CIS 5810 Final Project Report

Project Title:
Exploring Image Style Transfer: From Neural Style Transfer to CycleGAN

Group Members: Yicheng Xia, Songchen Xie, Lei Sun

December 15, 2023

Abstract

In this project, we implement two deep learning based methods for image style transfer. We start from using Neural Style Transfer (NST) to blend styles and contents from any two images and extend its use on video style transfer by enforcing consistency constraints. We then implement CycleGAN for unpaired image and video style transfer and conduct extensive test on the designs of the model to explore more characteristics of this method. We have promising results in both image and video generating and in both NST and CycleGAN.

1 Introduction

Our project is based on track 2: *Image Style Transfer*. In regards to this topic, there are two main components: adding the style, and preserving the content. To achieve this, there are two methods that have been very popular: Neural Style Transfer (NST) and Cycle-Consistent Adversarial Networks (CycleGAN). Although they have both been used for style transfer, they vary significantly in their approaches and generated results. In this project, we study each of them in great detail, find ways to improve them, and identify their strengths and weaknesses.

2 Related Works

Neural Style Transfer (NST), introduced in 2016, utilizes deep neural networks to blend the artistic style of one image with the content of another, resulting in a unique, composite image. CycleGAN, on the other hand, is a special type of Generative Adversarial Network (GAN), which employs two pairs of generators and discriminators. One generator G_{AB} takes inputs from domain A and generate outputs that mimic data from domain B, and the discriminator D_B tries to distinguish $G_{AB}(a \in A)$ from data in B. Similar for the other pair G_{BA} and D_A. Adversarial training will be conducted to make the generator better at creating 'fake' results that mimic the target domain and 'fooling' the discriminator.

The main feature of CycleGAN is the 'Cycle Consistency' lies in that: it constraints $G_{BA}(G_{AB}(a \in A))$ to be close to a, and $G_{AB}(G_{BA}(b \in B))$ close to b, such that in an intuitive way, the mapping is typically invertible, and the content is preserved.

As for video style transfer (or other characteristic translation), we need to beware of the fact that the generated scene does not change a lot across consecutive frames. In order to retain high consistency for videos, we may refer to the loss function applied in [3].

3 Methodology

Our image style transfer method consists of 2 parts: CNN-based Neural Style Transfer (NST) and CycleGAN-based style transfer.

3.1 Neural Style Transfer

3.1.1 Image Style Transfer.

We adopt the pipeline from the NST paper [2], which employs a normalized VGG-19 network [4] for deep image representation as the feature space, uses layered squared-error loss between feature representations for content, and adopts a special feature space [1] to capture texture information and Gram matrix to quantify the feature correlation as the basis for style content loss. The full style transfer pipeline is displayed in Fig. 1.

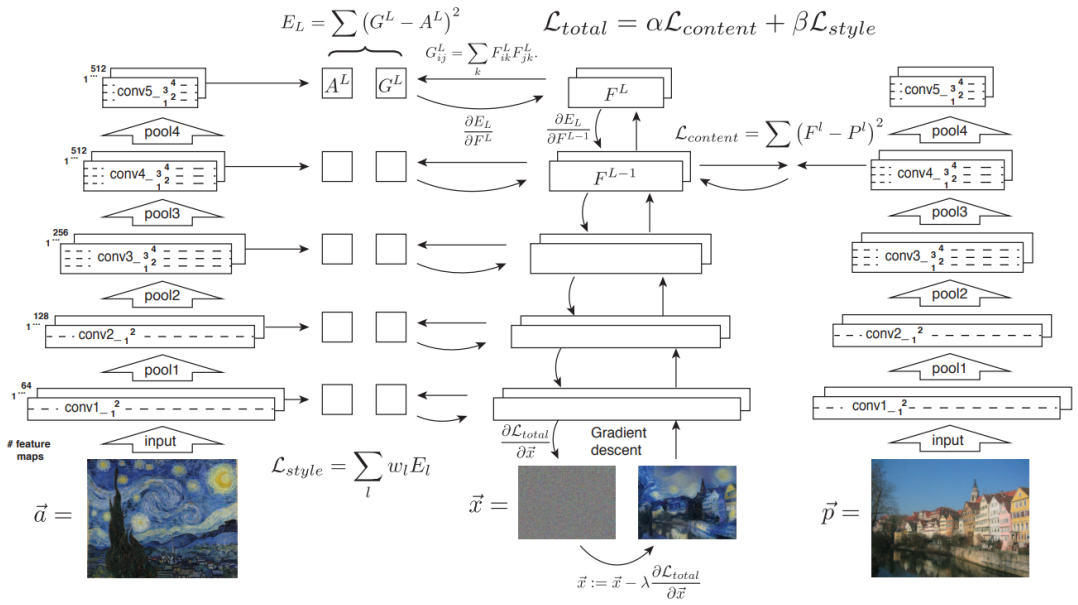

Figure 1: Demonstration of the CNN-based Neural Style Transfer algorithm [2]

Since this CNN model is already well-tuned by the authors, whenever we experiment with other architectures or different hyper-parameters, the image style transfer results will be compromised and deteriorated. Thus, we decide to keep the model and the parameters as its original version and try to do innovative work on the video style transfer part.

3.1.2 Video Style Transfer.

Since the CNN-based NST model takes 1 style image and 1 content image as inputs, when we transfer the artistic style to a video, there will be serious difference between consecutive video frames if we directly apply style transfer on the individual frames independently.

Thus, in order to prevent huge style difference and sudden changes across the video, we need to further add pairwise consistency constraints for consecutive frames of the video. Inspired by [3], we design a new loss function for the content representations by supplementing a squared-error consistency loss between 3 consecutive frames. Note that our innovation and project novelty over the CNN NST part is illustrated here in video style transfer.

In the original paper, the overall loss function (including content and style) is given by:

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{content}} + \beta \mathcal{L}_{\text{style}} \tag{1}$$

where in the source code, the tuned weight parameters are: $\alpha = 8 \times 10^{-4}$ and $\beta = 8 \times 10^{-1}$. For this loss, the original content loss is defined as:

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij}\right)^2 \tag{2}$$

where F^l_{ij} is the activation of the i^{th} filter at position j in layer l w.r.t. the generated feature representation and P^l_{ij} is the respective feature representation for the input content image.

From the values of α and β, we can see that the content loss has a significantly lower weight, and our consistency loss could only be added to the content loss, so the very first step is to enlarge the content loss weight. After experimenting with difference possible values, we set $\alpha_{video} = 8 \times 10^{-3}$, which is 10 times larger than the original value, and the style loss weight remains unchanged: $\beta_{video} = \beta = 8 \times 10^{-1}$. The trade-off here is that: after we increase the content loss, meaning the style loss will be less dominant, the final output frames may contain more 'style noise' than the original well-tuned weights. But this is the only solution we could figure out in this project. (Probably, in the future work, we could try with different methods by implementing the full pipeline in [3] for better performance.)

Subsequently, when generating each frame of the video, we incorporate the 2 previous consecutive frames, denoted as Q^l and R^l, respectively. The new content loss with frame-to-frame consistency is formulated as:

$$\mathcal{L}_{\text{content}}^{\text{consistent}} = \frac{1}{100} \times \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - P_{ij}^l\right)^2 + \frac{49}{100} \times \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - Q_{ij}^l\right)^2 + \frac{50}{100} \times \frac{1}{2} \sum_{i,j} \left(F_{ij}^l - R_{ij}^l\right)^2. \quad (3)$$

Here the reason we significantly increase the weight for the loss between current feature and the 2 previous frames lies in that: the contents & features in the consecutive frames of a video should be rather similar, and our main goal is to constrain the difference between this generated frame to the previous style-generated frame, so increasing the weights on loss over previous frames will not only NOT affect the style generation quality (owing to the high feature similarity between frames) but also impose larger constraints over the frame-to-frame difference.

Using this new consistency-based content loss, the difference between every 3 consecutive frames from the video will be tremendously minimized.

In addition to the consistency loss we have implemented, we also employ another trick on image initialization, as the second novelty here in CNN-based NST. For each frame that is not the first frame, we initialize the generated (style-transferred) image not by using a random matrix, but by adopting the generated image w.r.t. last frame. In this manner, we could implicitly improve the consistency between consecutive frames and we also believe that using previous-frame generated image for initialization could help accelerate the convergence during the training process. This strategy will further boost the performance and smoothness of the final output video.

3.2 CycleGAN

3.2.1 Datasets.

Our baseline dataset consists of over 7000 real photos and 400 paintings by Van Gogh. They have been resized to 256×256 and validated to be in RGB format.

3.2.2 Architectures.

1. **ResNet Generators.** The two generators G_{AB}, G_{BA} share the same ResNet based architecture. It includes an encoder with 3 convolutional layers, within which the last 2 are down-sampling, 9 skip-connection residual blocks, and a decoder with 2 up-sampling transpose convolutional layers and a final output convolutional layer. All hidden layers use ReLU as the activation function, while the output layer uses Tanh. For each convolutional layer and residual block, we add a instance normalisation layer before ReLU to remove instance-specific contrast information from the content image and simplifies generation.

2. **PatchGAN discriminators.** The discriminator D_A, D_B use the same 70×70 PatchGAN architecture. It divides the input image into overlapping 70×70 patches and classifies on each patch. Its output is a $1 \times 30 \times 30$ tensor with each entry representing its prediction on the corresponding patch. The network consists of multiple convolutional layers with Instance Normalization (except for the first layer) and Leaky ReLU activation, and a output convolutional layer that has no normalization or activation.

3.2.3 Loss functions.

Our objective functions are defined as:

$$Loss = loss_{GAN} + \lambda * loss_{cycle} + \mu * loss_{identity}$$

$$loss_{GAN} = (E_{x \in A}[(D_B(G_{AB}(x)) - x)^2] + E_{y \in B}[(D_A(G_{BA}(y)) - y)^2])/2$$
$$loss_{cycle} = (E_{x \in A}[||G_{BA}(G_{AB}(x)) - x||] + E_{y \in B}[||G_{AB}(G_{BA}(y)) - y||])/2$$
$$loss_{identity} = (E_{x \in A}[||G_{BA}(x) - x||] + E_{y \in B}[||G_{AB}(y) - y||])/2$$

We use MSE (L2) loss to measure adversarial loss and MAE (L1) loss to measure cycle loss and identity loss. Note that the identity loss is introduced on top of cycle loss and GAN loss. This will facilitate the preservation of overall features such as the image's tonality. λ and μ are hyper parameters that control the weights of identity loss and cycle loss. We choose $\lambda = 10$, $\mu = 5$.

3.2.4 Optimization details.

We use a learning rate = 0.0002 as our starting point and use the step scheduler with 10 total updates and a factor of 0.5 at each step, i.e. if we train for 100 epochs, we would have 0.0002 for epoch 1 to 10, and 0.0001 for epoch 11 to 20... Therefore we can achieve a smoother training process. We use the Adam optimizer for all our networks.

3.2.5 CycleGAN for video

Due to the fact that CycleGAN is trained on large pool of images and has multiple losses to force it to preserve the content and color profile, we generate videos frame by frame using trained model without enforcing extra constraints on frames (unlike with NST).

4 Experiments & Results

4.1 Neural Style Transfer

The image style transfer results are explicitly shown in Fig. 2 and 3 below. We adopt 4 famous paintings as 4 different style images, and transfer the artistic style to 6 different content images (including 1 sample image and 5 images shot by Lei Sun).

We can see that the style transfer results over different style and content data are overall successful.

However, tradition NST has a drawback. Specifically, when the context (e.g. objects, scenarios of the photo) of the content image does not match with that of the style image, the performance would be, in a sense, compromised. For example, since the *star night* style image has rather similar context as the first 5 input content images (all having a horizon differentiating the sky from the view or buildings), the style transfer results are quite promising; nonetheless, when it comes to the sixth image (waterfall), since there is no sky in this image, the water would be endowed with the style of the sky. Thus, the limitation of NST is that we have to ensure some extent of context similarity between the content and style images.

The video style transfer results are demonstrated in the project video and also in the result folder of our submission.

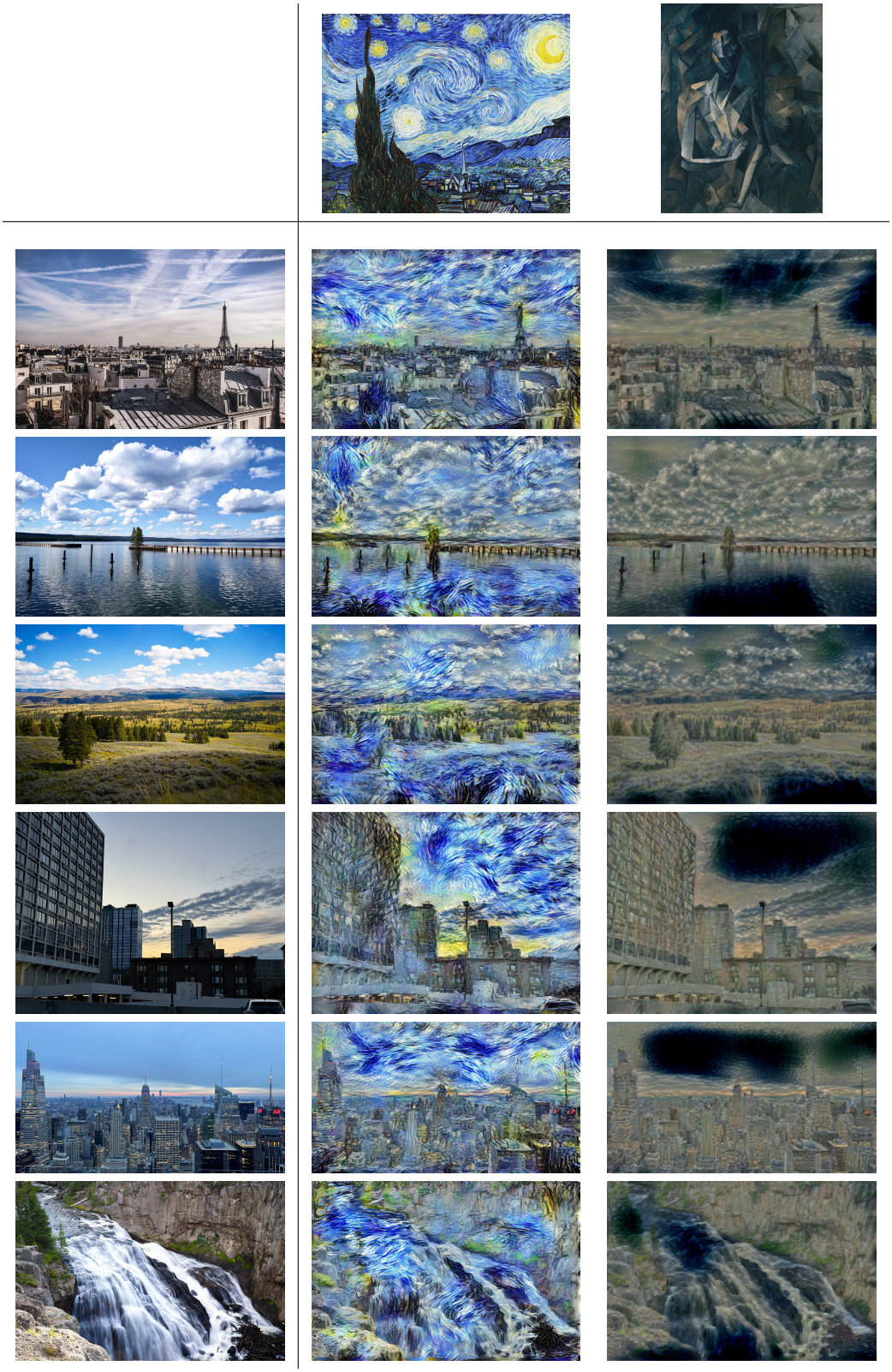

Figure 2: Artistic style transfer results using NST. The top row represents style images (*star night* and *seated nude*), and the left column shows the input content images. Then, the corresponding style-transferred output images are displayed.

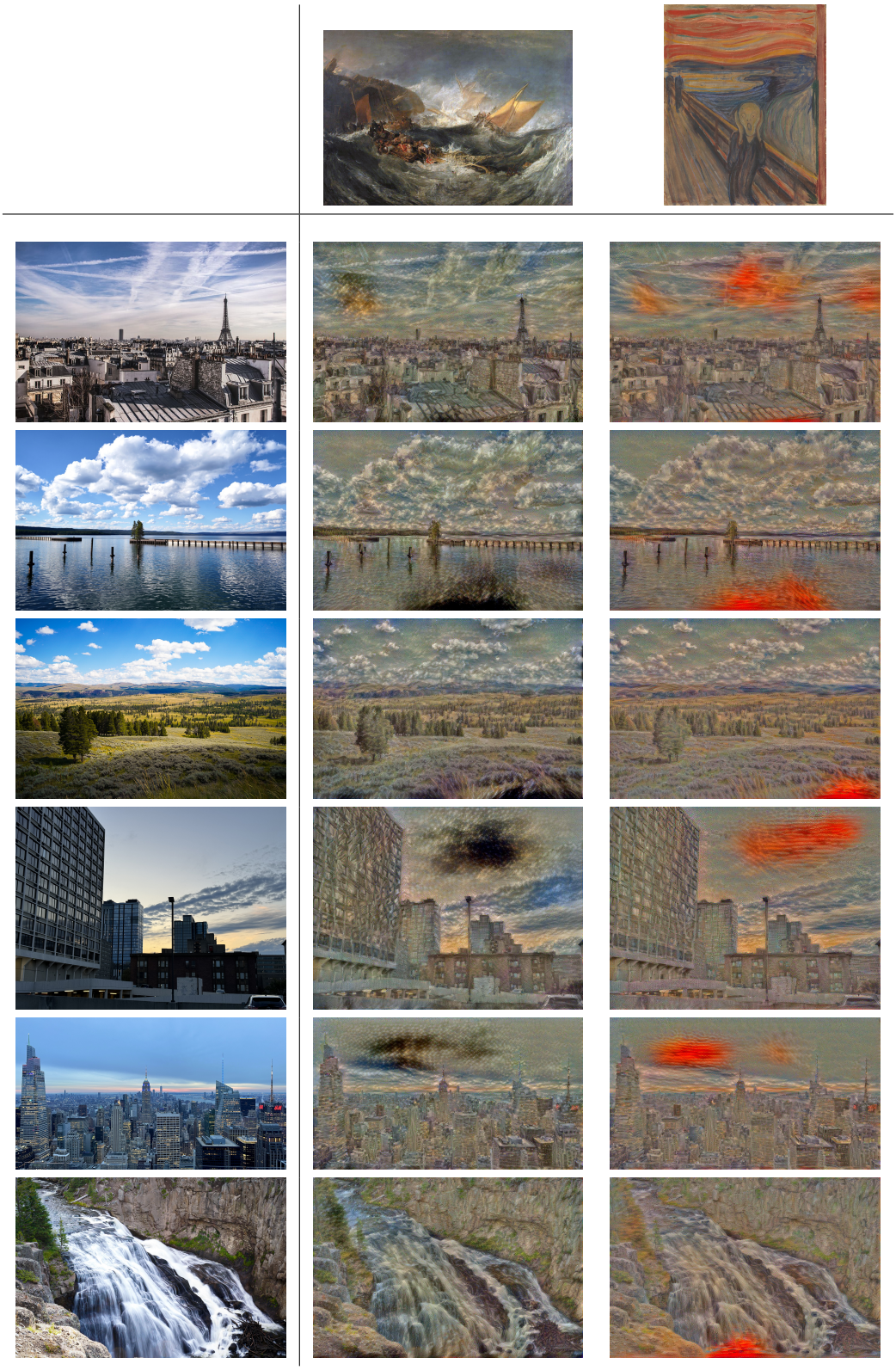

Figure 3: Artistic style transfer results using NST. The top row represents style images (*el paso* and *scream*), and the left column shows the input content images. Then, the corresponding style-transferred output images are displayed.

4.2 CycleGAN

4.2.1 Baseline

With our baseline implementation and dataset, we are able to generate impressive output images (see figure 4). We notice that CycleGAN is very good at transferring high-frequency features: the textures of water, rocks, and grass in real photos are nicely transferred into Van Gogh's course-grain, wavy style, while the same kind marks of style in Van Gogh's paintings are removed in generated photos.

While CycleGAN is good at transferring textures, it is not able to transfer any particular large object to be of new shapes. In other words, the in generated photos, the trees still look like Van Gogh's trees and not like real trees. The architecture does not have the capability of understanding what a real tree look like. Such feature may require more powerful architectures like transformers.

Figure 4: Row 1: Van Gogh' paintings, Row 2: generated photos, Row 3: real photos, Row 4: generated Van Gogh's paintings

4.2.2 Deeper Networks

In an effort to improve the visual quality of the generated results, we add more residual blocks to the original architecture. We hypothesize that the deeper the network becomes, the better the representations of style and content are learned and transformed in the embedding space, allowing improved details in the outputs.

We train CycleGAN models with generators that have 9, 15, 30 residual blocks using our baseline parameters and dataset. We notice that the deeper networks produce better color dynamics (more dynamic range) and less noise. The output images appear to be sharper and cleaner. But overall, we see limited difference in the change of shapes or textures (see figure 5).

Figure 5: left to right: original photo, outputs of 9-block, 15-block, 30-block ResNet generators.

4.2.3 Modified Up-sampling Layers

During early stage of training of out baseline model, we notice the checkerboard artifact (fig.6). We hypothesize that it is resulted by using transpose convolutional layers. The $nn.ConvTranspose2d$ we use for up-sampling works by filling each channel with implicit zeros (when it up-samples a 1×1 grid cell to 2×2, it first sets the other 3 cells to 0, then uses convolutional filters to transform the values), resulting in inconsistent color distributions inside each block. As we can see, the sky and water in figure 6 look like checkerboard.

To resolve this problem, we replace the transpose convolutional layers to a direct up-sampling layer (using interpolation) followed by a convolutional layer. The difference is that, by up-sampling using interpolation, the up-sampled channels retain the smoothness from last layer. As we can see in figure 7, the generated images are indeed smoother and not gird-like.

Figure 6: Checkerboard artifact when using transpose convolution for up-sampling

Figure 7: No checkerboard artifact if using interpolation up-sampling followed by convolutional layer, but can see "holes"

However, this introduces another artifact. It is noticeable that there are colored "holes" in figure 7. This is only noticed when we use interpolation up+sampling followed by convolution. We hope that if we train the model for more epochs, the "hole" artifact can go away. However, figure 8 shows the result after 100 epochs: the "holes" stay.

Figure 8: Holes still present after 100 epochs of training

Some research indicates that this kind of artifact is due to the "mode collapse" of GAN, which usually happens when the model is over-trained on small data. But why it does not appear when we use transpose convolution? We hypothesize that it is due to the fact that transpose convolutional layers set implicit zeros to fill the channels and force the filters to adapt and learn something new to produce better output, whereas the interpolation sets the whole optimization at a bad local minimum and stops the generator from escaping it.

4.2.4 Feature Extraction Backbone

In NST, we use pre-trained VGG-19 to extract features from input images, and the extracted features are used for further image synthesis. For CycleGAN, we experiment with the same idea: instead of training the encoder together with the whole generator network, why don't we use a feature extraction backbone as the encoder?

Conceptually, the encoder (the first three layers of the generator)'s job is to extract features from the input image and encode them into smaller representation space for later transformation (done in the residual blocks). However, the 3-layer encoder appear to be weak. We take out the encoder layers from the generators and use a pre-trained ResNet101 network to take the input image and pass the output features directly to the residual blocks. We use the last layer of the 4th block from the pre-trained ResNet101 as backbone and do not train it with the model.

Unfortunately, we do not have good results from this experiment, and we think the reasons are the following. First, the backbone gives 16×16 outputs, a factor of 4 times smaller than the original size, so we need 2 more up-sampling layers to decode the features when they come out from the residual blocks. However, decoding feature representations to a whole image is not easy. With the 16×16 features, too much information has been compressed compared to 64×64, resulting the training to take forever. Secondly, because we do not train the backbone with the model, the backbone is not optimized to extract more meaningful features for transferring. The backbone has been trained for classification objectives, but it does not necessarily have the capability to extract information about styles using the pre-trained parameters.

4.2.5 Pre-trained Encoding Layers

Similar to the backbone experiment, we are inspired by the idea of NST of using a pre-trained CNN architecture to improve our network. This time, we choose to replace the encoder with the first 3 blocks of a pre-trained VGG-19 (up to the 8th convolutional layers). We train our pretrained VGG encoder together with the model and obtain impressive results (see figure 9).

We notice that the images generated by this network are significantly sharper compared to the baseline. We think it is due to the fact that the pretrained VGG encoder is trained to extract abundant information about edges. It is a plausible hypothesis, because the layers we take from the pre-trained VGG are the shallower layers, which, as taught in the class, are known for having encoded information about edges.

However, we also notice some artifact when using the pre-trained encoder. Figure 9 shows that there are colored noise blocks appearing in the generated images. We think it is also due to some kind of overfitting: the discriminators have memorized some kind of color patterns from the small painting datasets, and so the

generators receive too specific feedback on generating their results, failing to generalize to produce smooth colors. However, deep GAN is hard to explain, so our hypothesis can only remain hypothesis.

Figure 9: Result using model with pre-trained VGG encoder

4.2.6 Different Datasets

As mentioned previously, most of the artifacts (holes, noise) can be attributed to overfitting, which in turn, can be caused by having small dataset. In an effort to have more pleasingly looking output images, we collect paintings from all famous impressionism artists to form a large painting dataset. The dataset has over 3,000 impressionism paintings, which are resized to 256×256 and validated to be in RGB format. With the new dataset, we now train our CycleGAN model to transfer between real photos and impressionism paintings (not just Van Gogh anymore) and have the following results (figure 10):

Figure 10: Transfer between real photos and impressionism paintings

We notice significant improvement in terms of texture smoothness and the amount of noise. The new generated paintings are more natural-looking and cleaner. However, this may also be attributed to the fact that the model is trained on impressionism art overall: Van Gogh is known for having strong style and more abstracted shapes, and the other impressionism artist such as Monet, have styles that are more natural and realistic. We can see that the image on the bottom left in figure 10 appears to have Monet peaceful style.

We know that larger datasets are better, but we can further improve the results by limiting datasets to contain specific things. Imagine we train our model on datasets that are mix of landscapes, faces, and plants, it is very difficult for the model to understand the substantial difference among the classes of content and be able to generalize its outputs. We therefore collect another pair of datasets that only contain landscape photos and impressionism paintings and obtain even more impressive results (see figure 11). We can see that not only the photos are generated into convincing paintings, the paintings are also generated into very realistic photos. By narrowing down the task to transferring styles for a specific type of content, the model faces less trade off and can be optimized to produce superb outputs that can even fool human eyes (and mind).

Figure 11: Transfer between real landscape photos and impressionism paintings of landscapes

4.2.7 Videos

The generated videos can be found at the submission folder. Overall, the frames are consistent (without abnormal changes in between frames) for all videos.

5 Qualitative Analysis

We have seen substantial differences between how NST and CycleGAN generate results and how their results differ visually. NST uses a specific style image and blends it with a content image, resulting the style to be more obvious in the generated output. However, it is also limited by the type of style images it take as inputs. We have seen that it does exceptionally well if the style image has evenly distributed style across the

whole frame (like the Starry Night). But if the style or color distribution is not consist in the style image, that unevenness can be blended into the content image to become artifact (big blocks of color). CycleGAN, on the other hand, is trained to have generalized style. It learns the generalized concept of a style (like that of Van Gogh's), not the style that appears in any specific images. However, we also see that the style that CycleGAN can learn and transfer, at the maximum capacity of the architecture, is mostly the texture and color profile.

Also, CycleGAN has more potential use cases. Because it can transfer the generalized style, it can perform tasks such as transferring between modalities of medical images. For example, it can transfer a MRI brain scan to a CT brain scan. As we have discussed, CycleGAN models improve significantly when they are trained on datasets that contain narrower range of objects. If we train a CycleGAN model in brain scan specifically, it can produce very accurate results.

6 Conclusion

In this project, a traditional NST model and a CycleGAN model are both applied to realize image and video artistic style transfer. For NST, frame-to-frame consistency constraints are imposed on the training process to extend image style transfer to style-based video synthesizing, on the basis of its original baseline. For CycleGAN, we employ various strategies including adopting deeper networks, modifying up-sampling layers, using feature extraction backbone and pre-training encoding layers to improve the model performance. Different image datasets are adopted for the training process. Both NST and CycleGAN could generate impressive style transfer results in our experiments.

References

[1] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. *Advances in neural information processing systems*, 28, 2015.

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[3] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *International Journal of Computer Vision*, 126(11):1199–1219, 2018.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.