data-Efficient GAN Training. *arXiv e-prints*, page arXiv:2006.10738, June 2020.

[14] Sharon Zhou, Mitchell L. Gordon, Ranjay Krishna, Austin Narcomey, Li Fei-Fei, and Michael S. Bernstein. HYPE: A Benchmark for Human eYe Perceptual Evaluation of Generative Models. *arXiv e-prints*, page arXiv:1904.01121, Apr. 2019.

[15] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv e-prints*, page arXiv:1703.10593, Mar. 2017.