Real-Time Style Transfer Based on Lapstyle

Yuanpeng Gao **Da Yi** **Qiuxiang Zhao**

Abstract

This project explores two neural style transfer methods: Gatys-style and Lapstyle. While Gatys-style preserves content structure, it often introduces artefacts. Lapstyle improves coherence by incorporating a Laplacian loss function, enhancing detail preservation. A feedforward neural network is also trained for real-time rendering. Using datasets like MS-COCO and Wikiart, the project aims for high-quality, efficient style transfer. The project advances style transfer by balancing quality and efficiency for practical applications.

1 Introduction

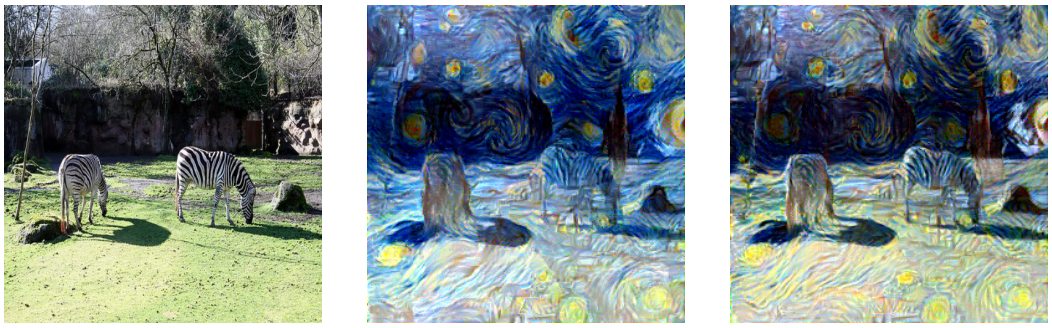
Style transfer is a widely studied technique in computer vision that combines the content of one image with the style of another. While traditional methods like Gatys-style have shown promising results in preserving content structure, they often suffer from artifacts and lack fine-grained coherence. Fig.1 presents an example of style transfer: given a content image 1(a) and a style image 1(b), a stylized image 1(c) is synthesized.

(a) content image (b) style image (c) Gatys-style

Figure 1: Given a content image (a) and a style image (b), style transfer creates a new one combining the content of (a) and the style of (b). (c): the stylized image by Gatys et al. [1] contains a lot of unpleasing artifacts

To address these limitations, this project explores an improved approach, Lapstyle, which incorporates a Laplacian loss function to enhance detail preservation. Fig.2 shows that in the presence of the Laplacian loss, severe Laplacian deviations are eliminated or alleviated, and the stylized image is more faithful to the original content image.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



(a) content image (b) Gatys-style (c) Lapstyle

Figure 2: A visual comparison between Gatys-style and Lapstyle. Due to the Laplacian loss term, the output become more faithful

Additionally, a feedforward neural network is trained to enable real-time style rendering, significantly improving efficiency. The project utilizes datasets such as MS-COCO and Wikiart to evaluate the performance of both methods. The goal is to achieve high-quality, efficient style transfer suitable for practical applications in art and image processing.

2 Related Work

Style transfer combines the content of one image with the style of another, gaining attention for its applications in art and image processing. Gatys et al. [1] pioneered this using convolutional neural networks, but their method required computationally expensive iterative optimization. Johnson et al. [3] later introduced a real-time approach using perceptual losses in a feedforward network, greatly improving efficiency while maintaining content and style fidelity. Li et al. [2] advanced this further with a Laplacian-steered method, enhancing structural coherence and reducing artifacts. Our work builds on these advancements, combining Lapstyle quality with feedforward efficiency.

3 Data

The data used in our project consists of two main components: content images and style images.

For content images, we utilize the validation set of the MS-COCO dataset[1], which contains 5,000 images. These images are used to train the feedforward neural network for style transfer. In addition to MS-COCO, we also include custom photographs taken by ourselves, primarily focusing on natural scenes and landscapes. These images provide a diverse range of content for testing and evaluation.

For style images, the style images are sourced from the WikiArt website[2], which hosts a large collection of artworks from various artists and styles. We selected a subset of these images to represent different artistic styles, such as impressionism, cubism, and abstract art.

For data processing, since the task involves style transfer, the primary preprocessing steps include:

- Resizing the images to a suitable resolution, such as (256, 256), to ensure consistency and compatibility with the neural network architecture.
- normalization: The pixel values of the images are normalized using the mean and standard deviation derived from the ImageNet dataset (mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]). This normalization is applied to each channel (R, G, B) separately and is crucial for stabilizing the training process and improving the convergence of the neural network.

[1]Visit the COCO dataset website: `https://cocodataset.org`.

[2]Visit the WikiArt: `https://www.wikiart.org`.

4 Methods

4.1 Neural Style Transfer

Given a content image $\mathbf{x}_c$ and a style image $\mathbf{x}_s$, let their corresponding CNN features at layer l be denoted as $F_l(\mathbf{x}_c)$ and $F_l(\mathbf{x}_s)$, respectively. Suppose their are N_l filters in layer l, then $F_L(\mathbf{x}) \in \mathcal{R}^{N_l \times M_l(\mathbf{x})}$ is a matrix with N_l rows. Each row of $F_L(\mathbf{x})$ is a feature vector containing $M_l(\mathbf{x}) = H_l(\mathbf{x}) \cdot W_l(\mathbf{x})$ elements where $H_l(\mathbf{x})$ and $W_l(\mathbf{x})$ are the height and width of each feature map, depending on the input image $\mathbf{x}$.

Neural style transfer generates an image $\hat{\mathbf{x}}$ that depicts the content of image $\mathbf{x}_c$ in the style of $\mathbf{x}_s$, by minimizing the following loss function of $\hat{\mathbf{x}}$ [1]:

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

where the content loss $\mathcal{L}_{content}$ is the mean-squared distance between the feature maps of $\mathbf{x}_c$ and $\hat{\mathbf{x}}$ at a prespecified content layer l_c:

$$\mathcal{L}_{content} = \frac{1}{N_{l_c} M_{l_c}(\mathbf{x}_c)} \sum_{ij} (F_{l_c}(\hat{\mathbf{x}}) - F_{l_c}(\mathbf{x}_c))^2_{ij}, \tag{2}$$

and the style loss $\mathcal{L}_{style}$ measures the distributional difference of the feature maps of $\mathbf{x}_c$ and $\hat{\mathbf{x}}$ at several prespecified style layers:

$$\begin{aligned} \mathcal{L}_{style} &= \sum_l w_l E_l(\hat{\mathbf{x}}, \mathbf{x}_s) \\ E_l(\hat{\mathbf{x}}, \mathbf{x}_s) &= \frac{1}{N_l^2} \sum_{ij} (G_l(\hat{\mathbf{x}}) - G_l(\mathbf{x}_s))^2_{ij} \end{aligned} \tag{3}$$

where w_l is the weight of layer l, and $G_l(\mathbf{x}) = \frac{1}{M_l(\mathbf{x})} F_l(\mathbf{x})^T F_l(\mathbf{x})$ is the Gram matrix of the feature maps at layer l.

Neural style transfer methods usually employ a pretrained 19 layer VGG network t oextract features and perform the optimization, because VGG preserves more information at the convolutional layers. In the original work [1], Gatys et al. choose 'conv4_2' as the content layer l_c, and 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' and 'conv5_1' as the style layers.

The task of style transfer translates to an optimization proceeds as follows:

1. Set the input of the CNN as the content image $\mathbf{x}_c$. Do a forward propagation and save the response $F_l(\mathbf{x}_c)$ of the content layer;
2. Set the input of the CNN as the style image $\mathbf{x}_s$. Do a forward propagation. Compute and save the Gram matrices $G_l(\mathbf{x}_s)$ of all style layers;
3. Initialize $\hat{\mathbf{x}}$. $\hat{\mathbf{x}}$ could be randomly initialized, or copied from the content image;
4. Iterate until reaching N iterations to update $\hat{\mathbf{x}}$.

4.2 Laplacian loss and Lapstyle

Considering the Laplacian operator Δ of a function, we write the Laplacian filter in the discrete approximation to the two dimensional operator, given by

$$D = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{4}$$

So the Laplacian of an image $\mathbf{x}$ is obtained by convolving the image with D, denoted by $D(\mathbf{x})$. In the original work of Li et al. [2] adopted an approximation for multi-channel images:

$$D(\mathbf{x}) = D(\mathbf{x}^R) + D(\mathbf{x}^G) + D(\mathbf{x}^B) \tag{5}$$

Given two images $\mathbf{x}_c$ and $\hat{\mathbf{x}}$, we define a Laplacian loss as follow:

$$\mathcal{L}_{lap} = \sum_{ij} (D(\mathbf{x}_c) - D(\hat{\mathbf{x}}))^2_{ij} \tag{6}$$

Then the total loss to be minimize changes to:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style} + \gamma\mathcal{L}_{lap} \tag{7}$$

where γ is a tunable parameter controlling the strength of the Laplacian loss. In situations where the output image is severely distorted, we could increase γ to demand a more faithful stylization.

Smoothing with a pool layer The Laplacian filter is sensitive to small perturbations in the input image, and smoothing the input image can make the Laplacian loss better reflect its true detail structures. So Li et al. [2] add a $p \times p$ average pooling layer before the Laplacian layer for smoothing. So the overall network architecture of Lapstyle is shown in Li et al. [2]

Furthermore, When the pooling layer has a wider kernel, it condenses a larger area into one pixel, capturing the structures in larger regions. Hence Li et al. [2] suppose that combining **multiple Laplacian losses** over increasingly dilated pooling layers may capture detail structures in different granularities. The optimization objective is accordingly extended to:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style} + \sum_{k}\gamma_k\mathcal{L}_{lap\mathrm{k}} \tag{8}$$

where $\mathcal{L}_{lap\mathrm{k}}$ is the Laplacian loss on the images pooled by an average pooling layer of size $p_k \times p_k$, and γ_k is its weight.

4.3 Novel Ideas

We have achieved Gatys-style and Lapstyle transfer, however, they perform style transfer on a single image based on a single style and the computational cost is high. So based on the reference [3], we achieved real-time style transfer, where a feed-forward network is trained to solve the optimization problem in real-time. We train a Image Transform Net to transform the input content image C into the output image Y. A loss network is used to compute the content and laplacian loss between the generated image Y and the content image C, and the style loss between the generated image Y and the style image S. Also, the loss network includes total variation loss to calculate the pixel differences of the generated image in the horizontal and vertical directions to make it smoother and reduce noise. After our training and testing, the model can perform style transfer on dozens of images within a second.
The above method only achieves real-time transfer with a single style, and requires separate training for different styles. To address this issue, we have implemented the transfer of any image in any style with Adaptive Instance Normalization(AdaIN) based on the reference [4] and its github repository[3]. In the AdaIN layer, after the content image's feature map is normalized through instance normalization, the mean and variance of the style image are applied to replace them, written in the form of a formula $\hat{x}_{\text{output}} = \sigma_{\text{style}}\left(\frac{x-\mu_{\text{content}}}{\sigma_{\text{content}}}\right) + \mu_{\text{style}}$, thereby achieving style transfer. The AdaIN layer is implemented in the latent space and a decoder is trained to invert the AdaIN output to the image spaces, with a VGG network as the encoder.

5 Experiments

In this section, we present our experiments comparing the Gatys-style method and the Lapstyle method for neural style transfer.

We use two landscape images as content and style images, and compare the generated images using both methods. The results are shown in the following figures.

5.1 Results

The Fig.1 shows the result of Gatys-style method. And the Fig.2 shows the result of Lapstyle method and the comparison between the Gatys-style and Lapstyle method. Also, the Fig.3 shows different content images, pku, city and duck, transferred in different styles, starry_night, Chinese landscape painting and mosaic.

[3]The github repository: `https://github.com/naoto0804/pytorch-AdaIN`.

Figure 3: Example style transfer results

5.2 Analysis

As shown in Fig.1 and Fig.3, the Gatys-style method can transfer content and style relatively well, but the details of the image, especially the edges, appear somewhat blurry. The results of the style transfer experiment show poor color preservation of the content image, with significant distortions and deformations. In some areas, color mixing blurring and mosaic artifacts also occur.

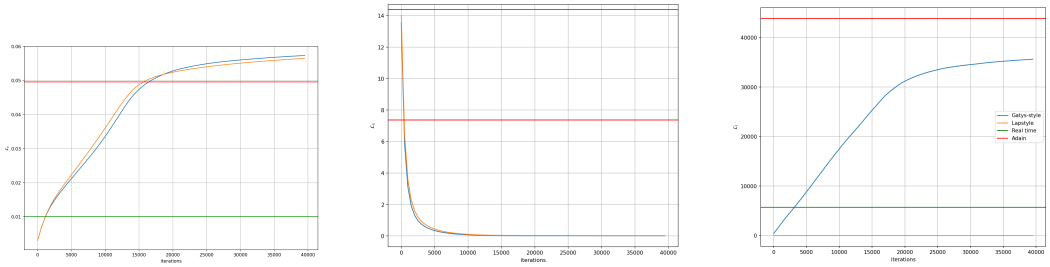
The Lapstyle method, by incorporating Laplacian loss, effectively suppresses artefacts. It significantly improves the texture diversity and quality of the generated image without affecting the original style transfer quality. As a result, the experiment produces images that not only capture some style features of the style image but also retain part of the color from the content image. The images appear clearer, and the overall visual artistic effect has been improved.

5.3 Comparisons

5.3.1 Qualitative comparisons

We have implemented a total of four methods, Gatys-style, Lapstyle, real-time transfer and AdaIN method, and we show transfer results from the four methods in Fig.3. Through qualitative comparisons, we can find that Gatys-style method achieve style transfer well but there are still some unappealing distorted structures and irregular artifacts in the results. In Lapstyle results, these problems are resolved greatly and the images appear clearer. For real-time transfer we trained, the quality of our first two stylized images is quite competitive compared to other methods, which show that the method can achieve real-time transfer while ensuring image quality. In the third example, the colors of the style image were not well transferred, because of our insufficient parameter tuning, insufficient training iterations and inadequate data volume. For the last column, we utilized the AdaIN method. In some cases, it may exhibit distortion such as the pku image and the style couldn't be perfectly transferred such as the duck image, but overall it can achieve the transfer of any image in any style while ensuring high quality of the generated images.

5.3.2 Quantitative comparisons



(a) content loss (b) style loss (c) Lapstyle loss

Figure 4: A visual comparison between Gatys-style and Lapstyle. Due to the Laplacian loss term, the output become more faithful

To compare the four methods quantitatively, we plotted the loss curves using the third example, the duck image in mosaic style, and the results are shown in Fig.4. The content loss and Lapstyle loss curve show an upward trend because we initialized the image as a content image.

From the content loss curve, we conclude that through training, the Gatys-style and Lapstyle images gradually evolve towards the style image, converging after approximately thirty thousand iterations. The AdaIN method's loss is close to the Gatys-style and Lapstyle methods, showing its change from the content image, also reflecting the possible distortion phenomenon observed in the result image. The real-time transfer we trained gets the lowest content loss, aligning with its similarity to the content image.

For the style loss curve, the Gatys-Style and Lapstyle reduce rapidly, inferior to the AdaIN method after approximately one thousand iterations and converges after ten thousand iterations. The AdaIN method has not fit to the style image after all, yet it can still maintain a relatively low loss when applied directly. The real-time transfer we trained has relatively high style loss because unreasonable weight settings potentially.

For the Lapstyle loss curve, the Lapstyle method is near to zero and the real-time transfer's loss is low. This aligns with the fact that artefacts rarely appear in these two types of images, proving the effectiveness of the improvements made to the Laplacian method. The Gatys-Style's loss and AdaIN method's Lapstyle loss are both high, aligning with the potential unappealing distorted structures in these two types of images.

Method	Time	Style	Pre-training on specific styles
Gatys-Style	53.532	1	Yes
Lapstyle	73.544	1	Yes
real-time	0.048	∞	Yes
AdaIN	0.067	∞	No

Table 1: Time and style comparisons

Meanwhile, we compare the time and style from the four methods. The Gatys-Style and Lapstyle method train an image for style transfer with 10000 iterations. All the methods produce 512*512 images. So from the Tab.1, We can see that the last two methods have significant advantages in terms of time and the variety of styles, making the task of style transfer more universal. At the same time, the AdaIN method does not require pre-training on specific styles, which greatly increases flexibility.

6 Conclusion

In this paper, we proposed and compared two neural style transfer methods: the original Gatys-style method and the Lapstyle method. The experiments demonstrate that the Lapstyle method significantly

improves the quality of the stylized images by preserving finer details and enhancing the overall structure of the content.

The addition of Laplacian loss in the Lapstyle method proves effective in maintaining edges and textures, which are often lost in traditional methods. This result suggests that Lapstyle may be more suitable for complex images with intricate structures. We also achieved real-time transfer and AdaIN method and proved their effectiveness.

Future work may explore further improvements to Lapstyle, such as experimenting with alternative architectures such as StyleGAN or loss functions, to achieve even better results.

References

[1] Gatys, L.A., Ecker, A.S. & Bethge, M. (2016) Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423. Piscataway, NJ: IEEE.

[2] Li, S., Zhao, Q., Jiang, Y., Liu, Y., Zhang, Y. & Tang, J. (2017) Laplacian-steered neural style transfer. In *Proceedings of the 25th ACM international conference on Multimedia*, pp. 1716–1724. New York, NY: ACM.

[3] Johnson, J., Alahi, A. & Fei-Fei, L. (2016) Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 694–711. Cham: Springer International Publishing.

[4] Huang, Xun, and Serge Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization."Proceedings of the IEEE international conference on computer vision. 2017.