Advancing Traditional Neural Style Transfer: Realistic Neural Style Transfer Architecture That Addresses Limitations With Abstract Art Styles And Photographic Input

1 Motivation

In recent years, AI driven image stylization has captivated broad audiences as exemplified by the viral "Studio Ghibli" trend on social media, where everyday photographs were stylized with the vivid colors and detailed textures from Studio Ghibli's animated films. This trend underscores the appeal of neural style transfer techniques for creating various artistic imagery, which is a traditional framework that introduces unique representations to "separate and recombine" content and style images based on users' inputs (Gatys et al., 2016). However, alongside the excitement, this trend also exposed fundamental limitations of traditional style transfer methods. We observed that applying a highly abstract art style to a photographic image can yield distorted and incoherent outputs. By matching local patterns or textures between the style and content features, traditional algorithms work well when the style image provides rich textures or strokes the content. However, it breaks down for abstract art where no clear patch correspondences exist. Without discrete shapes to guide the model, patch based methods may introduce arbitrary noise or overly blur the output, which fails to capture the intended artistic effect.

We propose a new approach to neural style transfer that better handles the abstract and non figurative style references. Our approach aims to preserve the global structure of the content image while transferring the global color distribution of the style image where the output retains the content image's structures with the style image's overall color palette, tone, and mood. By preserving global structure from the content, the method avoids the distortions and semantic mismatches that other techniques when faced with ambiguous styles. As one can see, with our new proposed architecture, it will unlock new possibilities in computational artwork and creative AI tools. It would enable artists to apply highly abstract or minimalist art styles to photographs without sacrificing content fidelity. This leads to more creative applications such as transferring the color atmosphere of a Rothko painting onto a landscape photo while keeping the scene's geometry or recoloring graphics with the mood of an abstract artwork without manual editing. In the long term, we hope to utilize this research to develop smarter artistic AI systems that can understand and manipulate high level artworks. By addressing those limitations in current style transfer methods, our approach generates more general and reliable stylization results, which improves the toolbox for both digital artists and everyday content creators.

2 Approach

By evaluating Google's pre-trained Neural style transfer model with a Picasso painting as the style image and a skyline as the content image, we notice the model's convolutional layers have significant impacts to the stylization processsince certain convolutional layers hold greater weight in capturing stylistic features. Building on these findings, we manually adjust the layer weightings and compare the outputs across different configurations. We evaluate results using style loss, content loss, and various metrics to quantify style fidelity and content preservation. After running several trials, we realize that existing models have limited capabilities in preserving straight lines and boundaries inside content. To improve that, we include different edge loss functions to preserve edges and straight lines.

3 Implementation Details

This project is implemented in Python by utilizing both the PyTorch and TensorFlow frameworks to compare baseline models and our proposed neural style transfer architecture. We utilized TensorFlow and TensorFlow Hub to explore traditional neural style transfer architecture. In order to implement our proposed method, we chose PyTorch with Google pre-trained VGG19 model and additional libraries. For baseline model, we recreated the traditional neural style transfer model in Tensorflow with a pre-trained VGG19 model. For hardware and environment, our experiments were primarily conducted on CPU and we use to A100 GPU from Google Colab to speed up the training process. For external resources, our style and content images were obtained from publicly available image datasets and online art repositories.

4 Final Results

In order to measure the success of our proposed neural style transfer architecture, we choose the traditional neural style transfer algorithm from Google's TensorFlow library and famous visual language model ChatGPT 4o as our baseline models. Our method significantly improves performance over the TensorFlow model as it enhances SSIM by 77% and MS-SSIM by 49% as well as reducing perceptual error metrics LPIPS by 23% and DISTS by 17%. Compared to ChatGPT-4o model, the improvements are even more noticeable as it increases 342% in SSIM and 536% in MS-SSIM as well as reducing LPIPS errors by 35% and DISTS errors by 25%. For more detailed information, you could visit our github page `https://github.com/Edward-H26/Realistic-Neural-Style-Transfer-Architecture` and our slides `https://www.canva.com/design/DAGlL4DAJlQ/DlvTjAIlUQvyPsy7m5auTw/edit`.

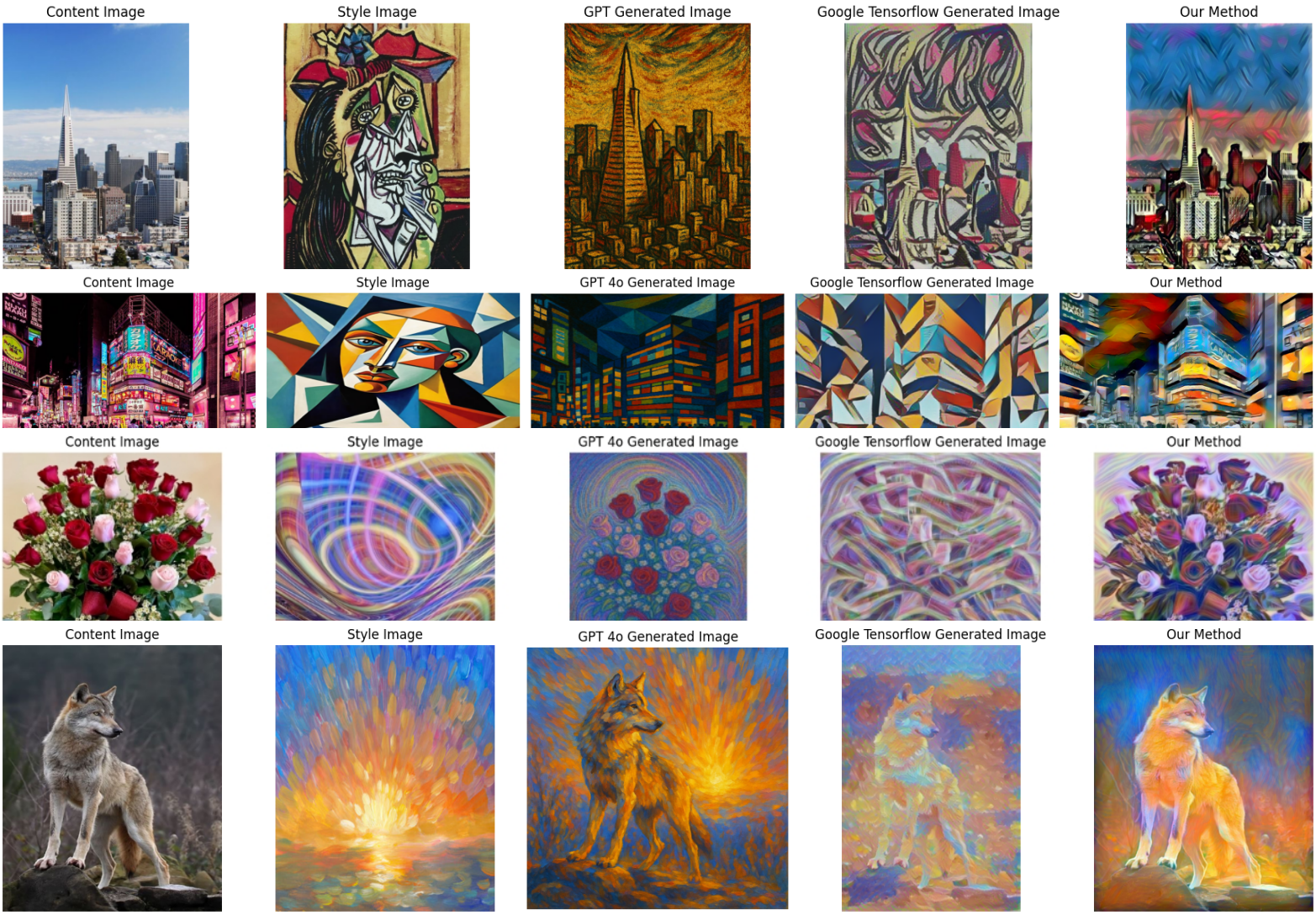

Figure 1: Result Comparison

Metric	Model	Set 1	Set 2	Set 3	Set 4	Mean
SSIM	Visual Language Model: ChatGPT 4o	0.052955	0.077377	0.064246	0.264646	0.114806
	Traditional Method: Google Tensorflow	0.258627	0.174317	0.275503	0.437423	0.286468
	Our Proposed Method	**0.518870**	**0.270604**	**0.605339**	**0.635151**	**0.507491**
MS-SSIM	Visual Language Model: ChatGPT 4o	0.138054	0.086546	0.050213	0.130794	0.101402
	Traditional Method: Google Tensorflow	0.403297	0.427665	0.424199	0.472805	0.431991
	Our Proposed Method	**0.604274**	**0.533430**	**0.787865**	**0.653285**	**0.644713**
LPIPS	Visual Language Model: ChatGPT 4o	0.809233	0.567288	0.765684	0.645421	0.696907
	Traditional Method: Google Tensorflow	0.596268	0.595597	0.583921	0.550891	0.581669
	Our Proposed Method	**0.511587**	**0.488956**	**0.362220**	**0.438489**	**0.450313**
DISTS	Visual Language Model: ChatGPT 4o	0.453987	0.424876	0.500621	0.486678	0.466540
	Traditional Method: Google Tensorflow	0.405118	0.398024	0.436548	0.448619	0.422077
	Our Proposed Method	**0.369006**	**0.306310**	**0.351140**	**0.376602**	**0.350765**

Table 1: Performance Comparison Across Methods

5 Challenge / Innovation

5.1 Traditional Single-layer Content Loss Function

Introduced by Gatys and his colleagues in “A Neural Algorithm of Artistic Style”, traditional neural style transfer combines the “content of one image with the style” of another image by using deep convolutional neural networks (Gatys et al, 2016). This process involves utilizing a pre-trained VGG neural network to extra feature maps and an “iterative optimization process” that incorporates

content loss and style loss functions to generate stylized image (Liu et al., 2023). The traditional content loss function maintains the structural information of the content image in the generated image, which is usually computed as the mean squared error between feature maps at a single chosen layer:

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

where $\tilde{p}$ is the original content image, $\tilde{x}$ is the generated image, l is the chosen layer, and F_{ij}^{l}, P_{ij}^{l} are the feature activations at positions i, j in layer l with respect to $\tilde{x}$ and $\tilde{p}$.

Based on the Gatys et al's research, they used higher layer "conv4_2" as their content loss function to preserve "content in terms of objects and their arrangement" from the content image since the higher layers can effectively identify and represent object level structure without constraining the "exact pixel values" of the reconstruction (Gatys et al, 2016). However, the obtained features "focus more on a central object" which leads to generated images that match the content images in overall structures but lack detailed information (Kotiyal, 2021). Specifically, when the input content image is an photographic image, the single-layer content loss function often fails to capture low-level details due to the single chosen layer and the generated images that may appear overly stylized and less realistic.

5.2 Proposed Muti-layer Content Loss Function

Unlike the traditional content loss that oversimplifies texture representation, we propose a multi-layer content loss function to address these shortcomings by incorporating features from multiple layers. With "multiple layers" to extract "hierarchical features" for the model, the content loss function can capture both low-level details and high-level structures simultaneously (Zhang et al., 2021). Thus, we can emphasize the output of some of these layers as content features or style features to create more photo realistic results by assigning different weights to specific layers:

$$\mathcal{L}_{\text{content}}^{\text{multi-layer}}(\tilde{p}, \tilde{x}) = \sum_{l \in L} w_l \left(\frac{1}{N_l M_l} \sum_{i=1}^{N_l} \sum_{j=1}^{M_l} \left(F_{ij}^{l} - P_{ij}^{l} \right)^2 \right) \tag{2}$$

where $\tilde{p}$ is the original content image, $\tilde{x}$ is the generated image, l is the chosen layer, $l \in L$ is the set of selected layers, w_l are weights, N_l is the number of filters, M_l is the number of positions, and F_{ij}^{l}, P_{ij}^{l} are feature activations.

By measuring the content image at various levels, our multi-layer content loss function preserves both the structural integrity and the intricate details, which leads to a more realistic output. With the contributions of different weights w_l in the multi-layer content loss, they can better seek the balance between high-level and low-level features, which creates flexibility to adapt to different photographic content.

5.3 Traditional Laplacian Edge Loss Function With Muti-layer Content Loss Function

Traditional loss functions that incorporate mean squared error tend to generate images that are "overly smooth and blurry" because mean squared error functions do not take image edges and structures into account (Seif, G., & Androutsos, D., 2018). On the other hand, the Laplacian loss function "directly optimizes the edge pixels" in the deep neural network to produce sharp edges and detailed images during image reconstruction, which addresses fundamental limitations in traditional approaches (Seif, G., & Androutsos, D., 2018). The Laplacian loss works by detecting edges and details in an image using the Laplacian of Gaussian operator and then minimizing the difference between the Laplacian of Gaussian of the input image and the Laplacian of Gaussian of the output image:

$$L_{\text{laplacian}} = \frac{1}{C\,H\,W} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} \left((\Delta \tilde{p})_{c,h,w} - (\Delta \tilde{x})_{c,h,w} \right)^2$$

where C represents the number of channels, H represents the height dimensions, W represents the width dimensions, $\Delta \tilde{p}_{c,h,w}$ is the Laplacian of the content image $\tilde{p}$ at channel c, height h, and width w, and $\Delta \tilde{x}_{c,h,w}$ is the Laplacian of the generated image $\tilde{x}$ at channel c, height h, and width w.

Furthermore, we can combine traditional Laplacian loss function with the multi-layer content loss to create the total loss function to generate realistic images by preserving edges and straight lines as well as the overall structure:

$$L_{\text{total}} = \alpha \cdot L_{\text{content}} + \beta \cdot L_{\text{style}} + \gamma \cdot L_{\text{laplacian}}$$

where L_{content} is the multi-layer content loss function L_{style} is the style loss function, $L_{\text{laplacian}}$ is the Laplacian edge loss function, α is the parameter to control the influence of the content loss function, β is the parameter to control the influence of the style loss function, and γ is the parameter to control the influence of the Laplacian edge loss function.

The Laplacian loss function is sensitive to "high-frequency components of the image" that are correspond to edges and details because it "relies on the second derivative of the image intensity" rather than directional gradients (Wang, 2007). In contrast to other loss functions, the Laplacian loss functions highlights areas of rapid intensity change across all the color channels. As a result, with Laplacian loss function and muti-layer content loss, the total content loss preserves both high-level semantic information and low-level textural details from the content image.

5.4 Proposed Sobel Edge Loss Function With Muti-layer Content Loss Function

Despite the inclusion of multi-layer content loss and Laplacian edge loss, the generated images exhibit curved lines and distorted edges; it does not fully prioritize edge preservation in this case. While the Laplacian edge loss introduced in the previous section helps to preserve fine content details, it suffers from a fundamental limitation. The Laplacian operator is "a 2-D isotropic measure of the 2nd spatial derivative of an image" that highlights regions of rapid intensity change (Fisher et al., 2023). However, as Wang pointed out, the Laplacian operator may "lacks directional sensitivity" compared with the first derivative-based edge detectors such as Sobel operator (Wang, 2007).

As a consequence, a Laplacian based loss cannot distinguish, for example, a perfectly vertical edge from the same edge slightly rotated. This gap is critical because even slight distortions in edge orientation can lead to noticeable artifacts in the generated image, especially for photographic content with prominent linear structures like architecture. Previous neural style transfer results without structural constraints often exhibited such distortions with "wavy strokes " that can adversely affect the original content (Reimann et al., 2022). Therefore, for perfect photorealistic stylization, it is imperative that these straight edges in the content remain straight in the result.

To address the drawbacks of the traditional Laplacian edge loss function, we introduce Sobel edge loss function instead of the Laplacian loss with our muti-layer content loss function to better preserve edges and straight lines. The Sobel edge loss is calculated as:

$$L_{\text{sobel}} = \frac{1}{B \cdot C \cdot H \cdot W} \sum_{b=1}^{B} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} \left[(\tilde{x}_x(\tilde{p})_{b,c,h,w} - \tilde{x}_x(\tilde{x})_{b,c,h,w})^2 + (\tilde{x}_y(\tilde{p})_{b,c,h,w} - \tilde{x}_y(\tilde{x})_{b,c,h,w})^2 \right]$$

where B is the batch size, C is the number of channels, H is the height of the feature maps, W is the width of the feature maps, $\tilde{p}$ represents the content image, $\tilde{x}$ represents the generated image, $\tilde{x}_x(\tilde{p})_{b,c,h,w}$ denotes the horizontal Sobel operator in the content image at position b, c, h, w, $\tilde{x}_y(\tilde{p})_{b,c,h,w}$ denotes the vertical Sobel operator in the content image at position b, c, h, w, $\tilde{x}_x(\tilde{x})_{b,c,h,w}$ corresponds to the horizontal Sobel operator in the generated image at position b, c, h, w, $\tilde{x}_y(\tilde{x})_{b,c,h,w}$ are the corresponds to the vertical Sobel operator in the generated image at position b, c, h, w, $\tilde{x}_x$ is the Sobel operator for horizontal gradients that is defined by the kernel $\tilde{x}_x$, and $\tilde{x}_y$ is the Sobel operator for vertical gradients that is defined by the kernel $\tilde{x}_y$.

$$\tilde{x}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \tilde{x}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

By integrating L_{sobel} into our architecture, the resulting stylized image exhibits enhanced preservation of complex outlines and linear structures. As noted in recent research, the Sobel operator is "a widely employed gradient-based technique" for edge detection that utilizes convolution with two 3x3 kernels where one kernel is used for detecting horizontal edges and the other one is used for vertical edges (BenHajyoussef & Saidani, 2023). This quality improvement occurs without compromising stylistic transfer fidelity, as Sobel detection functions specifically target high-frequency structural information. As a discrete differentiation operator, it computes an approximation of the gradient of the image intensity function. Unlike the Laplacian approach, which "relies on the second derivative of the image intensity", the Sobel method provides directional awareness that better preserves architectural elements (Malche, 2024). Our experimental evidence also demonstrates that substituting the isotropic Laplacian loss with the direction-aware Sobel gradient approach produces images with more defined edges and reduced geometric distortions.

5.5 Proposed Total Variation Loss Function

Sometimes, the learned synthesized image has a lot of high frequency noise such as bright or dark pixels. Since the original total variation loss function is designed for gray images, we have improved it so that it can work on colored images as well.

$$L_{\text{tv}} = \frac{1}{B \cdot C \cdot H \cdot W} \sum_{b=1}^{B} \sum_{c=1}^{C} \sum_{h=1}^{H} \sum_{w=1}^{W} [(\tilde{x}_{b,c,h,w+1} - \tilde{x}_{b,c,h,w})^2 + (\tilde{x}_{b,c,h+1,w} - \tilde{x}_{b,c,h,w})^2]$$

where B is the batch size, C is the number of channels, H is the height of the feature maps, W is the width of the feature maps, $\tilde{x}$ represents the generated image, $\tilde{x}_{b,c,h,w}$ denotes the pixel value at position b, c, h, w, $\tilde{x}_{b,c,h,w+1}$ represents the adjacent pixel in the horizontal direction, and $\tilde{x}_{b,c,h+1,w}$ represents the adjacent pixel in the vertical direction.

According to the Johnson and his research team, their findings demonstrate that the total variation loss results "spatial smoothness" in the generated image (Johnson et al., 2016). Hence, the TV loss effectively reduces undesirable artifacts during the style transfer process because it penalizes rapid changes between adjacent pixels.

After, we can combine our proposed total variation loss and Sobel edge loss function with the multi-layer content loss to create the total loss function to generate realistic images by preserving edges and straight lines as well as the overall structure:

$$L_{\text{total}} = \alpha \cdot L_{\text{content}} + \beta \cdot L_{\text{style}} + \gamma \cdot L_{\text{sobel}} + \delta \cdot L_{\text{tv}}$$

where L_{content} is the multi-layer content loss function L_{style} is the style loss function, L_{sobel} is the Sobel edge loss function, L_{tv} is the total variation loss function, α is the parameter to control the influence of the content loss function, β is the parameter to control the influence of the style loss function, γ is the parameter to control the influence of the Sobel edge loss function, and δ controls the total variation loss.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 1, 2016, pp. 2414–2423. doi:10.1109/CVPR.2016.265

[2] K. Liu, G. Yuan, H. Wu, and W. Qian, "Coarse-to-Fine Structure-Aware Artistic Style Transfer," *Applied Sciences*, vol. 13, no. 2, p. 952, 2023. doi:10.3390/app13020952

[3] C. Lad, "Understanding Neural Style Transfer – Analytics Vidhya," Medium blog, Analytics Vidhya, Dec. 2019. https://medium.com/analytics-vidhya/understanding-neural-style-transfer-3061cd92648

[4] P. Baheti, "Neural Style Transfer: Everything You Need to Know [Guide]," V7 Labs blog, Sept. 8, 2021. https://www.v7labs.com/blog/neural-style-transfer

[5] A. S. Kotiyal, N. Choudhary, and D. K. S. Kumar, "Neural Style Transfer with Structure Preservation of Content Image," *International Journal of Engineering Research & Technology*, vol. 9, no. 7, 2021. https://www.ijert.org/neural-style-transfer-with-structure-preservation-of-content-image

[6] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*, arXiv preprint arXiv:2106.11342 [cs], 2021. https://arxiv.org/abs/2106.11342

[7] G. Seif and D. Androutsos, "Edge-Based Loss Function for Single Image Super-Resolution," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2018. doi:10.1109/ICASSP.2018.8461664

[8] X. Wang, "Laplacian Operator-Based Edge Detectors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 886–890, 2007. doi:10.1109/TPAMI.2007.1027

[9] A. BenHajyoussef and A. Saidani, "Recent Advances on Image Edge Detection," InTechOpen, Jan. 15, 2024. doi:10.5772/intechopen.1003763

[10] T. Malche, "Edge Detection in Image Processing: An Introduction," Roboflow Blog, June 14, 2024. https://blog.roboflow.com/edge-detection/

[11] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, "Laplacian/Laplacian of Gaussian," Spatial Filters – Laplacian/Laplacian of Gaussian. https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

[12] M. Reimann, B. Buchheim, A. Semmo, J. Döllner, and M. Trapp, "Controlling Strokes in Fast Neural Style Transfer Using Content Transforms," *The Visual Computer*, 2022. doi:10.1007/s00371-022-02518-x

[13] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *Proc. European Conf. on Computer Vision (ECCV)*, 2016. https://arxiv.org/abs/1603.08155