ABSTRACT

We started our style transfer project from a github demo based on VGG19 architecture. After learning the structure of existing code, we built our artistic style transfer project based on a VGG16 pre-trained model (weighted on ImageNet). Inside the VGG16 model, we modified the layers that extract style and content of images, the loss function, added total variation loss modification properly, and successfully generated our style transfer program with a single content image with a single style image input. We evaluated our style transfer model based on timing and human opinion. We also implement a multi-style transfer feature, which can transfer multiple style images onto one content image and create beautiful output that contains multiple styles.

2. Introduction: Overview, Motivations, Significance, and Related works

Our project aims to implement state-of-art neural networks architecture to obtain artistic style transfer. Artistic style transfer is a neural network application that designs to synthesize an image that inherits the content of one image while preserving a similar style of the other one. We also implement a multi-style transfer feature, which transfers multiple styles from style images onto one content image. We will also compare the style transfer performance of different neural network frameworks.

Neural network style transfer is a popular machine learning application. It is a technique that takes two images, a content image, a style image, and blends them together to generate an output image that contains the content from the content input image while painting in the style of the style image. Style transfer demonstrated the significance of deep learning in the interdisciplinary research area, connecting computer science with art in the real world. This technique also provides an astonishing method for artists to develop artworks based on existing art styles. Our group finds the style transfer technique interesting and a comprehensive application of neural networks, which is related to the course topic.

Problem formation:

- For input content image *p* and style image *q*, we create a stylized image *s* with content from *p* and style extracted from *q*. The image *p* is fed through the VGG16 architecture and extracted from a certain convolution layer' output to represent the content of *p*. The image *q* is also fed through the same architecture, while extracting multiple convolution layers' output to represent the style of *q* (style layers are usually early to middle layers of VGG16). Gram matrix is used to encode the style of these layers' output activations. Total variation loss is used to smooth the noise in the stylized image. Gradient descent with mean-squared loss is used as iterative optimization to generate the stylized image with content from *p* and style from *q*. Multiple style transfer is generated in the same procedure with multiple style inputs *q1, q2, etc,* with slightly different loss function setup, in order to create a stylized image with content from *p* and styles from *q1, q2, etc*.

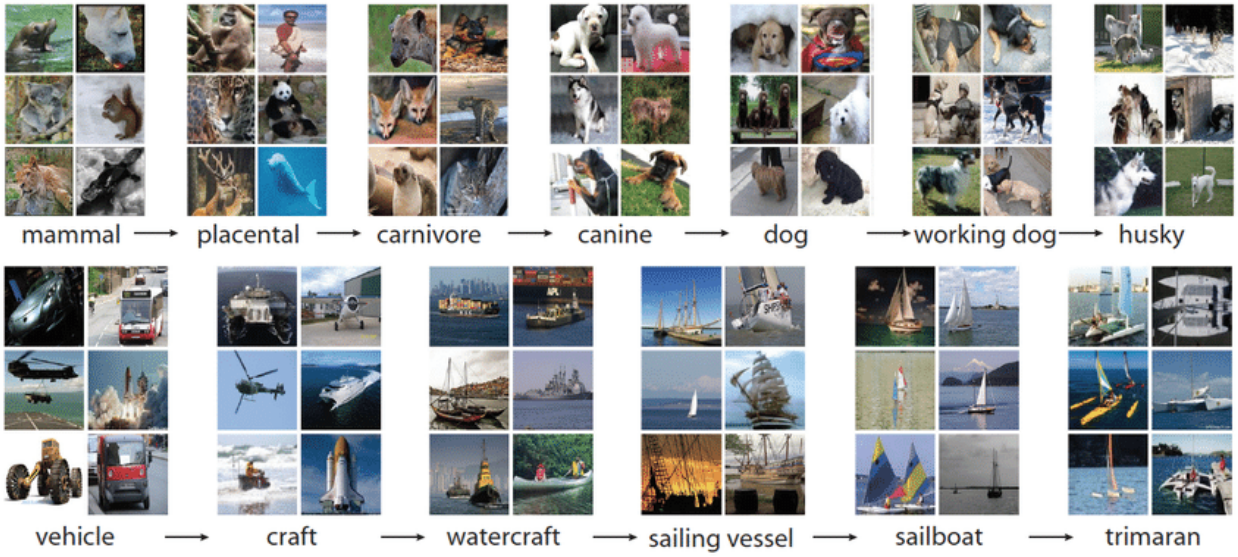
Style transfer is a relatively new topic in deep learning. To the best of our knowledge, *Gatys et al.(2015) "A Neural Algorithm of Artistic Style"* first proposed to use neural networks in style transfer. As for current state-of-art work, *Johnson et al.(2016) "Perceptual Losses for Real-Time Style Transfer and Super-Resolution"* use perceptual loss functions for training feed-forward networks for image transformation tasks; *Hu et al.(2020) "Aesthetic-Aware Image Style Transfer"* propose to transfer color and texture independently to control the aesthetic property of the output image; *Jing et al.(2020) "Dynamic Instance Normalization for Arbitrary Style Transfer"* propose a new and generalized normalization module enables an arbitrary style transfer using MobileNet-based lightweight architecture, leading to a reduction factor of more than twenty in computational cost as compared to existing approaches.

3. Method: Data, algorithm and program, platform, how to conduct experiments, what are the results, how to evaluate results (performance metrics)

3.1 Data

The dataset we use is ImageNet (https://www.image-net.org/about.php). It is because we found a pre-trained VGG16 model weighted on ImageNet, so that we can easily make use of it to generate compelling stylized images. ImageNet is one of the largest databases designed for deep learning. It contains more than 14 million labeled images from more than 20 thousands categories. The average image resolution on ImageNet is 469x387 pixels. Images are partitioned into 1,000,000+ training images, 50,000+ validation images, and 100,000 test images.

An example of ImageNet data:



While ImageNet is used to train the VGG16 model, we also collected our small content & style dataset to perform style transfer training. We collected 4 content images and 8 style images with distinct features to perform style transfer. We limited the image's maximum dimension to 512 pixels. Images in the **result section** are examples of our content & style dataset. This content & style dataset is in our github repository.

3.2 Tasks performed

We adapted from a github codebase using the VGG19 model and changed the structure into VGG16. After some experimentation, we selected “block5_conv2” as the content layer and “block1_conv1, block2_conv1, block3_conv1, block4_conv1, block5_conv1” as style layers. We followed the state-of-art technique, and used the gram matrix to calculate style features.

For style and content loss, we chose to use mean square error to calculate style loss and content loss. In 1 content 1 style image transfer, we weight style loss and content loss equally and use

their sum as the total loss. In multiple style transfers, we calculated style loss compared to each input style image respectively. We give each style loss equal weight, and calculate total style loss by summing up all style loss with respect to each style image. Then, we give total style loss and content loss equal weight, to prevent overfitting the style but losing the content information.

Hyperparameter. We use Adam optimizer with learning rate=0.03, beta_1=0.99, epsilon=1e-1. We set style weight to 1e-2 and content weight to 1e4, and total variation weight to 20.

To perform our result, we use our own testing dataset as image input, and generate various stylized output images. We repeated the experiment on multiple sets of image input and proved that our program pipeline has good generality. All our experiments are implemented on the Colab platform.

We evaluate the output images based on timing and human rate. We measured the time of producing the output image using our framework. Also, we ask people to rate our generated images (i.e, whether it contains the style and content from inputs, whether it is beautiful, etc). The higher the rate, the better the output is.

All code and images are saved to this Github repository:
https://github.com/xingpengsun/CS539-Group-Project

4. Results and Discussions

4.1 Results: results should be presented in tabular/graph format. Experiment conditions, parameters and related details should be provided in the text.

All experiments are conducted on the Colab platform. Using methods mentioned in the previous section, using VGG16 pretrained on ImageNet.

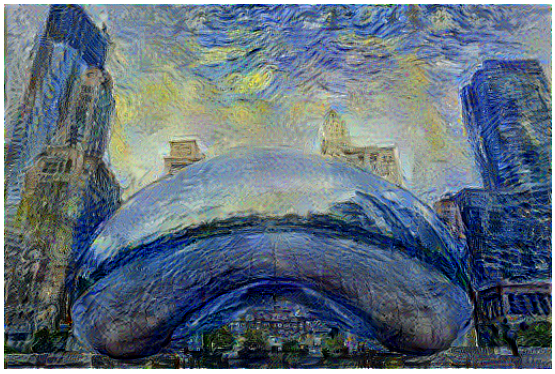
First Set of Style Transfer:

Content Image:

Style Image 1 & 2:

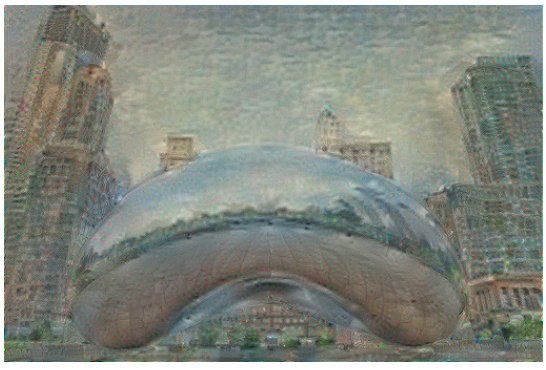
Starry Night - Van Gogh *Impression Sunrise - Claude Monet*

Stylized Image 1: Content Image + *Starry Night - Van Gogh*
Train Step: 500; Epochs: 5; Total Training Time: 1969.4 s



No total variation loss optimization *Add total variation loss optimization*

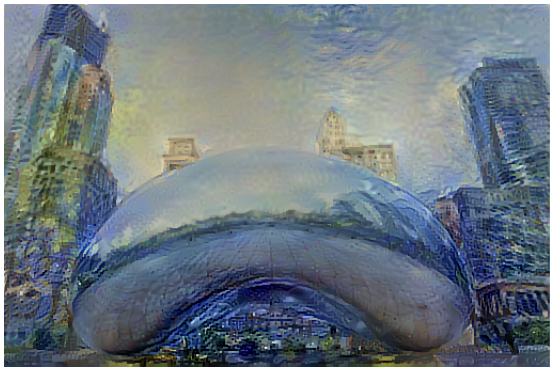
Stylized Image 2: Content Image + *Impression Sunrise - Claude Monet*
Train Step: 500; Epochs: 5; Total Training Time: 2084.3 s



No total variation loss optimization *Add total variation loss optimization*

Stylized Image 3: Content Image+*Starry Night-Van Gogh+Impression Sunrise-Claude Monet*
Train Step: 500; Epochs: 5; Total Training Time: 2249.4 s

No total variation loss optimization



Add total variation loss optimization

<u>Second Set of Style Transfer:</u>

Content Image:

Style Image 1 & 2:

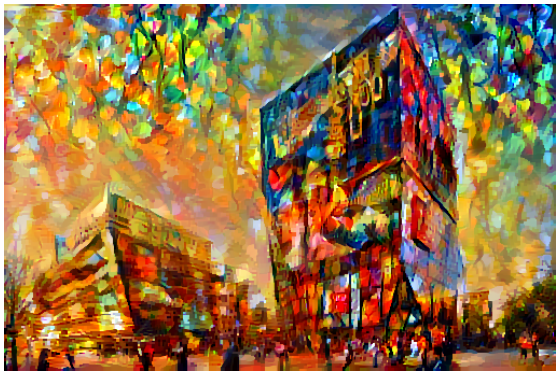
Starry Night - Van Gogh

Farewell to Anger - Leonid Afremov

Stylized Image 1: Content Image + *Starry Night - Van Gogh*
Train Step: 1000; Epochs: 10; Total Training Time: 4519.1 s

Add total variation loss optimization

Stylized Image 2: Content Image + *Farewell to Anger - Leonid Afremov*
Train Step: 1000; Epochs: 10; Total Training Time: 3984.8 s



Add total variation loss optimization

Stylized Image 3: Content Image + *Style Image 1* + *Style Image 2*
Train Step: 1000; Epochs: 10; Total Training Time: 4054.2 s

Add total variation loss optimization

Parameters:
Optimizer = Adam; Learning rate=0.03; beta_1=0.99; epsilon=1e-1;
Loss function = MSE loss

Evaluation:
The training time for single style transfer (32.8 min & 34.7 min for 500 steps) is less than multiple style transfer (37.5 min). From our perspective, we could see that the colors (starry night's dark blue, sunrise's light orange blue) of the style image are transferred to the content image successfully.

In stylized image 1, we could see the characteristics of the short, painterly brushstrokes, curly shape, and focus on luminescence, which are significant features from *starry night-van gogh.*

In stylized image 2, we could see the characteristics of impressionism, undulating reflections, and focus on vagueness, which are significant features from *Impression Sunrise - Claude Monet.*

In stylized image 3, we could see the medley of characteristics from both *starry night* and *impression sunrise.*

Similarly, the style feature and colors are transferred to the content images in the second set of images. Timing increases proportionally while increasing training epochs from 5 to 10.

Therefore, we conclude that we successfully transfer styles to the content image.

4.2 Discussions: Comparison of results reported in 4.1 against existing results on the same or similar data. Discuss causes of less than ideal performance, hint on how the result may be improved.

First, using a different dataset, ImageNet vs MSCOCO.

From stylized images we find that in the first set of images the cloud gate is clearer than background buildings, and the cloud gate learned the style better than background buildings. Also, the starry night style is not transferred to the whole sky of the second content image. We hypothesize that it is because ImageNet only learns the most significant object in the input image and classifies the image into one category. However, using MSCOCO can detect all objects inside the image, thus may apply the style to all objects better, performing a clearer image.

Second, using a different optimizer.

Currently we use Adam optimizer, but some research experiments suggest LBFGS optimizer will converge faster and produce better stylized images within a shorter time period.

Third, the impact of extracting different content layers.

We tried to set the content layer to "block5_conv3". However, this produced a really bad result. The style is only transferred to a small region inside the image, not the whole image. Therefore, we think the content layer should not be a very small vision domain. Thus, we selected "block5_conv2" and produced good results.

Fourth, total variation loss optimization.

Add total variation loss optimization could eliminate some artifacts and perturbation inside the stylized image. However, it may also downgrade the transferred style. Therefore,we will try to increase the training epoch and decrease the weight loss to balance the perturbation with feature style.

Fifth, different deep learning models.

Some state-of-art papers make use of more complicated deep learning models such as ResNet50. We hypothesize that complicated models may perform better results as they segment content and styles into smaller view domains and classify objects inside images more precisely.

To sum up, changing the training dataset into MSCOCO, using complex deep learning architectures, utilizing LBFGS optimizers, and other techniques may all improve the result.

5. References: you should use the standard reference format.

- Gatys et al. (2015). *A Neural Algorithm of Artistic Style.* https://arxiv.org/abs/1508.06576
- Hu et al. (2020). *Aesthetic-Aware Image Style Transfer.* https://dl.acm.org/doi/10.1145/3394171.3413853
- Jing, Y., Liu, X., Ding, Y., Wang, X., Ding, E., Song, M., & Wen, S. (2019, November 16). *Dynamic instance normalization for arbitrary style transfer*. arXiv.org. https://arxiv.org/abs/1911.06953
- Johnson et al. (2016). *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*. https://arxiv.org/abs/1603.08155
- Lin et al. (2015). *Microsoft COCO: Common Objects in Context*. https://arxiv.org/abs/1405.0312
- Mao-Chuang Yeh et al., (2020). *Improving Style Transfer with Calibrated Metrics*. https://arxiv.org/abs/1910.09447
- Pragati Baheti. (2021). *Neural Style Transfer: Everything You Need to Know [Guide]* https://www.v7labs.com/blog/neural-style-transfer
- TensorFlow. (2022). *TensorFlow Style Transfer Tutorial.* https://www.tensorflow.org/tutorials/generative/style_transfer