

Neural Style Transfer

Yuejiao Qiu, Ruoting Shen, Wenxuan Tang, Yuening Wang, Sirui Wang

Georgetown University

ANLY-590 Neural Nets and Deep Learning

Dr. James Hickman

December 9, 2022

Introduction

Around 2012, convolutional neural networks (CNN) enjoyed a huge surge in popularity (which continues today) after a CNN called AlexNet achieved state-of-the-art performance labeling pictures in the ImageNet challenge. Convolutional neural networks can capture abstract aspects of images, making it possible for computers to perform challenging visual jobs. Neural style transfer (NST) generally targets creating textures based on source images; meanwhile, it also restricts the creation of textures to protect the target image's content. Because it is difficult to explain and satisfy theoretically, this task is particularly challenging compared to standard machine learning issues. Image-optimization-based (IOB) online neural methods and model-optimization-based (MOB) offline neural methods are the two primary methods to handle this NST problem.

This paper has two phases for the implementation of the model. The first generative model uses the IOB-NST algorithm Gatys et al. (2016) proposed to implement fixed style to fixed content style transfer. The second generative model, to maintain better visual quality and operational efficiency, follows Johnson et al. (2016) and Ulyanov et al. (2016) by using an image transform network and instance normalization to achieve real-time style transfer.

Dataset

Because the first model uses the basic style transfer of the IOB-NST algorithm and uses a pre-trained VGG-19 as the loss network, there is no need for a training dataset, only some style images, and photos that need to be transformed. Here, well-known paintings are used as style images, and these images can be found in the Kaggle link (<https://www.kaggle.com/momincks/paintings-for-artistic-style-transfer>).

The second model is a generative model for real-time style transfer. Training this model requires using a large enough dataset of content images so that it learns what needs to be retained. For this purpose, the selected dataset is LabelMe, which is a large dataset containing 187,240 images created by the MIT Computer Science and Artificial Intelligence Laboratory. Based on memory limitations, 20,000 of them were used in the experiment.

Methods and Algorithms

In this section, models and algorithms that are implemented in the project will be introduced. The basic style transfer model is based on Image-Optimization-Based (IOB) Online Neural Methods, and the fast style transfer model is based on Model-Optimization-Based (MOB) Offline Neural Methods with an image transformation network to improve. Both of them utilize a pre-trained VGG-19 as the loss network.

Basic style transfer

IOB-NST algorithms extract style and content from source images to produce a target representation and then output the stylized results that match target representations through iterations (Jing et al., 2019). However, this model is also limited by its iterative optimization process, which leads to low efficiency.

A pre-trained VGG-19 (Figure 1) is utilized as the loss network, implementing an irreversible network consisting of 16 layers of convolution and 5 layers of pooling. The selection of styles and content layers is an important part of style transformation. According to the study by Gatys et al. (2016), conv1_1, conv2_1, conv3_1, conv4_1, and conv5_1 are used as style layers and conv4_2 as the content layer. The loss can be calculated with the features in layers. The goal is to transfer the style of the source image onto the target image. The final new image matches both the content and style. And LBFG_S is utilized as an optimizer to iteratively adjust the gradient and minimize the total loss.

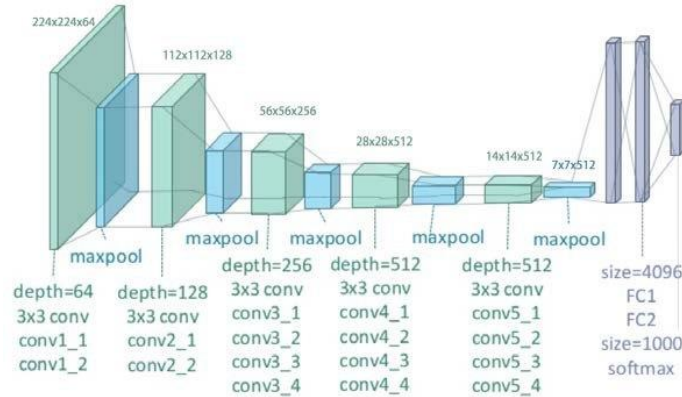


Figure 1. VGG-19 System (Zheng et al., 2018)

Fast style transfer

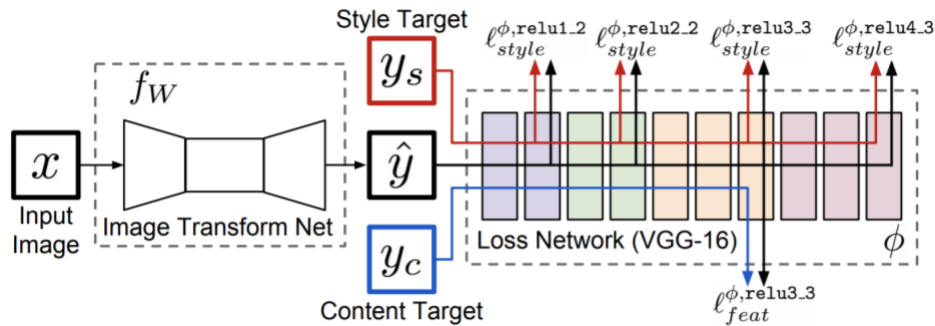


Figure 2. System Overview (Johnson et al., 2016)

Although the basic style transfer shows quite exceptional results, it still has many limitations, and the most significant one is the efficiency problem. That is, if a new style or content of the image is supposed to be used, the basic style transfer model requires inputting a new image and iterating every time. Therefore, in order to improve the speed and reduce the computational cost, the MOB-NST algorithms are implemented to create a new model which can conduct a real-time style transfer.

The MOB-NST algorithm can be achieved through three methods, which are Per-Style-Per-Model (PSPM) MOB-NST methods, Multiple-Style-Per-Model (MSPM) MOB-NST Methods, and Arbitrary-Style-Per-Model (ASPM) MOB-NST Methods. The first two MOB-NST algorithms are raised by Ulyanov et al. and Johnson et al., both of them can pre-train a feed-forward network and generate stylized results with a single forward pass during the test phase (Jing et al., 2019). The only difference is that Johnson et al. use residual blocks and fractionally strided convolutions, while Ulyanov et al. use a multi-scale generator network (Jing et al., 2019).

The network architecture used in this project is proposed by Johnson et al., which is more simplified. As Figure 2 shows, the images are input and trained by the image transformation network and then transformed into output images. Following is the first part of this fast style transfer model.

Layer	Activation size
Input	$3 \times 256 \times 256$
Reflection Padding (40×40)	$3 \times 336 \times 336$
$32 \times 9 \times 9$ conv, stride 1	$32 \times 336 \times 336$
$64 \times 3 \times 3$ conv, stride 2	$64 \times 168 \times 168$
$128 \times 3 \times 3$ conv, stride 2	$128 \times 84 \times 84$
Residual block, 128 filters	$128 \times 80 \times 80$
Residual block, 128 filters	$128 \times 76 \times 76$
Residual block, 128 filters	$128 \times 72 \times 72$
Residual block, 128 filters	$128 \times 68 \times 68$
Residual block, 128 filters	$128 \times 64 \times 64$
$64 \times 3 \times 3$ conv, stride 1/2	$64 \times 128 \times 128$
$32 \times 3 \times 3$ conv, stride 1/2	$32 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$

Table 1. Fast style network architecture (Johnson et al., 2016)

The structure of the fast style transform network is shown as Table 1, this structure can be divided into three parts for different processes. The input images with three channels are accepted by the network. The first part of the network, which includes three convolutional layers, aims to transform the original images to increase the channel number to 128, two layers with a stride of 2 are applied for downsampling. The second part of the network is several residual blocks, for each residual block, there are two convolutional layers without padding. And for better performance on classifying images, the ReLU activation is removed from the last layer for all residual blocks. The reason for adding residual blocks to the network is that a residual network has a faster running time compared to a non-residual network while they eventually achieved similar training loss (Johnson et al., 2016). The last part of the network includes two convolutional transpose layers

with a stride of 1/2, which contribute to the upsampling process. With the stride of fraction, the convolutional layer can accomplish the upsampling process with the rest of the network.

The last layer is a convolutional layer with scaled-Tanh as activation instead of ReLU, it should be noticed that ReLU and batch normalization is applied to all layers except convolutional layers as residual blocks and the final layer of the network. Applying tanh on the final layer can guarantee the network output remains in RGB format (Johnson et al., 2016). For the normalization, instance normalization is applied instead of batch normalization for a better performance on shorter runtime and better image results.

Furthermore, as Figure 2 shows, VGG-19 is used as a pre-trained loss network instead of VGG-16 to determine perceptual loss functions, which calculate the perceptual differences between style and content for images (Johnson et al., 2016).

$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2 \quad (1)$$

$$\ell_{style}^{\phi,j}(\hat{y}, y) = \|G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)\|_F^2. \quad (2)$$

$$\hat{y} = \arg \min_y \lambda_c \ell_{feat}^{\phi,j}(y, y_c) + \lambda_s \ell_{style}^{\phi,J}(y, y_s) + \lambda_{TV} \ell_{TV}(y) \quad (3)$$

Equation 1 computes the content loss via Euclidean distance, equation 2 computes the style loss via “the squared Frobenius norm of the difference between the Gram matrices of the output and target images,” and equation 3 represents the process that minimizes the total loss (Johnson et al., 2016). The calculation of perceptual loss for fast style transfer is similar to that of basic style transfer. The only difference is that the basic style transfer shows optimizations over outputs of provided pairs of images, while fast style transfer trains the image transformation network and is able to deal with any content or style of images.

Experiments

Basic style transfer

In experiments, Python Pytorch 1.13.0 is used to implement basic style transfer. The hyperparameters in this paper's model are consistent with those of Gatys et al. (2016), that is, loss weight $\alpha = 1$ and style loss weight $\beta = 1 \times 10^6$. An overview of outputs is provided below.



Figure 3. Outputs from basic style transfer

(Photo of Luna from Dr. James Hickman)

As Figure 3 shows, basic style transfer successfully transfers the textures of image style with the preservation of the content, and style and content are roughly mixed. Unfortunately, one major limitation is that the model requires a new input and iteration in order to use a new style or content of the image and iteratively apply the gradient descent to minimize the loss during the process. In the experiment with GPU, with the number of iterations set to 800, the average time to output the image is 63.27 seconds, which is too slow for business purposes. Therefore, fast style transfer is built to achieve the real-time style transfer, which will be conducted and discussed in the following.

Fast style transfer

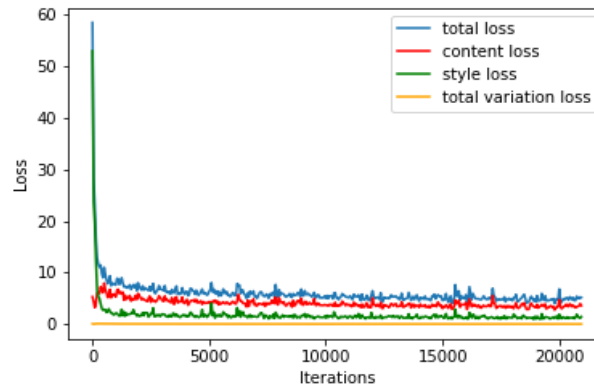


Figure 4. Loss trend for fast style transfer (Vincent Van Gogh)

Seven separated fast style transfer models based on paintings from different famous artists are built and the loss trend descent plots during the training process are attached in the appendix. Figure 4 shows the loss plot of the fast style transfer model based on Vincent Van Gogh as an example, since the loss plot patterns only show slightly different. The overall trends are identical in general and several interesting facts need to be noted. First, the content loss variation is not significant, which is due to the initialization settings made during the training process. Regarding style loss, it rapidly decreases after approximately 1000 iterations, then the downward trend

becomes comparatively stabilized. Numerous methods were tried in this procedure to find an ideal point to stop training, but the outcome showed that the more the number of iterations, the greater the integration of style and content. Hence, despite the slow decreasing trend of the loss, the iterations still increased to 20,000.

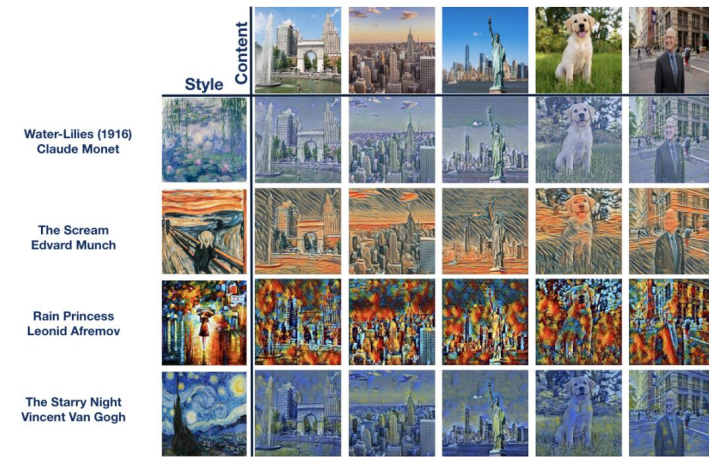


Figure 5. Outputs from fast style transfer

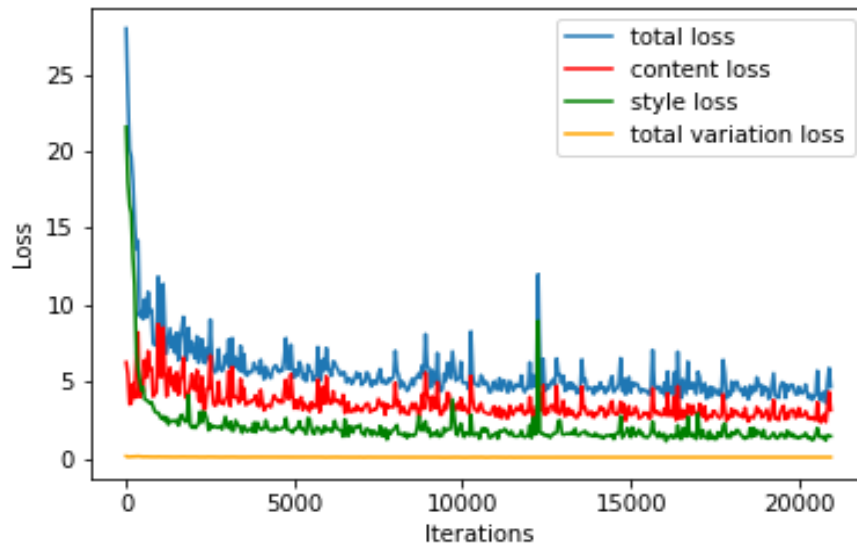
Figure 5 is a demonstration of the results for all models. The average time to transform a random image is approximately 0.17 seconds once the painting-specific style transfer model is trained, which partially implements real-time style transfer, especially when compared to the one-minute run time of the basic transfer. For a stronger test of the model performance, images with numbers were examined and better results were obtained. The model not only transferred the color but also the texture of the style painting. Simultaneously, the objects in the content images are well preserved, sharp object shapes can still be noticed in the output images, regardless of the relatively messy stylized paintings.

Not surprisingly for both transfer methods, the basic transfer produces superior outcomes, especially for the portrait of the Mona Lisa (Appendix 8). It is reasonable that the basic transfer gives better performance because it is trained upon a specific genre of painting and a content-specific image. For images with stochastic content, however, fast style transfer is extremely time saving and flexible.

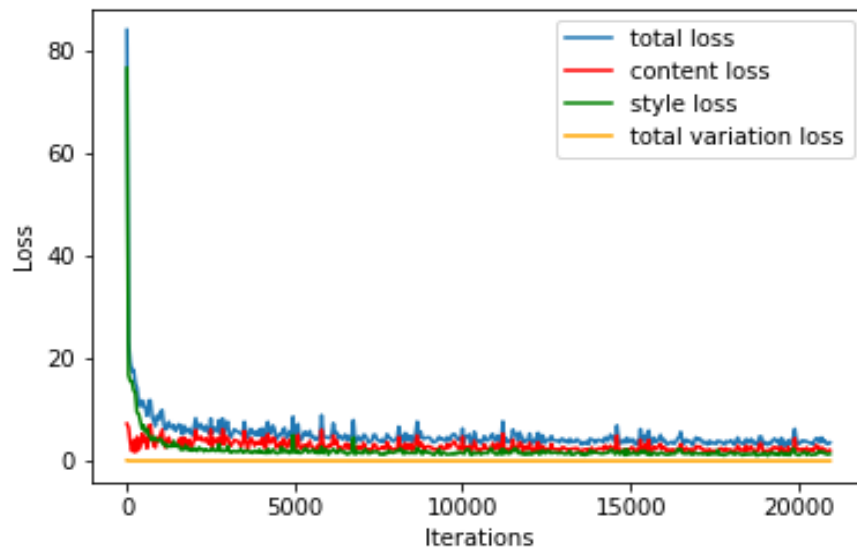
Conclusion

In this project, basic style transfer and fast style transfer models are performed to extract different artistic styles and apply them as filters to the target images. Due to the limitations of the basic style transfer model, a fast style transfer model with an image transformation network is constructed to improve efficiency. The style can be utilized most while preserving the image content by minimizing the total loss. Besides, steps such as using basic style transfer output to fine-tune a fast style transfer model, achieving real-time style transfer regardless of the style, and blending multiple styles into one content image, might be considered as further explorations.

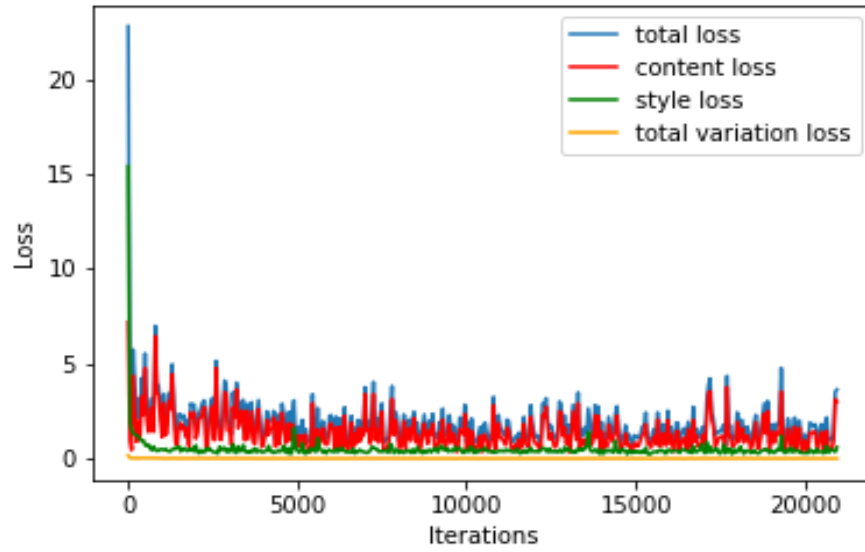
Appendix



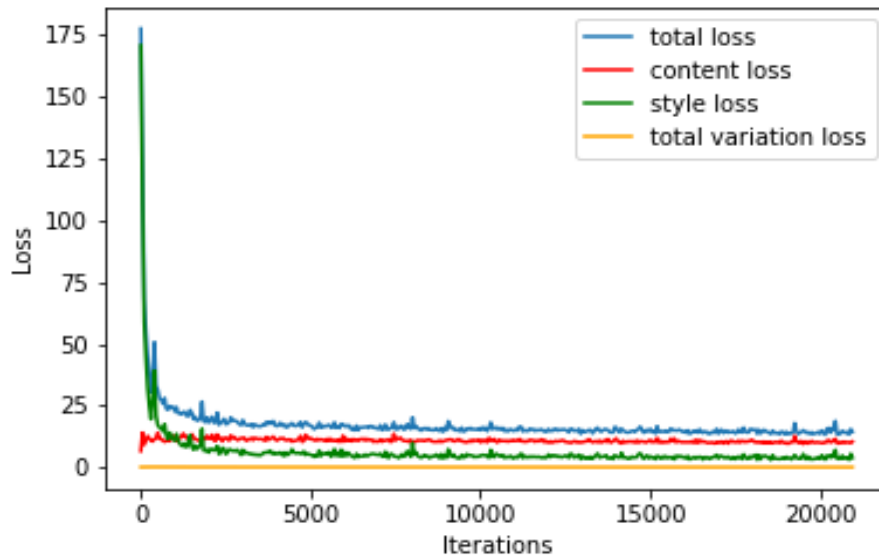
Appendix 1. The Great Wave off Kanagawa (Katsushika Hokusai)



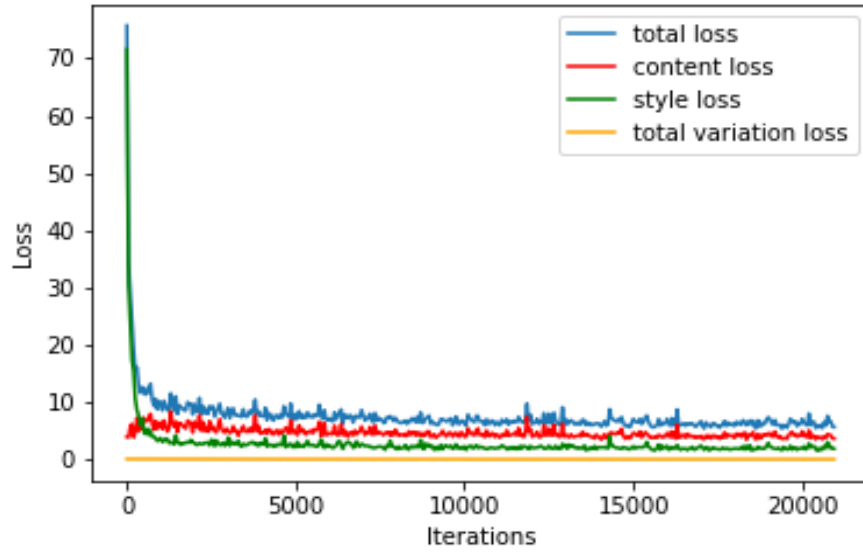
Appendix 2. Old Canal Port (Oscar Florianus Bluemner)



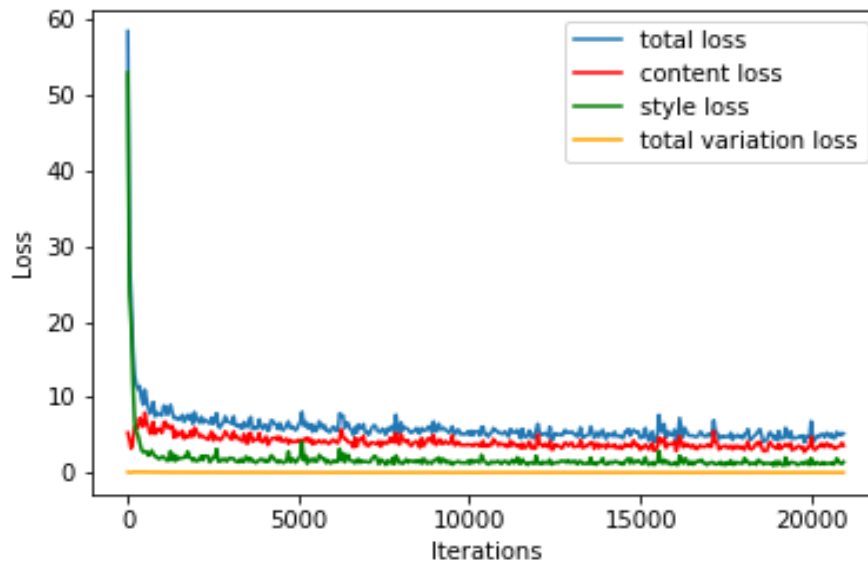
Appendix 3. Portrait of Lisa Gherardini (Leonardo Da Vinci)



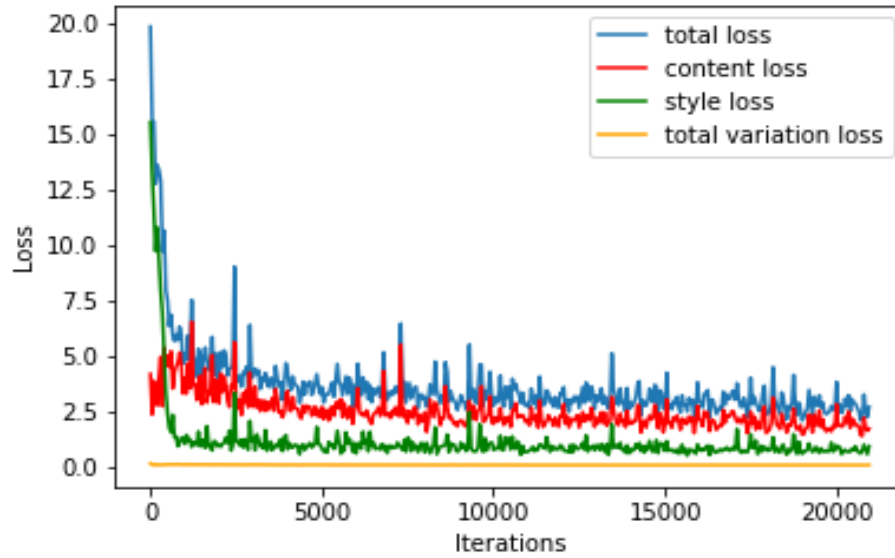
Appendix 4. Rain Princess (Leonid Afremov)



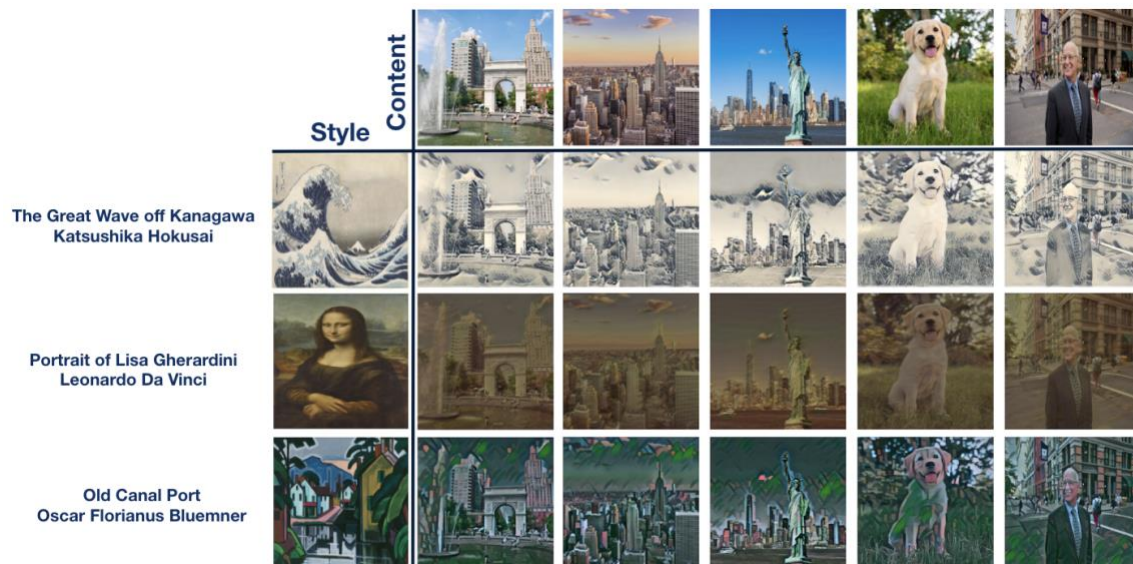
Appendix 5. The Scream (Edvard Munch)






























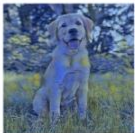

Appendix 6. The Starry Night (Vincent Van Gogh)



Appendix 7. Water-Lilies (1916) (Claude Monet)



Appendix 8. Output of Basic Style Transfer

	Style	Content					
Water-Lilies (1916) Claude Monet							
The Scream Edvard Munch							
Rain Princess Leonid Afremov							
The Starry Night Vincent Van Gogh							

Appendix 9. Output of Fast Style Transfer

References

- Gatys, L. A., Ecker, A. S., & Bethge, M. (2016), "Image Style Transfer Using Convolutional Neural Networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414-2423, doi: 10.1109/CVPR.2016.265.
- Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., & Song, M. (2019, June 06), "Neural Style Transfer: A Review," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 11, pp. 3365-3385, doi: 10.1109/TVCG.2019.2921336.
- Johnson, J., Alahi, A., & Fei-Fei, L. (2016, March 27). Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *arXiv.org*. Retrieved December 6, 2022, from <https://arxiv.org/abs/1603.08155>
- Johnson, J., Alahi, A., & Fei-Fei, L. (2016). Perceptual Losses for Real-Time Style Transfer and Super-Resolution: Supplementary Material. *Computer Vision – ECCV 2016*. Retrieved from <https://cs.stanford.edu/people/jcjohns/papers/eccv16/JohnsonECCV16Supplementary.pdf>
- Ulyanov, D., Lebedev, V., Vedaldi, A., & Lempitsky, V. (2016, March 10). *Texture networks: Feed-forward synthesis of textures and stylized images*. *arXiv.org*. Retrieved December 8, 2022, from <https://arxiv.org/abs/1603.03417>
- Zheng, Y., Yang, C. K., & Merkulov, A. (2018, May). Breast cancer screening using convolutional neural network and follow-up digital mammography. 4. 10.1117/12.2304564.