CS F437: Generative AI

Course Assignment

On

Stylized Image-to-Image Translation

Authors:

Ayush Gupta
2020B3A70838P

Ria Shekhawat
2020B4A71986P

Under the supervision of

Prof. Poonam Goyal

Professor, Department of Computer Science & Information Systems,
BITS Pilani, Pilani

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

April, 2024

Table of Contents

Introduction

The task was to create a system for transforming images from one artistic style to another, specifically from Impressionist to Cubist. We were expected to train a hybrid model of VAEs and GANs for this task. We chose to use the Bicycle GAN proposed in the research paper, ***Toward Multimodal Image-to-Image Translation*** [1], which consists of two parts, namely conditional VAE-GAN(cVAE-GAN) and conditional Latent Regressor-GAN(cLR-GAN). The overall architecture consists of two cycles, B->z->B' and z->B'->z' and hence the name BicycleGAN. Furthermore, the authors of the paper claim that the full hybrid BicycleGAN does not suffer from mode collapse and has the highest realism score by a significant margin, which further motivated us to choose this model over other VAE-GANs.

We chose the WikiArt dataset to procure impressionist images and generated the labeled dataset. Further sections discuss the methods used to make the complete dataset. Henceforth, in this report, we shall discuss the model architecture and training process concerning Impressionist to Cubist conversion. A similar approach has been followed for Cubist to Impressionist translation and can be trivially understood from the discussion provided.

Considering paired instances from these domains (A∈Impressionist, B∈Cubist) representing a joint distribution **p(A, B)**. The model is trained over such pairs to give a corresponding cubist image B for input A(Impressionist). However, given a new instance A during test time, our model should be able to generate a diverse set of output $\hat{B}$'s, corresponding to different modes in the distribution **p(B|A)**.
We would like to learn the mapping that could sample the output $\hat{B}$ from a true conditional distribution given A, and produce diverse and realistic results.

To do so, we learn a low-dimensional latent space $\mathbf{z} \in \mathbf{R}^z$, which encapsulates the ambiguous aspects of the output mode that are not present in the input image. For example, a sketch of a shoe could map to a variety of colors and textures, which could get compressed in this latent code. We then learn a deterministic mapping **G:(A,z)⇒B** to the output. To enable stochastic sampling, we desire the latent code vector z to be drawn from some prior distribution **p(z)**; we use a **standard Gaussian distribution N(0, I)** in this work.

We further discuss the model's architecture, loss functions, and training strategy.
Finally, we shall discuss the results obtained and the model's performance evaluation.

Dataset Preparation

The WikiArt dataset contains 80,042 images in various styles such as Baroque, New Realism, Impressionism, Cubism, etc. This project aims to translate images from Impressionism to Cubism and vice-versa. Hence, we will be using images of the relevant styles only. The dataset contains 13,060 and 2,235 Impressionism and Cubism images respectively.

Since paired images are required for training, we cannot utilize the arbitrary images in both sets. Hence we shall first curate our dataset using these images. The idea is to use the impressionist images and

find corresponding cubist images that can be paired with them. The two approaches we used for the same are:

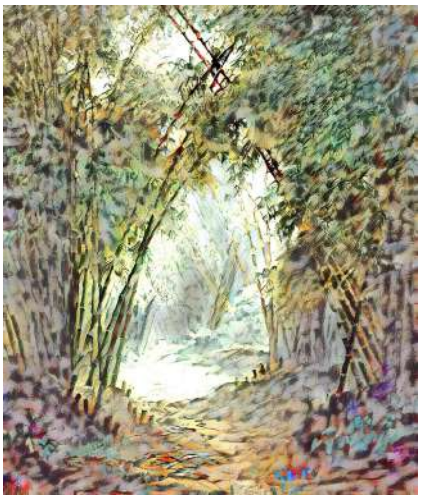
1. Prompt-based image generation using a Stable diffusion model
2. Neural Style-transfer optimized using AdaIN(Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization)

The results obtained from the **Stable diffusion model** were not feasible for our use case since each prompt needed to be fine-tuned for the corresponding image, which is hard to do for larger datasets. To generate prompts we used Salesforce's Blip Large Image Captioning model. We would first caption the image using Blip and then concatenate "depicting cubist artistic style" to the caption and pass it as a prompt to the stable diffusion model. Without tailoring every prompt, we obtained generic results in most cases, which could be improved further to match the input image.
Some of the outputs obtained have been displayed below.

(a) Impressionist image of a forest(left) and corresponding cubist image(right) generated by stable diffusion model

(b) Impressionist image of a lady wearing a hat(left) and corresponding cubist image(right) generated by stable diffusion model

Owing to the lack of similarity and correspondence between the generated and input images, we decided to use **Neural Style Transfer**, as proposed in **AdaIN** [2]. The 2,235 images present in the Cubism style of images in WikiArt were used as the style reference for this model. The results i.e. Cubist images obtained for some of the Impressionist images are shown below.

(c) Impressionist image of a forest(left) and corresponding cubist image(right) generated by AdaIN


(d) Impressionist image of a town square(left) and corresponding cubist image(right) generated by AdaIN

Since 13,060 Impressionist images would require higher storage, we constructed pairs of only 4,500 Impressionist and Cubist images. The dataset was split into train and test sets in an 80:20 ratio, resulting in 3,600 training image pairs and 900 testing image pairs.

Model Architecture

Bicycle GAN as proposed in [1], consists of two parts, namely a cVAE-GAN and a cLR-GAN. It efficiently uses both to train with constraints in both directions, aiming to take advantage of both cycles ($B \rightarrow z \rightarrow B$ and $z \rightarrow B \rightarrow z$). We shall discuss both separately here.

I. A ***cVAE-GAN*** (Conditional Variational Autoencoder - Generative Adversarial Network-) is used to encode the ground truth output image B to latent vector z which is then used to reconstruct the output image B' i.e., B -> z -> B'.

The objective function in this case is:

$$G^*, E^* = \arg\min_{G,E}\max_{D} \mathcal{L}_{\text{GAN}}^{\text{VAE}}(G, D, E) + \lambda \mathcal{L}1^{\text{VAE}}(G, E) + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}(E)$$

The ground truth B is directly mapped with latent code(z) using an encoder E. The generator G then uses the latent code and the input image A to synthesize the desired output $\hat{B}$. The distribution Q(z|B) of latent code z (output of the encoder E) is dealt with a Gaussian assumption, Q(z|B)=E(B).

II. For inverse mapping (z->B'->z'), we use ***cLR-GAN*** (Conditional Latent Regressor Generative Adversarial Networks) in which a Generator is used to generate B' from input images A and z.

The objective function here is:

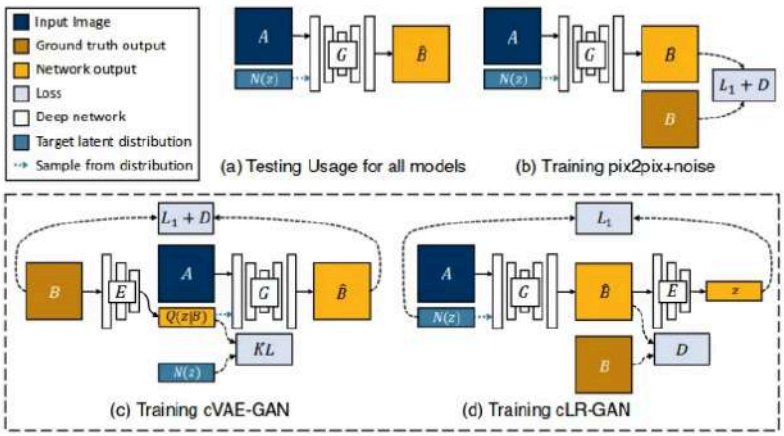
$$G^*, E^* = \arg\min_{G,E}\max_{D} \mathcal{L}_{\text{GAN}}(G, D) + \lambda_{\text{latent}}\mathcal{L}_1^{\text{latent}}(G, E)$$

The discriminator loss LGAN(G, D) on $\hat{B}$ encourages the network to generate realistic results.

Combining both models gives Bicycle GAN with the objective function:

$$G^*, E^* = \arg\min_{G,E}\max \mathcal{L}_{\text{GAN}}^{\text{VAE}}(G, D, E) + \lambda \mathcal{L}_{1^{\text{AE}}}(G, E) + \mathcal{L}_{\text{GAN}}(G, D) + \lambda_{\text{latent}}\mathcal{L}_1^{\text{latent}}(G, E) + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}(E),$$

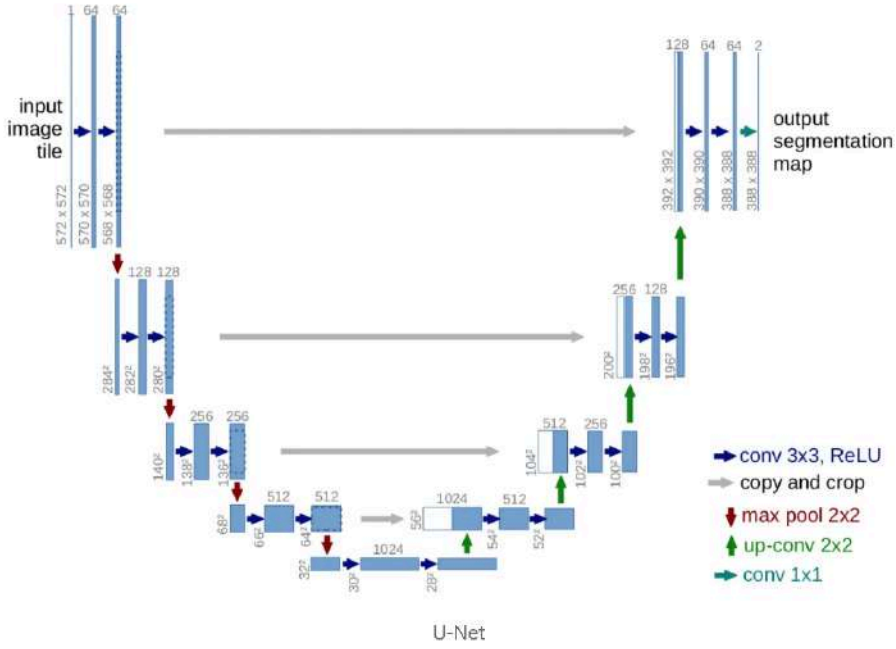
where λ, λlatent and λ□□ are hyper-parameters.



Training process of Bicycle GAN

For generator G, U-Net is used, which contains an encoder-decoder architecture with symmetric skip connections. The architecture has produced strong results in the unimodal image prediction setting when there is a spatial correspondence between input and output pairs.

UNet Architecture:



U-Net

Generally, two PatchGAN discriminators at different scales are used for discriminator D, which aims to predict real vs. fake overlapping image patches. We use multiple discriminators, which return different output sizes (i.e. different local probabilities). In training, the generator needs to fool all the discriminators, and it makes the generator more robust.

For the encoder E, these two networks are preferred:
(1) **E-CNN**: CNN with a few convolutional and downsampling layers.
(2) **E-Resnet**: a classifier with several residual blocks.

The model is trained using Adam optimizer using BatchNormalization with batch size 8. Leaky ReLU activation function is used for all types of networks.

Loss Functions

As discussed, the model architecture has been divided into two parts. Here, we will discuss the loss functions used in each part. Note that A refers to an element from the Impressionist domain, and B is its corresponding Cubist image from the Cubist images domain. A similar discussion can be made for the Cubist to Impressionist translation model.

For the cVAE-LAN, the following loss functions have been used in the objective function discussed previously:

$$\mathcal{L}_{\text{GAN}}^{\text{VAE}} = \mathrm{E}_{\mathrm{A,B}\sim p(\mathrm{A,B})}[\log(D(\mathrm{A},\ \mathrm{B}))] + \mathrm{E}_{\mathrm{A,B}\sim p(\mathrm{A,B}),\mathrm{z}\sim E(\mathrm{B})}[\log(1 - D(\mathrm{A},\ G(\mathrm{A},\ \mathrm{z})))]$$

This is the typical loss function of GAN, where the Generator and Discriminator play a min-max game. Here, the Generator tries to fool the Discriminator, whereas the Discriminator tries to distinguish the images generated by the Generator from the original ones (the true Cubist images).

$$\mathcal{L}_1^{\text{VAE}}(G) = \mathrm{E}_{\mathrm{A,B}\sim p(\mathrm{A,B}),\mathrm{z}\sim E(\mathrm{B})}||\mathrm{B} - G(\mathrm{A},\ \mathrm{z})||_1$$

To encourage the generator's output to match the input and stabilize the training, we use an ℓ1 loss between the output and the ground truth image.

$$\mathcal{L}_{\text{KL}}(E) = \mathrm{E}_{\mathrm{B}\sim p(\mathrm{B})}[\mathcal{D}_{\text{KL}}(E(\mathrm{B})||\mathcal{N}(0,\ I))]$$

The latent distribution encoded by E(B) is encouraged to be close to a random Gaussian to enable sampling at inference time when B is not known. Here, Kullback–Leibler divergence is given as

$$\mathcal{D}_{\text{KL}}(p||q) = -\int p(z)\log\frac{p(z)}{q(z)}dz$$

For the cLR-GAN, i.e. the second part of the cycle in Bicycle GAN, the following loss function has been used:

$$\mathcal{L}_1^{\text{latent}}(G,\ E) = \mathrm{E}_{\mathrm{A}\sim p(\mathrm{A}),\mathrm{z}\sim p(\mathrm{z})}||\mathrm{z} - E(G(\mathrm{A},\ \mathrm{z}))||_1$$

Where $\hat{z}$ =E(G(A, z)) is encouraged to be close to the randomly drawn z to enable bijective mapping.

Training Strategy

The model is built on the Least Squares GANs (LSGANs) variant, which uses a least-squares objective instead of a cross entropy loss. LSGANs produce high-quality results with stable training.
Weight tying has not been applied for the generators and encoders in the cVAE-GAN and cLR-GAN models since using two separate discriminators yields slightly better visual results compared to sharing weights. For the encoder, only the predicted mean is used in cLR-GAN.
We trained the model on the 3,600 image pairs in the training set in the initial experiments by changing hyperparameters and visually evaluating the images obtained.
The learning rates, weights for losses, first-order and second-order momentum decay constants, and other parameters were changed for different training sessions to evaluate differences and choose the optimum values of these parameters. However due to hardware limitations we couldn't train the model long enough to view the differences.

Evaluation Metrics

The model's performance was evaluated using the following strategies:

- Image Quality: Visual inspection of generated images.
- FID Score: Fréchet Inception Distance to measure similarity between generated and ground truth images.

Image quality inspection is done trivially by looking at the images and observing differences or similarities between generated and target images.

FID is one of the most popular metrics for measuring the feature distance between the real and the generated images. Frechet Distance is a measure of similarity between curves considering the location and ordering of the points along the curves. It can also be used to measure the distance between two distributions.
We use the Inception V3 model pre-trained on the Imagenet dataset. First, we extract the features of the generated image and the target image using the Inception v3 model and then compute the mean and covariance using the obtained features. These values are then plugged into the formula to calculate FID and obtain the score.

$$FID = ||\mu_X - \mu_Y||^2 - Tr(\sum_X + \sum_Y - 2\sqrt{\sum_X \sum_Y})$$

Here, X and Y are the real and fake embeddings(activation from the Inception model) assumed to be two multivariate normal distributions. μx and μy are the magnitudes of the vector X and Y.Tr is the trace of the matrix and $\sum X$ and $\sum Y$ are the covariance matrix of the vectors.

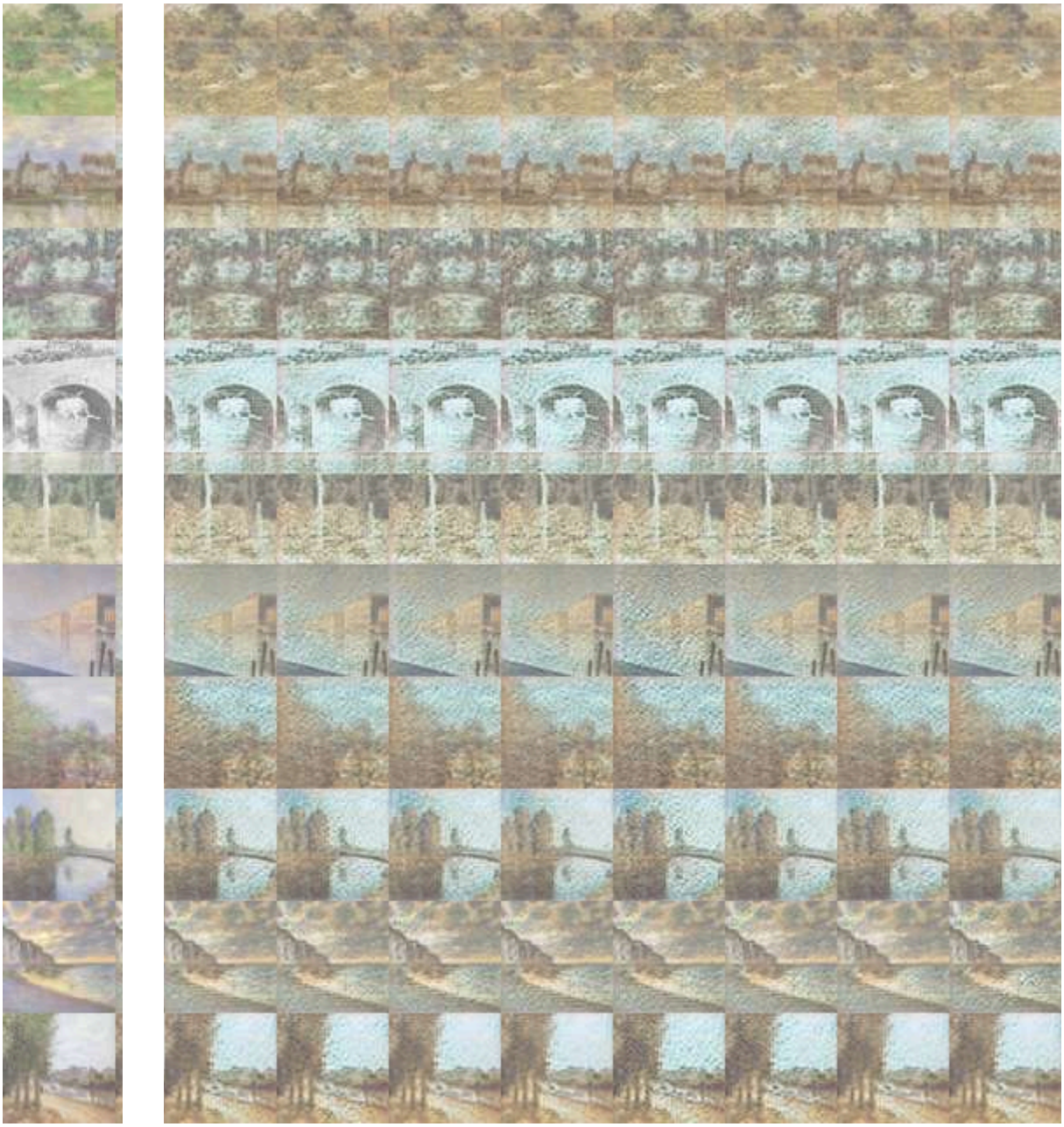
Results and Discussion

We performed 6 experiments on the generated dataset, of which, 4 were to train the model to translate from Impressionist to Cubist images and 2 experiments were done to train the model to translate from Cubist to Impressionist images. The results and inferences for these tests have been discussed in this section.

Specifications: All the models are trained using Ryzen 5 processor (8 GB Memory) and Nvidia GeForce GTX 1650 (4GB Memory).

Experiment 1 (Impressionist to Cubist) : 3600 Image pairs in Training Set
Input size of Image = 128 x 128
Batch size = 32
Model architecture changes = None
Loss values after training for 5 epochs:
KL Divergence Loss: 4.348
VAE Loss: 2.657
Pixel Loss: 0.543
Latent Loss: 0.730
Generator Loss: 6.283

Original Images Generated Images



Inference: The generated images resemble the original images closely as can be visually inferred. The generated images have very poor FID scores in the range of 500 to 700. The model's KL Divergence Loss and Generator Loss are high. This is because of the fact that the model was trained on only 5 epochs due to memory constraints. In the upcoming experiments we decrease the batch size to 8 so that we can train the model for a longer period.

Experiment 2 (Impressionist to Cubist) : 3600 Image pairs in Training Set
Input size of Image = 128 x 128
Batch size = 8
Model architecture changes = None
Loss values after training for 10 epochs:
KL Divergence Loss: 0.324

VAE Loss: 1.453
Pixel Loss: 0.184
Latent Loss: 0.397
Generator Loss: 4.853
Results:

Original Image

(Impressionist)

Target Image

(Cubist)

Generated Images:

FID Scores: 181.945 183.799 153.374 169.999

Original Images

Generated Images

Target Images

The FID scores for the above images vary between 150 and 400.

Inference: The generated images resemble the target images closely as can be inferred from the FID score, and the model's PIxel and Latent losses also converge near zero but generator loss is still high. However, training for more epochs (10) and gaining agreeable images comes at the cost of picture quality since the image size is kept small for this experiment. In further experiments, we shall increase the size and train for fewer epochs since training larger images with more parameters is computationally expensive.

Experiment 3 (Impressionist to Cubist): 1788 Image pairs in Training Set
Input size of Image = 256 x 256
Batch size = 8
Model architecture changes = None
Loss values after training for 10 epochs:
KL Divergence Loss: 3.324
VAE Loss: 1.452
Pixel Loss: 3.124
Latent Loss: 1.23
Generator Loss: 5.614

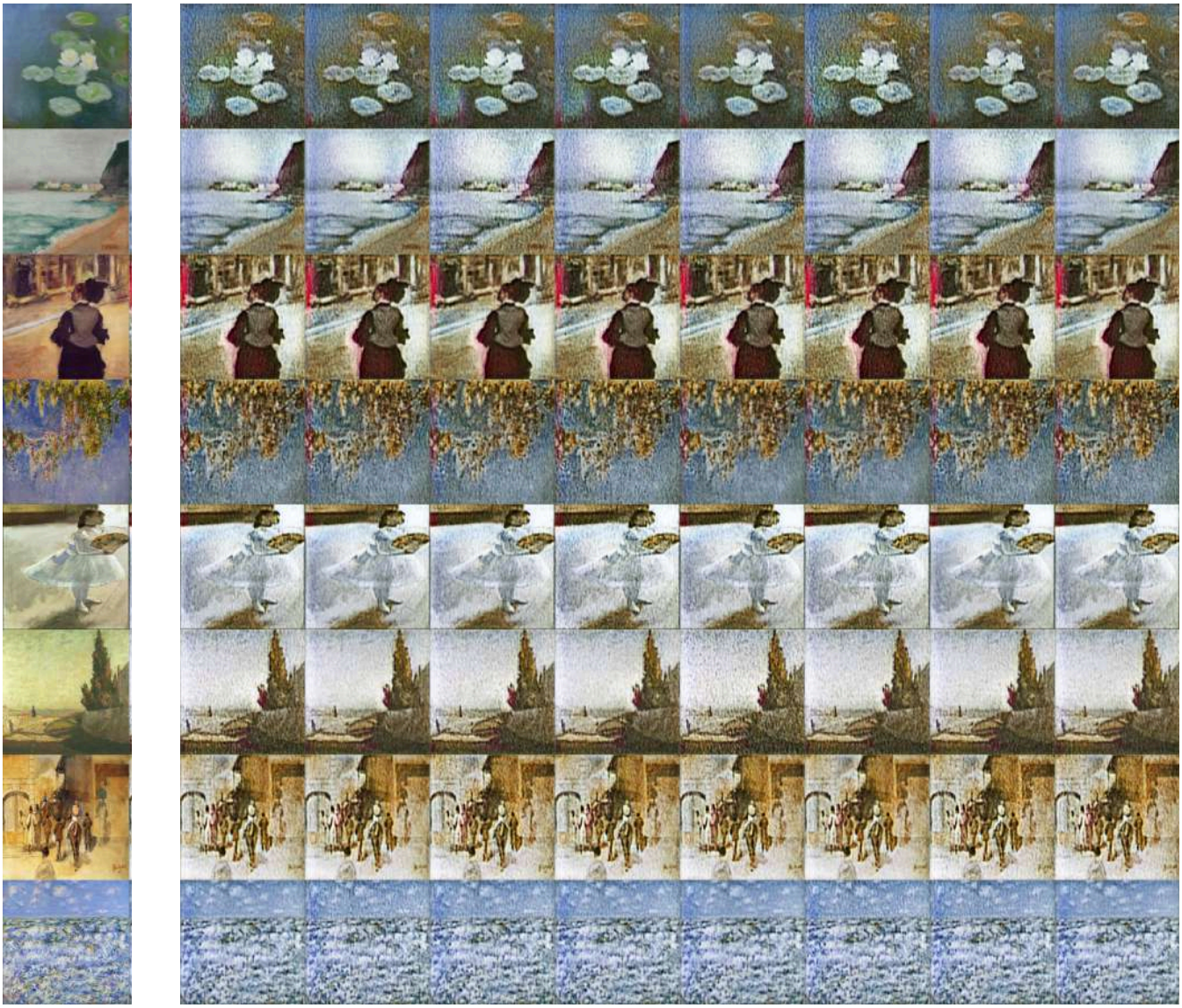
Original Image

Generated Images

Inference: FID scores for the above images vary between 200 and 600. The input image sizes have been increased in this case but it can be observed that loss values are still high for all PIxel, Latent, and Generator losses. This is probably due to insufficient training for the larger number of parameters required in this case.

Experiment 4 (Impressionist to Cubist): 400 Image pair in Training Set
Input size of Image = 256 x 256
Batch size = 8
Model architecture changes = Increased the number of layers in the Encoder by changing the layers from resnet18 to resnet50
Loss values after training for 8 epochs:
KL Divergence Loss: 6.124
VAE Loss: 3.862
Pixel Loss: 2.462
Latent Loss: 2.365
Generator Loss: 6.254

Original Images Generated Images

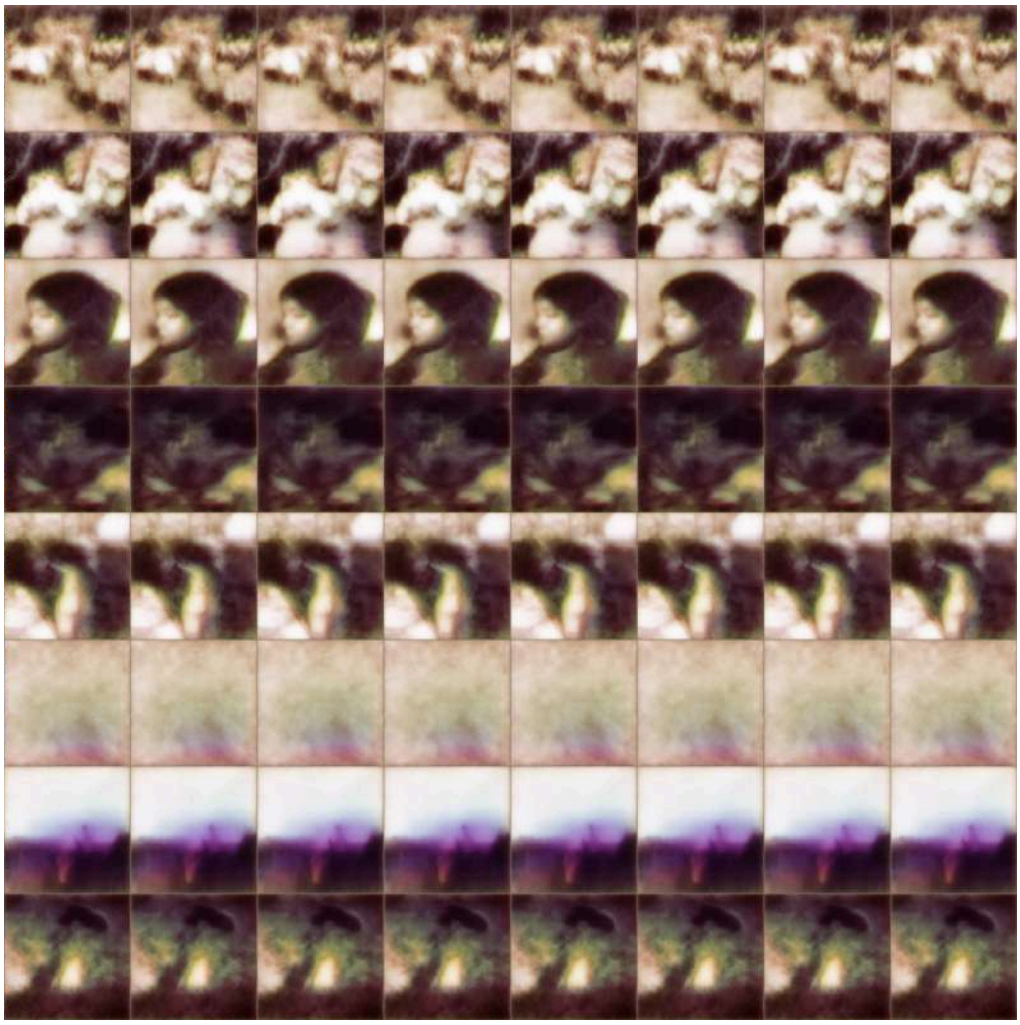


Inference: Training on resnet50 leads to better results because it is deeper(having more layers) than resnet18 and thus training more parameters. However, resnet50 is more computationally expensive and infeasible in this case, given our limited resources.

Experiment 5 (Cubist to Impressionist): 1788 Image pairs in Training Set
Input size of Image = 256 x 256
Batch size = 8
Model architecture changes =None
Loss values after training for 8 epochs:
KL Divergence Loss: 12.885
VAE Loss: 1.216
Pixel Loss: 0.312
Latent Loss: 0.471
Generator Loss: 5.236

Original Images

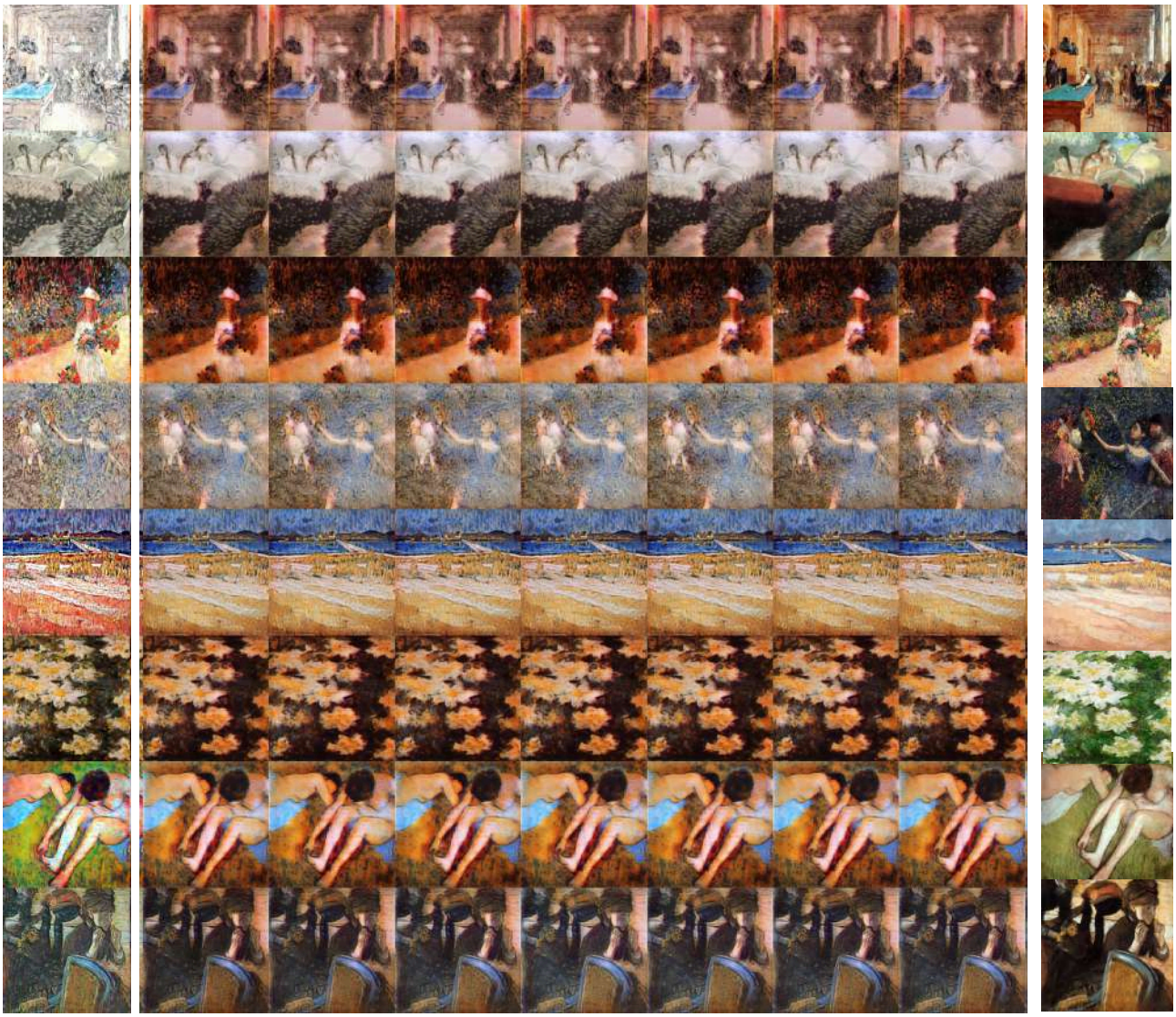
Generated Images



Inference: This experiment trained the model to learn parameters to generate Impressionist images for given Cubist images. Large input image sizes and fewer epochs for training have led to higher loss values and poorer results. The next experiment aims to overcome these limitations.

Experiment 6 (Cubist to Impressionist): 400 Image pairs in Training Set
Input size of Image = 256 x 256
Batch size = 8
Model architecture changes = Increased the number of layers in the Encoder by changing the layers from resnet18 to resnet50
Loss values after training for 20 epochs:
KL Divergence Loss: 1.055
VAE Loss: 1.467
Pixel Loss: 0.241
Latent Loss: 0.410
Generator Loss: 2.345

Original Images Generated Images Target Images



FID Scores:

334.988	278.963	431.125	331.066	187.035	271.377	501.069	186.065

Inference: The loss values have decreased considerably compared to the other experiments since this time we have trained the model for a higher number of epochs. Visually the generated images look good and even the FID scores have improved.

Conclusion: While the images are good representations of the desired style, the FID score is relatively high because the Inception model is trained on Imagenet, which constitutes natural images. In contrast, our model, Bicycle GAN, is trained on the WikiArt dataset. Another possible reason could be insufficient training due to a lack of GPU power and storage space. We have a limitation of 4 GB on the GPU memory and hence cannot load images of large dimensions. Another limitation is that we have used the output of neural style transfer as input images in our model; hence, we do not have the true distribution of cubist images. In some cases, the output of the neural style transfer model is almost the same as the input image with just some color inversion. This hampers the performance of our model. BicycleGAN establishes a one-to-one relationship between the latent space and the output space. It ensures that each output corresponds to a distinct latent code, and conversely, each latent code generates a unique output. This prevents mode collapse, where multiple latent codes might produce identical outputs.This is even visible in our model's results where no two different inputs are generating the same output.

References

1. Zhu, Jun-Yan & Zhang, Richard & Pathak, Deepak & Darrell, Trevor & Efros, Alexei & Wang, Oliver & Shechtman, Eli. (2017). Toward Multimodal Image-to-Image Translation.
2. Xun Huang and Serge Belongie. (2017). Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization