Style Transfer: Laplacian, Depth loss and AdaIN

Zhang Zhaoji
Peking Univ.
2100011726@stu.pku.edu.cn

Jing Dinghao
Peking Univ.
2100013166@stu.pku.edu.cn

Peng Yicheng
Peking Univ.
2100013167@stu.pku.edu.cn

Abstract

It is a key research topic in the field of image processing to achieve image style transfer by preserving high-level semantic content while extracting textures for rendering. Previous researchers have achieved many satisfactory results in this field using convolutional neural networks. Among these results, we have replicated Gatys-style/Lap-style neural style transfer, arbitrary style fast patch-based style transfer, and AdaIN. Furthermore, we attempt to further optimize the neural style transfer algorithm by introducing depth loss and Laplacian loss into the loss function of AdaIN. Experimental results demonstrate that the optimized AdaIN not only enhances the naturalness of image details but also restores the textures of the style image to a greater extent.

1. Introduction

Nearly 41,000 years ago, the earliest known paintings were painted on the rock walls of Altamira in Spain [1]. Since then, the art of painting has become one of the most critical parts of human art. Since the mid-1990s, computer scientists have also begun to explore this field [3].

Some early research focused on non-photorealistic rendering in the field of computer graphics (NPR) [4,5] and texture synthesis technology in the field of computer vision [6]. However, most NPR methods can only be used in specific areas such as animation rendering and lack generalization ability, while texture synthesis technology can only utilize primitive image features and lacks the ability to utilize image structure. The migration of style between images is essentially a problem of texture migration. The goal of texture migration is to synthesize texture on the target image while preserving semantic information of the target image. A series of previous research results are limited to using only low-level image features for texture migration, while the ideal style transfer algorithm should first extract semantic content from the target image, and then use the style of the source image to render the extracted semantic content through the texture migration process.

In 2015, Gatys et.al [7] proposed a novel method achieve impressive visual quality come from the use of convolutional neural networks (CNN) for feature extraction. This and similar methods are based on optimizing a complex pixel-wise problem which can take minutes to solve. Many works have been done to improve the problem or to use feed forward neural networks to decrease processing cost [9, 10, 19], which finally could achieve almost real-time style transfer.

2. Related Work

Style Transfer as Optimization. In 2015, Gatys et al. first used convolutional neural networks to study this problem [7]. They used a pre-trained VGG model [8] to extract the content and style of two images, and used the calculated Gram matrix to iteratively update the image, thereby outputting a new image with reference style and original image content. By using labeled datasets to train convolutional neural networks, it is possible to extract high-level semantic content of images in general scenarios. Furthermore, this neural network can also complete advanced visual information processing tasks such as texture recognition and artistic style classification. This work has opened up a new field known as neural style transfer (NST, Neural Style Transfer).

Gatys's algorithm is undoubtedly better than previous algorithms, but it has not completely solved this problem. It is noticed that many details of the content image are lost in the synthetic image, and iteration methods cost enormously large amount of time.

Optimization Style Transfer with Laplacian Loss. In order to maintain continuity of microstructure in the style transfer process, Li Shaohua and others at Nanyang Technological University added Laplacian loss to describe the detailed content of the image on the basis of Gatys's algorithm [9]. They claimed that this could lead to more detailed images in terms of details and be less prone to unnatural defects. However, this work did not consider problems such as semantics, depth, and brushstroke changes.

Fast Style Transfer Using Style Swap. Tian Qi Chen from University of British Columbia proposed Style Swap

[10], a new solution to speed up complex optimization according to finding a closest matching style patch based on the normalized cross correlation measure. Chen claimed Style Swap iteration could be 10 times faster than Gatys's algorithm. Chen also trained an inverse CNN to derive synthesis image, which is 50 times faster. But the speed is still not fast enough for real-time image processing.

Adaptive Instance Normalization. Gatys-style/ lap-style neural style transfer has achieved satisfactory results in the task scenario given texture images and reference images. However, it has obvious deficiency: for each texture image and reference image, a separate convolutional neural network needs to be trained. If it is in the task scenario of mass neural style transfer of different texture images and reference images, the efficiency of this algorithm is unacceptable. Therefore, researchers creatively proposed Adaptive Instance Normalization, which can achieve real-time neural style transfer. That is, under the condition of basically no loss of style transfer quality, only one neural network needs to be trained to support users uploading arbitrary reference images and texture images for neural style transfer. In addition, AdaIN also allows users to manually control the weight of reference images and texture images in the output image, which means that users can flexibly adjust the degree of image stylization.

We have replicated four research results: Gatys-style neural style transfer, Lap-style neural style transfer, fast patch-based style transfer of arbitrary style, and AdaIN. At the same time, we observed that AdaIN has a relatively rough design in terms of loss function. It directly uses simple Euclidean distance as the loss function. Based on this, we attempted to introduce depth loss and Laplacian loss into AdaIN's loss function for further optimization. After experiments, we observed that after introducing these two loss functions, AdaIN's output images have more smooth and natural details, while also more greatly restoring the style of texture images.

Replication. In our paper replication work on Gatys-style/lap-style neural style transfer, we used the NPRgeneral dataset[13], which is a standardized image collection for evaluating image stylization algorithms, and texture images were painted using oil painting techniques.

Dataset. We use Microsoft COCO image dataset [11] and wikiart dataset hosted by kaggle [12] to train our self-made AdaIN models. COCO is a dataset containing roughly 80,000 images of common objects in daily life. Due to time constraints, we utilized approximately three-fourths of the data from the COCO dataset. The WikiArt dataset is a large-scale collection of art pieces, encompassing works from different periods, styles, and artists. We utilize images in WikiArt as our style images.

3. Methods & Experiments

3.1. Technical Approach

Gatys Style Transfer. The method proposed by Gatys et al. (Gatys-style) utilizes the strong feature extraction ability of convolutional neural networks to abstract stylized features and content structural features separately, guiding the generation of target images from two aspects. The organization of the method takes the pretrained VGG network as the main body. After the content image passes through the VGG network, it can obtain different levels of features. The features of the high-level layer are selected as the content representation, and the MSE loss is calculated by comparing the corresponding layer outputs of the target image, guiding our generation. Let $\vec{p}$ and $\vec{x}$ be the original image and the image that is generated, and P^l, F^l be the respective feature representation in layer l. The squared-error loss between the two feature representations is

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}\left(F_{ij}^l - P_{ij}^l\right)^2$$

On the other hand, after the styled image passes through the VGG network, corresponding features are output at different layers. As we are not concerned about the layout information features in the styled image, we only focus on style features. These feature correlations are given by the Gram matrix $G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$.

Let $\vec{p}$ and $\vec{x}$ be the original image and the image that is generated, and P^l, F^l be the respective feature representation in layer l. The squared-error loss between the two feature representations is

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}\left(G_{ij}^l - A_{ij}^l\right)^2$$

And we write the style loss as

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

Therefore, we calculate the autocorrelation feature of the feature using the Gram matrix and use this feature as a benchmark for aligning the target image, guiding its generation. Therefore, the main loss of Gatys-style consists of these two parts, namely, content loss and style loss.

$$\mathcal{L}_{total}(\vec{p}, \vec{x}, \vec{a}) = \alpha\mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta\mathcal{L}_{style}(\vec{a}, \vec{x})$$

The gradients of $\mathcal{L}_{total}$ with respect to the pixel values $\vec{x}$ could be computed to back propagation.

Laplacian Style Transfer. Li Shaohua and others at Nanyang Technological University have made optimizations on the basis of Gatys. The Laplacian filter itself is applied to edge detection in images, which is

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

And we can measure the difference of Laplacians as

$$\mathcal{L}_{lap} = \sum_{ij} \left(D_{\vec{p}} - D_{\vec{x}}\right)^2_{ij}$$

Here, in order to retain more structural information of the content image in the target image, both the content image and the target image are passed through the Laplacian filter. An alignment is made for the output results, which is denoted as Laplacian loss. The loss is added from the original two parts to three parts.

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

Style Swap. Let C and S be the representations of content and style images, and let $\Phi(\cdot)$ be the function represented by a fully convolutional part of a pretrained CNN that maps an image to some intermediate activation space. After calculating two sets of patches, $\{\phi_i(C)\}_{i\in n_c}$ and $\{\phi_i(S)\}_{i\in n_s}$. For every content activation patch, determine a closest matching style patch based on the normalized cross correlation measure,

$$\phi_i^{ss}(C,S) = argmax_{\phi_j(S), j=1,2,\ldots n_s} \frac{< \phi_i(C), \phi_i(S) >}{||\phi_i(C)|| \cdot ||\phi_i(S)||}$$

And swap each content activation patch $\phi_i(C)$ with its closest-matching style patch $\phi_i^{ss}(C,S)$. Thus reconstruct the complete content activations finally.

Adaptive Instance Normalization. To overcome the limitations of previous methods, which are often confined to a predetermined set of styles, Huang et al. introduced a novel approach that enables real-time arbitrary style transfer for the first time. The method is called *adaptive instance normalization* (AdaIN)
To fully interpret this method, we should start with Batch Normalization [14], Batch Normalization layers are designed to normalize data within a neural network, aiding in training by ensuring consistent data distributions across different layers. Given an input batch $x \in R^{N\times H\times C\times W}$, BN normalizes the mean and standard deviation for each individual feature channel:

$$BN(x) = \gamma(\frac{x - \mu(x)}{\sigma(x)}) + \beta$$

$\mu(x)$ and $\sigma(x)$ refers to mean and standard deviation respectively. α and β are affine parameters learned from data.

Ulyanov et al [15] surprisingly discovered that replacing Batch Normalization with Instance Normalization improves the performance of the feed-forward stylization method:

$$IN(x) = \gamma\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta$$

Different from BN, IN computes the $\mu(x)$ and $\sigma(x)$ across spatial dimensions independently for each channel and sample.

Instead of learning a single set of affine parameters γ and β, Dumoulin et al[16] proposed a conditional instance normalization (CIN) layer that learns a different set of parameter γ^* and β^* for each style s :

$$CIN(x{:}\, s) = \gamma^*\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \beta^*$$

This is based on the discovery that affine parameters can significantly influence the style of generated images. Even when using the same convolutional parameters, different affine parameters result in different styles.

Enlightened by the conclusion that affine parameters have a significant impact on the style of generated images, Huang et al. [17] explore the possibility of adapting these parameters to arbitrary given styles through the use of adaptive affine transformations (AdaIN). AdaIN receives a content input x and a style input y, and simply aligns the channel-wise mean and variance of x to match those of y:

$$AdaIN(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

$\sigma(y)$ and $\mu(y)$ refers to the standard deviation and the mean of style image y. AdaIN performs style transfer in the feature space by transferring feature statistics, specifically the channel-wise mean and variance.

The architecture of AdaIN takes a content image c and an arbitrary style image s as input. Huang et al adopt a encoder f (pre-trained VGG-19) to extract features and an AdaIN layer to align features.

$$t = AdaIN\big(f(c), f(s)\big)$$

The loss contains two parts: content loss and style loss. Content loss is computed as:

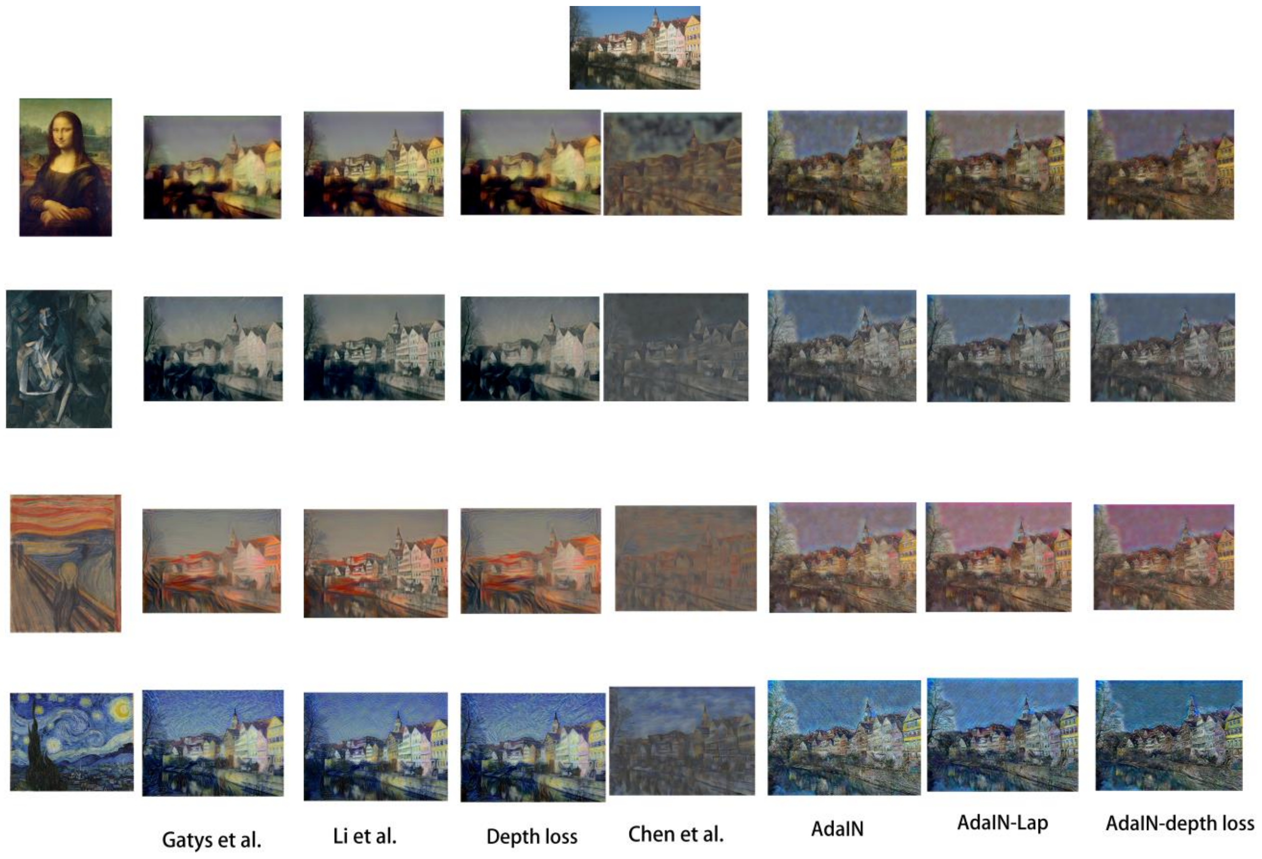


Figure 1 Example Style Transfer Results.

$$\mathcal{L}_c = ||f(g(t) - t||_2$$

where g refers to a randomly initialized decoder. The style loss is computed as:

$$\mathcal{L}_s = \sum_{i=1}^{L} \left|\left|\mu\left(\varphi_i\left(g(t)\right)\right) - \mu\left(\varphi_i(s)\right)\right|\right|_2 + \sum_{i=1}^{L} \left|\left|\mu\left(\sigma_i\left(g(t)\right)\right) - \mu\left(\sigma_i(s)\right)\right|\right|_2$$

And the final loss is:

$$\mathcal{L}_{total} = \mathcal{L}_c + \alpha\mathcal{L}_s$$

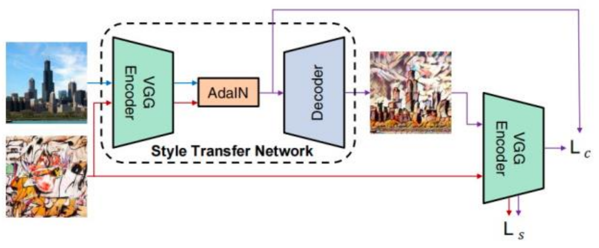


Figure 2: AdaIN Architecture

Novel ideas.

In analyzing the architecture figure of the previous AdaIN method, we observe that its content loss computation part is rather crude. Here, we raise two skeptical questions. Firstly, since t has already undergone AdaIN layers and is thus stylized, using the MSE loss between the obtained feature and t as the content loss seems mismatched. Secondly, obtaining the feature by passing t through a decoder and then an encoder reflects more on the differences in the encoder-decoder framework rather than being closely related to the content loss.

Motivated by this, and combining with the findings from the work of Li Shaohua et al., we considered an improvement method, which involves adding a Laplacian loss on top of the AdaIN-based architecture to preserve stronger structural information. We obtain corresponding images from t obtained through the decoder and the original image, pass them through pooling layers and Laplacian filters, and calculate the MSE loss between the resulting images as our Laplacian loss (lap_loss).

$$L_{lap} = ||lp(g(t)) - lp(\,content\_img)||_2$$

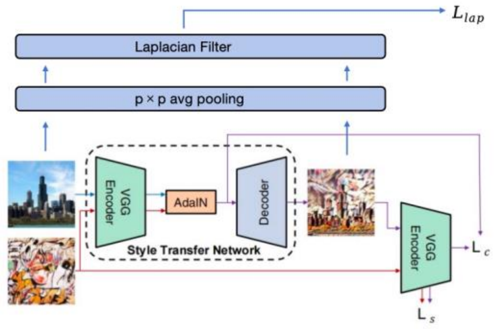


Figure 3: AdaIN + lap_loss

We noticed that all these traditional ideas do not take the depth information form the content pictures into consider, so we use a fast monocular depth estimator, monodepth2[18], adding depth losses into optimization problem. We define the depth loss as:

$$\mathcal{L}_{depth} = \left||d(\vec{p}) - d(\vec{x})|\right|$$

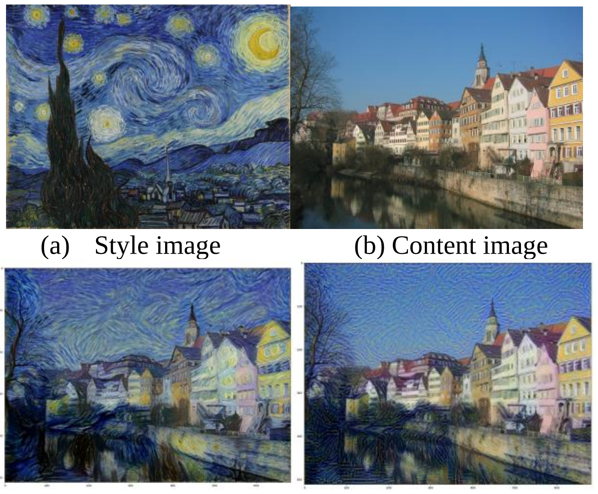
And let the total loss function be:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style} + \gamma\mathcal{L}_{depth}$$

However, adding Laplacian loss or depth loss to original Gatys's loss still need to meet the same problem that complex optimization consumes plenty of time. We apply these losses into the AdaIN framework to train a decoder, which could one time feed forward neural network to derive a new image.

3.2. Experiments

We have implemented the reproduction of Gatys-style neural style transfer and Lapstyle neural style transfer. By selecting human photos or landscape images as the base images and Vincent van Gogh's Starry Night as the texture, we can see that the current reproduction algorithm has achieved good results, achieving a relatively natural and successful style transfer that extracts high-level semantic content.


(a) Style image (b) Content image

(c) Gatys-style output (d) Lapstyle output

Figure 4 Style transfer of Gatys and Laplacian Loss

Compared to Gatys-style neural style transfer, the details of the Lapstyle neural style transfer images appear to be more natural, while the overall style of the images is smoother and gentler. We believe that the reason for this difference is that the newly introduced lap-loss pays more attention to the differences in details, which then affects the overall style of the image.

We Also tested replicated 4 algorithms and 3 algorithms we implementing, and the result is shown in the Figure 1. We could find that both Lap and AdaIN-Lap algorithms have better maintenance of detailed feature than Gatys's style transfer, but lost some of brush strokes texture. Applying depth loss in iteration optimization may not have big influence in style transfer, but may have more importance in AdaIN-Depth algorithm, which adjust the color to keep better detailed depth features.

Iteration optimization methods also make better styled textures of monotonous features like the sky part. In AdaIN-processed images, the upper sky is occupied by jittering-like noisy background patterns, while in Gatys-like iteration optimizations have uniformly styled backgrounds.

4. Conclusion

In this project, our main focus was to explore the task of neural style transfer (NST). With just two images provided, we are able to achieve style transfer effectively. We first delved into the pioneering work in this field, originating from Gatys et al.[7], which utilizes the feature extraction capability of the VGG network to achieve style transfer. Building upon the work of Gatys et al.[7], several other improvements have emerged, such as the addition of laplacian loss proposed by Li Shaohua et al.[9]. With the development of Neural Style Transfer (NST), the demand has shifted away from single-style constrained feed-forward networks. This has led to the emergence of works primarily based on Adaptive Instance Normalization (AdaIN)[17], facilitating real-time arbitrary style transfer and achieving more versatile transfer networks.

We successfully replicated the classic or mainstream NST methods mentioned above and proposed our own insights, leading to improvements. For example, we adjusted the loss function of AdaIN to preserve stronger structural information or retain more depth information. We compared the results of the improved method with the original paper's method and found that while we achieved improvements in certain aspects as desired, we might also lose some other information, such as brush strokes, in the process. This insight suggests that in NST tasks, it may be necessary to balance between different objectives to achieve the desired results.

For the future development of NST tasks, we anticipate leveraging the rapidly evolving transformer models. There are already works such as AdaAttN [20] that combine

AdaIN and transformer, which we believe will play a significant role in advancing NST further. At the same time, we believe that alternative NST frameworks may emerge, such as leveraging popular generative models like LDM [21] for NST tasks. During the generation process, one could inject style features to achieve the desired stylization.

5. Code Usage

We designed our Gatys-style, Laplcian style and depth loss iteration algorithms on the basis of Gatys's implementation code [22]. The monocular depth estimation model, monodepth2, could be found at [23]. The Tiqi Chen's Style Swap algorithm could be fetched at [24]. And our AdaIN-Lap, AdaIN-depth loss and original AdaIN code are adapted on the basis of work [25]

6. Contribution

Zhang Zhaoji wrote the origin Gatys's code, Gatys-depth loss code and AdaIN-depth code, Jing Dinghao wrote the Laplacian loss code, AdaIN-Lap code and trained feed forward neural networks. Peng Yicheng mainly wrote the final report and all authors wrote the report together.

References

[1] Pike, A. W. G.; Hoffmann, D. L.; García-Diez, M.; Pettitt, P. B.; Alcolea, J.; De Balbín, R.; González-Sainz, C.; de las Heras, C.; Lasheras, J. A.; Montes, R.; Zilhão, J., U-Series Dating of Paleolithic Art in 11 Caves in Spain. *Science* **2012,** *336* (6087), 1409-1413.

[2] Carrasco, I. How much money was Michelangelo paid to paint the Sistine Chapel

[3] Jing, Y.; Yang, Y.; Feng, Z.; Ye, J.; Yu, Y.; Song, M., Neural Style Transfer: A Review. *IEEE Transactions on Visualization and Computer Graphics* **2020,** *26* (11), 3365-3385.

[4] In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, Dallas, TX, USA, Association for Computing Machinery: Dallas, TX, USA, **1990.**

[5] Schofield, S. Non-photorealistic rendering: a critical examination and proposed system. **1994**.

[6] Efros, A. A.; Freeman, W. T., Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, Association for Computing Machinery: **2001**; pp 341–346.

[7] Gatys, L. A.; Ecker, A. S.; Bethge, M., A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* **2015**.

[8] Simonyan, K.; Zisserman, A., Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* **2014**.

[9] Li, S.; Xu, X.; Nie, L.; Chua, T.-S., Laplacian-Steered Neural Style Transfer. In *Proceedings of the 25th ACM international conference on Multimedia*, Association for Computing Machinery: Mountain View, California, USA, **2017**; pp 1716–1724.

[10] Chen T Q, Schmidt M. Fast patch-based style transfer of arbitrary style. arXiv preprint arXiv:1612.04337, **2016**.

[11] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollar, and ´ C. L. Zitnick. Microsoft COCO: common objects in context. CoRR, abs/1405.0312, 2014.

[12] S. Y. Duck. Painter by numbers, wikiart.org. https:// www.kaggle.com/c/painter-by-numbers, 2016

[13] The Graphics, Imaging, and Games Lab, or GIGL, Carleton University,Ottawa,Canada.https://gigl.scs.carleton.ca/benchmark_npr_general

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. InJMLR, 2015. 2

[15] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In CVPR, 2017. 1, 2, 3, 5, 6, 7, 8

[16] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In ICLR, 2017. 1, 2, 3, 5, 6, 7

[17] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE International Conference on Computer Vision, pages 1501–1510, 2017. 1, 2, 3, 4, 5, 7, 8]

[18] Godard, Clément, et al. "Digging into self-supervised monocular depth estimation." Proceedings of the IEEE/CVF international conference on computer vision. 2019.

[19] Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer International Publishing, 2016.

[20] Liu S, Lin T, He D, et al. Adaattn: Revisit attention mechanism in arbitrary neural style transfer[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 6649-6658.

[21] Rombach R, Blattmann A, Lorenz D, et al. High-resolution image synthesis with latent diffusion models[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 10684-10695.

[22] https://github.com/leongatys/PytorchNeuralStyleTransfer

[23] https://github.com/nianticlabs/monodepth2

[24] https://github.com/irasin/Pytorch_Style_Swap

[25] https://github.com/irasin/Pytorch_AdaIN