Multi-source Transfer Learning Models: NST and Conditional CycleGAN

Jiankun Wei[1] **Ruiting Chen**[1] **Hanqi Zeng**[1]
Department of Computer Science
University of Toronto
Toronto, ON

Abstract

This paper compares two models for multi style transfer: a pretrained neural style transfer (NST) applied multiple times, and a conditional CycleGAN that includes an encoding label for each artist style. We discuss the effect of several hyperparameters and analyze the output using various metrics.

1 Introduction

This paper investigates two models on the task of multi style transfer. The first model is a baseline model based on pretrained neural style transfer (NST) which applies NST multiple times in a chain on the previous output of NST to add multiple styles. The second model is a modified conditional CycleGAN, where the generator and discriminator networks were modified to include an encoding label for each artist style. The modified CycleGAN was then applied multiple times in a chain similar to NST to achieve multi style transfer. The paper investigates the impact of varying hyperparameters on the training loss of the modified CycleGAN model and compare the output of both models using multiple metrics.

2 Related Works

Prior research in style transfer has mostly focused on single style transfer, with few works exploring multi-style transfer. The first approach incorporates Neural Style Transfer (NST)[6], a technique that has not been extensively explored in previous research. The second proposed approach builds upon the power of Generative Adversarial Networks (GANs)[1], particularly the CycleGAN, to transfer styles between two different image styles. To achieve this, the generator networks of the CycleGAN are modified to take labels and generate twice. While previous works have also explored CycleGAN-based multi-style transfer[4, 3, 5, 8], this approach is unique in its conditional implementation. NST is a powerful deep-learning model that can generate artistic images by combining the style of one image with the content of another. However, when it comes to multi-style transfer, NST has some limitations. The output of applying NST for multi-style transfer often shows that the intermediate features are completely overridden, resulting in a style transfer that only reflects the final style applied. This may be due to the fact that NST mainly focuses on extracting the content of the content image while ignoring its style. As a result, the style assigned to the intermediate steps is lost. To overcome this limitation, we need a different model that can preserve the original style during the style transfer process.

3 Methods and Algorithms

3.1 NST on Multi-style Transfer

We apply the pretrained NST twice and include the styles step by step. Details are in Figure 4.

3.2 Conditional CycleGAN

We borrow the Amy Jiang's cycleGAN architecture and loss[1] with label embedding added. Conditional CycleGAN has potential to retain previous application results while applying style transfer multiple times. In this model, all parameters except the label embedding are shared among multiple applications. The generator networks is a U-net taking in an input imageand a label. Images are processed through sequences of downsampling then upsampling layers, with label embedding mixed in later in downsampling, and output image size is the same as the input image size. (Details see Figure 5) The discriminator networks take in the same inputs, processed through downsampling layers, and output a 30×30 matrix of binary value indicating the real or fake for each portion. (Details see Figure 6) In cycleGAN, we have artist generator (AG), photo generator (PG), artist discriminator (AD), and photo discriminator (PD): the real photo are transfered to fake artwork by AG then back to photo by PG which is criticized by PD. And the real artwork is processed symmetrically. We also let PG and AG to generate photo and artwork directly, and intergrate all these information into loss function (described in appendix). We trained our model on the artwork of 3 artists, first Vangogh, then Monet, then Ukiyoe.

4 Experiment

4.1 Hyperparameters and Training Loss for Conditional CycleGAN

To experiment the effect of hyperparameters on the training result of conditional CycleGAN, we trained the model on four different dataset sizes 50, 100, 150, and 200 each with batch sizes 1 and 5, and we selected dataset size 200, batch size 5. using this, we tested epoch size 30, 60, 90 in total (epoch 10, 20, 30 per artist, for 3 artists).

4.2 Evaluation Metrics

We employ three metrics to evaluate the performance of our image translation model. Firstly, we use the Structural Similarity Score (SSIM) to assess the extent to which the output image retains the content of the input content image. Secondly, we use the Style Reconstruction Loss to evaluate the consistency of the styles of output with the input style image. Thirdly, we use the Frechet Inception Distance (FID)[2] score, a classical metric used to evaluate the overall quality of conditional cycleGAN outputs. By utilizing these three metrics, we aim to provide a comprehensive evaluation of our image translation model.

5 Evaluation and Discussion

5.1 Hyperparameters

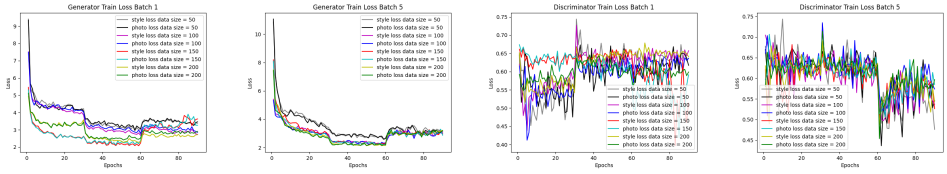

Figure 1: Training Losses

Patterns in loss plots: Discriminator loss and generator losses follow a general trend, that is, a decrease in generator losses is coupled with an increase of discriminator losses, and vice versa, which is as expected given the relation of generator and discriminator networks.
All losses in the generator loss plots show a concave-shaped valley pattern. We think this could be due to the model sequentially fit on 3 datasets each with 30 epochs for each artist. When switching from Vangogh to Monet, the model is still stabilizing (seeing decrease in generator loss and a big fluctuation in discriminator loss), and the style of Monet and Vangogh are similar, so we see a decreasing cliff in generator loss at around epoch 30. However, when switching from Monet to

Ukiyoe, the model already stabilizes and their style are very different, so the generator is not quite confident with the new artist and discriminator can quickly detect the difference, leading to the big drop in the discriminator loss and increase in generator loss at around epoch 60. Within each artist, the generator loss gradually decreases and the discriminator loss gradually increases as the model learns the artist's style better over time.
Since we only use the generator to produce the final output and there is less pattern in the discriminator losses, the following discussion will focus on generator losses.

Batch sizes: Generator loss graphs for both batch sizes 1 and 5, show a similar trend with decreasing loss as epoch number increases. Still, batch size 5 has more stability and robustness with slightly better results. Models trained with batch size 1 can produce funny outputs (Figure 4) possibly due to unstable updates with too specific information on small batches. We have more discussion on this in section 6 Interesting Findings.

Dataset sizes: Within a single generator loss graph on either batch size, the losses all exhibit a similar pattern. However, the green and yellow lines (dataset size 200 per artist) still stand out as having a lower training loss throughout the entire training process, suggesting that larger datasets has enabled the generator to learn more efficiently and achieve better performance. Due to time and resource restrictions, we will use 200 as our data size for our final model.

Epoch sizes:

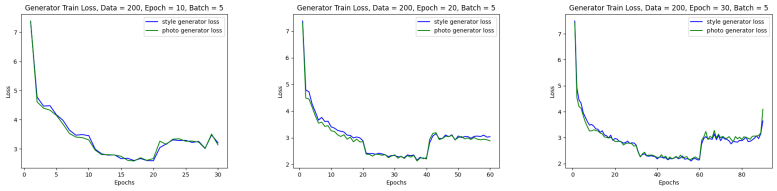

Figure 2: Training losses for different epoch sizes

Following the previous conclusion, we test epoch size on data size 200 (per artist), batch size 5. When using 30 epoch, we can see the model is still stabilizing as the graph fluctuates a lot. Using 60 epoch, we can see the model is still decreasing at the end, and using 90 epoch, the loss starts increasing at the end indicates overfitting. Therefore, we conclude that a better choice of epoch is around 83 to 85.

5.2 Final Output and Metric Evaluation

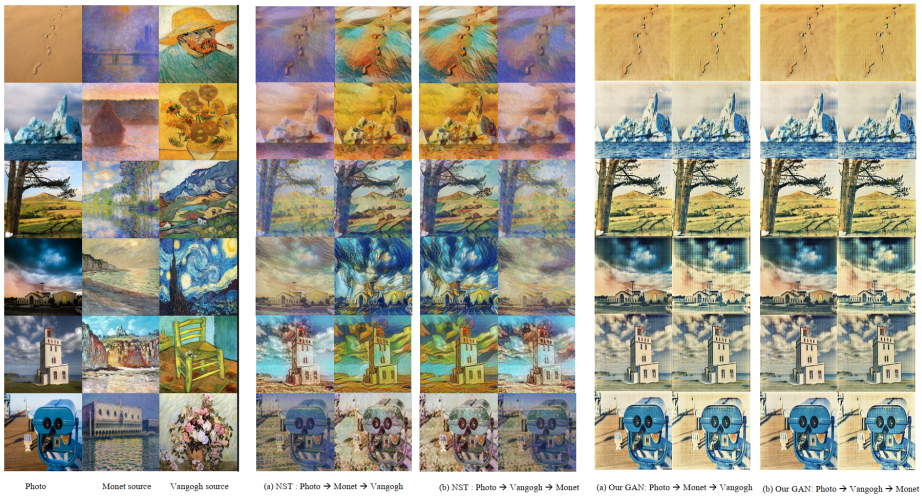

Figure 3: Output Image Comparision

In general, NST output provides high quality style transfer based on the last input artwork, however, the second application of NST washes out the style learned in the first application. This can be shown that the output of applying Monet then Vangogh (Figure 3 NST (a) column 2) is similar to applying Vangogh directly (Figure 3 NST (b) column 1), and vice versa. In addition, NST can only focus on the style of one single image. It cannot inject the general feel of an artist summarized from thousands of artworks into the photo. On the other hand, conditional cycleGAN convert the photo to an image that has a great taste of being an artwork, and we can see a gradual improvement in the sequential applications of the model. However, conditional cycleGAN seems learning only the general idea of being an artwork, instead of the specific style of each artist (Figure 3 our GAN (a) and (b) column 1 looks almost the same). We think this is because currently we concatenated the embedded label for each artist later in downsampling, where only core feature map of image is left. Thus when switching between artists, most weights are preserved and only a small amount of weights are associated with the style of each artist. Hence the output of photo to Monet and photo to Vangogh produce similar output. A further improvement should focus on increasing the parameters in manipulating the labels which will correspond to learning the style of each artist, mixing in the embedded label in earlier layers of downsampling, or training 3 separate cycleGAN one for each artist.

To calculate the style of each artist, we average the output over 10 artworks from each artist.

Metric Summary

	ssim score	Monet style consistency	Vangogh style consistency	fid score
NST photo to monet	0.5956275512774786	3215.4159342447915	3926.185750325521	N/A
NST monet to vangogh	0.46998626987139386	3190.095906575521	3496.750040690104	N/A
NST photo to vangogh	0.5157584498325983	3420.288818359375	3698.6288248697915	N/A
NST vangogh to monet	0.5403685222069422	2834.611043294271	3596.1290690104165	N/A
GAN photo to monet	0.7267969051996866	5668.7100830078125	6052.8751220703125	7.466384703188173
GAN monet to vangogh	0.6464660714070002	5282.1290283203125	5602.1706136067705	8.613811402819152
GAN photo to vangogh	0.7267969151337942	5668.1846923828125	5605.2952067057295	7.466447195490319
GAN vangogh to monet	0.6464698612689972	5284.446614583333	5604.3223063151045	8.614357750548924

General Trend

Within GAN outputs, we did not see much difference in scores between each pair of applications. This means that even the first application of GAN model already set a good grounds for the output: it well preserved the input content and already set the grounds for style transfer. This also implies that GAN models are commutitive meaning apply whichever style first does not seriously affect the output of the model output.

Within NST outputs, injecting Monet style into Vangogh style degrade metrics while injecting Vangogh style into Monet style improve metrics. This aligns with our intuition as Vangogh is more objective and focus more on the exact appearance of content/object, whereas Monet and Impressionism in general is subjective and focus on imagination, or "the colour within his head".

Structual Similarity

The structural similarity of GAN outputs are higher than that of NST output, meaning that GAN output will preserve more information of the original input than the NST output. Also applying model twice usually have a lower SSIM score than applying once, showing each model application replace some content with style.

Style Reconstruction Loss

The score of NST output is in general smaller than that of GAN output, meaning NST indeed mixed more of each artists' style into the picture. And higher score for GAN signifies that GAN did not quite focus on the specific style of artists but instead on the general characteristic of artworks in late 19th century.

Applying either model twice is lower in style loss than applying only once, showing that each application of NST or GAN inject some general idea of style into the image.

Also for both NST and GAN, the Monet style score are smaller than Vangogh style score, this makes sense because Monet is more stable in terms of drawing techniques and colour usage. Whereas Vangogh has a wider variety due to his rough path of life. Hence models can learn Monet style easier than Vangogh style. This is also justified by applying Monet style first then Vangogh generally result in a higher score.

FID

FID score is between 0 and ∞. The lower the FID score, the better the output. Since all our scores are below 10, our overall quality are all very well.

If we apply GAN twice, the FID score increases, signifying that we are sacrificing some quality to include more style.

6 Interesting Findings and Conclusion

6.1 Funny Results when Batch Size = 1

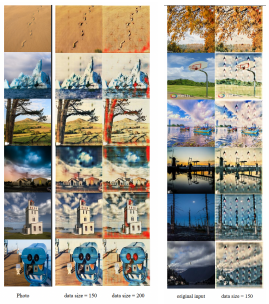

Figure 4: Abnormal Output(batch size=1)

When batch size is 1, we see funny and sometimes creepy outputs even though we train with large data size and many epochs. These include red eyes, small patches of blackness, and repeated nonsense patterns.
We hypothesize that these exists because when batch size is 1, each update is only focused on information of one photo, which can lead the model to learn abnormal/too specific information, leading to unstable updates and overfitting.
When we switch to batch size 5, the model is able to see more diverse and representative data in each updates. This can help stabilize the training process and reduce the noise in the gradients, leading to better results.
Interestingly, in the original model trained by Amy Jiang, her output also suffers from repeating nonsense patterns/shadows because it is trained on batch size 1. In addition, her model was trained on a total of 25 epochs, which is insufficient. Therefore, her output see almost no difference from her input photo. These problem might be overcomed by using batch size 5 with more epochs, ie. 85 epochs.

6.2 Conclusion

7 Acknowledgement

We give thanks to Abhishek Moturu for early discussions, and providing helpful comments and suggestions throughout the project. Special thanks to Jimmy Ba and Bo Wang for helping us incorporate important responsible AI practices around this project. We appreciate the dataset sourced from the UC Berkeley CycleGAN Directory authored by Taesung Park[7] and TensorFlow for providing the pretrained models. We give special thanks to Amy Jiang for providing cycleGAN architecture that grounds our model.

References

[1] Amyjang. "Monet CycleGAN Tutorial". In: *Kaggle* (Aug. 29, 2020). URL: https://www.kaggle.com/code/amyjang/monet-cyclegan-tutorial/notebook.

[2] Jason Brownlee. "How to Implement the Frechet Inception Distance (FID) for Evaluating GANs". In: *MachineLearningMastery.com* (Oct. 10, 2019). URL: https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/.

[3] Saul Dobilas. "cGAN: Conditional Generative Adversarial Network — How to Gain Control Over GAN Outputs". In: (Oct. 15, 2022). URL: https://towardsdatascience.com/cgan-conditional-generative-adversarial-network-how-to-gain-control-over-gan-outputs-b30620bd0cc8.

[4] Konstantin Dobler et al. "Art Creation with Multi-Conditional StyleGANs". In: *arXiv (Cornell University)* (Feb. 23, 2022). DOI: 10.48550/arxiv.2202.11777. URL: http://arxiv.org/pdf/2202.11777.

[5] *keras-io/CycleGAN · Hugging Face*. URL: https://huggingface.co/keras-io/CycleGAN.

[6] "Neural style transfer". In: *TensorFlow* (). URL: https://www.tensorflow.org/tutorials/generative/style_transfer.

[7] Taesung Park. *CycleGAN*. July 30, 2020. URL: https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/.

[8] Ching-Ting Tu, Hwei-Jen Lin, and Yu-Tsung Tsia. "Multi-style image transfer system using conditional cycleGAN". In: *The Imaging Science Journal* 69.1-4 (Feb. 1, 2021), pp. 1–14. DOI: 10.1080/13682199.2020.1759977.

Appendix

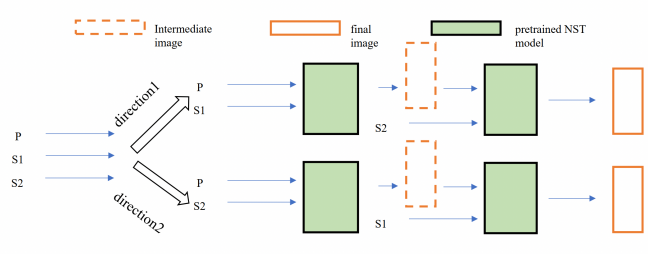

Figure 5: Overview of our NST on Multi-style transfer Model where NST pretrained models sourced from tensorflow[6].

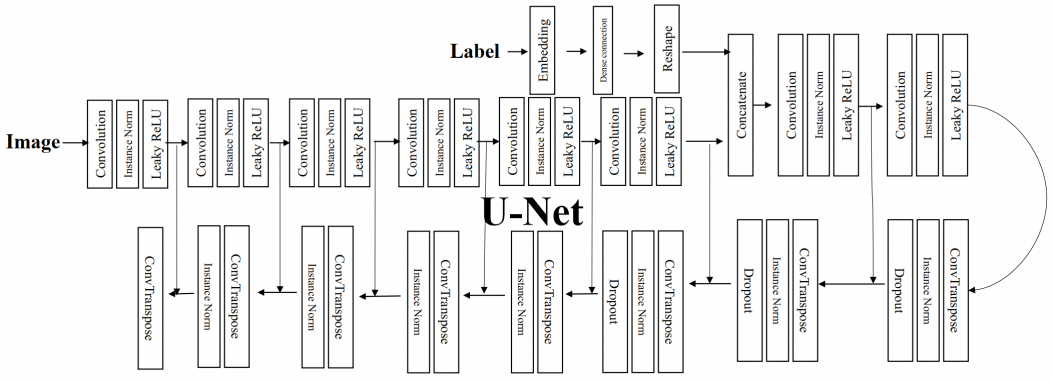

Figure 6: Overview of conditional CycleGAN generator encoder-decoder

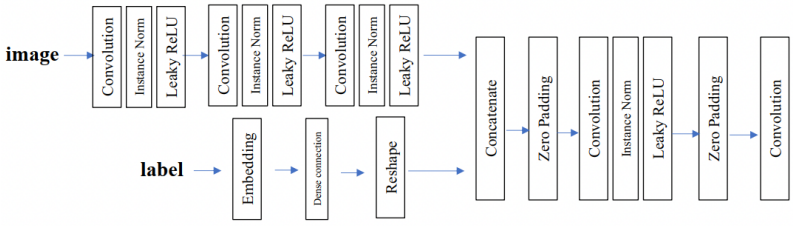

Figure 7: conditional CycleGAN discriminator architecture

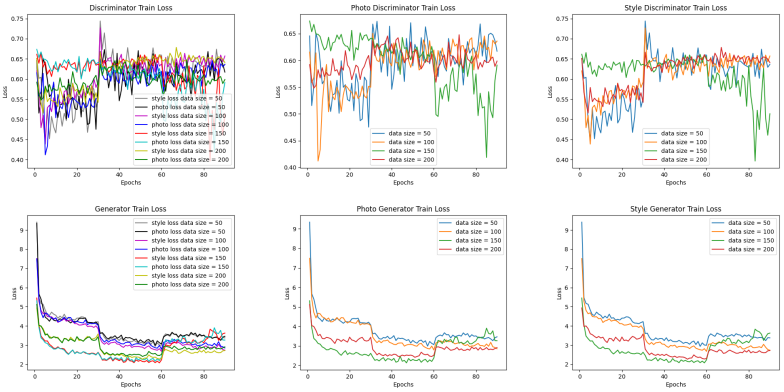

Figure 8: Other supplementary loss plots for Batch Size = 1

loss function

The losses for the Artist Generator, Photo Generator, Artist Discriminator, and Photo Discriminator are defined as follows:

$$1.\text{Artist Generator Loss} = generator\_loss(disc\_fake\_artist) + cycle\_loss(real\_artist, cycled\_artist) + cycle\_loss(real\_photo, cycled\_photo) + identity\_loss(real\_artist, same\_artist)$$

$$2.\text{Photo Generator Loss} = generator\_loss(disc\_fake\_photo) + cycle\_loss(real\_artist, cycled\_artist) + cycle\_loss(real\_photo, cycled\_photo) + identity\_loss(real\_photo, same\_photo)$$

$$3.\text{Artist Discriminator Loss} = discriminator\_loss(disc\_real\_artist, disc\_fake\_artist)$$

$$4.\text{Photo Discriminator Loss} = discriminator\_loss(disc\_real\_photo, disc\_fake\_photo)$$

There are four fundamental losses (discriminator, generator, cycle,indentity) defined in our model. After that, Generator Loss and Discrimator Loss for artist and photo respectively are computed.

real-photo $\xrightarrow{s\ \text{gen}}$ fake-artist $\xrightarrow{p\ \text{gen}}$ cycle-photo $\xrightarrow{p\ \text{disc}}$ disc-fake-photo

real-artist $\xrightarrow{p\ \text{gen}}$ fake-photo $\xrightarrow{s\ \text{gen}}$ cycle-artist $\xrightarrow{s\ \text{disc}}$ disc-fake-artist

real-photo $\xrightarrow{p\ \text{gen}}$ same-photo $\xrightarrow{p\ \text{disc}}$ disc-real-photo

real-artist $\xrightarrow{s\ \text{gen}}$ same-artist $\xrightarrow{s\ \text{disc}}$ disc-real-artist.

Multi-style transfer aims to generate images that are both artistically stylized and photo-realistic. Let x be the real image, y be the generated image, G be the generator, D be the discriminator, and $\hat{x}$ be the reconstructed image. Then the loss functions can be defined as:

Discriminator loss: $L_D(x, y) = \frac{1}{2}[\text{BCE}(D(x), 1) + \text{BCE}(D(y), 0)]$
Generator loss: $L_G(y) = \text{BCE}(D(y), 1)$
Cycle consistency loss: $L_{cyc}(x, \hat{x}) = \mathbb{E}x \sim pdata(x)[||x - \hat{x}(G(y))||_1]$
Identity loss: $L_{id}(x) = \mathbb{E}x \sim pdata(x)[||x - G(x)||_1]$

where BCE is the binary cross-entropy loss, $pdata(x)$ is the data distribution, and $\mathbb{E}$ denotes the expected value. These loss functions are used to train the generator and discriminator networks to produce high-quality stylized images that are both artistically pleasing and photo-realistic.